

RM05/3/2

RM05/3/2 EXT'D DR TS AH-S074A-MC
CZRMVA0 FICHE 1 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



The main body of the document is a microfiche card containing a grid of approximately 15 columns and 15 rows of data. Each cell in the grid contains a small, dense table of information, likely representing a data point or a specific record. The text is extremely small and difficult to read, but the overall structure is a regular grid of data points.

RM05/3/2

RM05/3/2 EXT'D DR TS AH-S074A-MC
CZRMVAO FICHE 2 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



[Faded, illegible text columns on the left side of the page]

[Small, illegible text mark in the bottom right corner]

.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-S072A-MC
PRODUCT NAME: CZRMVAO RM05/3/2 EXT'D DR TST
DATE: APRIL 1980
MAINTAINER: CX DIAGNOSTIC ENGINEERING
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 TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1980 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 RH11 - RH70 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 12 - 15) 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 ACCESS TIMES ARE WITHIN TOLERANCE, THAT THE TRACK AND SECTOR ADDRESSING CIRCUITRY OPERATES PROPERLY, AND THAT THE DATA STORAGE AND RETRIEVAL CAPABILITIES ARE FUNCTIONING.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
16K MEMORY
TELETYPE
PROGRAM LOADING DEVICE
KW11-L OR KW11-P (THE KW11-P IS REQUIRED FOR THE TIMING TESTS)
1 TO 8 DISK DRIVES (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

2.2 PRELIMINARY PROGRAMS

RM05/3/2 DISKLESS CONTROLLER TEST, PART 1 & 2
RM05/3/2 FUNCTIONAL CONTROLLER TEST, PART 1, 2 & 3

2.3 MEDIA

THE PROGRAM REQUIRES THAT EACH DRIVE TO BE TESTED HAS A FORMATTED DISK PACK. THE PACK 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 ASSOCIATED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS BEING RUN ON A PROCESSOR WITH 16K, THE 'XXDP' LOADER WILL NOT BE PRESERVED. THE PROGRAM MUST BE RUN ON A SYSTEM WITH 20K OR MORE TO PRESERVE THE 'XXDP' LOADER. THE 'ABSOLUTE' LOADER WILL BE PRESERVED IN A 16K SYSTEM, HOWEVER.

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 RH11-RH70 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. LOAD A FORMATTED PACK INTO DRIVE(S) TO BE TESTED
3. BRING DRIVE(S) TO ONLINE STATE, WRITE ENABLED, AND LOCKED ON PORT.
4. LOAD ADDRESS 200.
5. SET SWITCHES (SEE SECTION 5.)
6. PRESS START.
7. 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

115 AND TEST NUMBERS.
116
117 </> SLASH
118
119 A MODIFICATION INDICATOR: IF A SLASH FOLLOWS A TEST
120 NUMBER, THE PROGRAM WILL OPEN THAT TEST FOR PARAMETER
121 MODIFICATION.
122
123 <^U> CONTROL-U
124
125 DELETE THE PRESENT INPUT STRING AND START A NEW
126 LINE. TYPED BY DEPRESSING THE "CONTROL KEY"
127 (CTRL) AND THEN STRIKING THE 'U'.
128
129 <\> RUBOUT
130
131 DELETE THE LAST CHARACTER FROM THE INPUT STRING.
132 TYPED BY STRIKING THE 'RUBOUT' KEY. WHICH WILL
133 BE ECHOED BY A BACKSLASH (\) FOLLOWED BY THE
134 CHARACTER DELETED.
135
136
137

138 4.3.1 CONTROL SWITCH SELECTION

139
140 STARTING THE PROGRAM AT ANY OF THE POSSIBLE STARTING ADDRESSES
141 WITH SW<07>=1 WILL RESULT IN ENTERING THE "CONTROL SWITCH
142 SETTING" MODE. THUS, ALLOWING THE OPERATOR TO SPECIFY THE
143 DESIRED STATE OF "C.SWR".
144

145 CONTROL SWITCH SELECTION EXAMPLES:

146 EXAMPLE #1

147
148 SET SW<07>=0
149 C.SWR=000000 / 400..
150

151 EXAMPLE #2

152
153 SET SW<07>=0
154 C.SWR=000000 / 220.
155 C.SWR=000000 / 220..
156
157

158 4.3.2 RH11 - RH70 ADDRESS SELECTION

159
160 STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC
161 SELECT OF THE DEFAULT VALUES OF BUS ADDRESS (RMCS1),
162 VECTOR ADDRESS, AND PRIORITY LEVEL OF THE RH11-RH70.
163 IF THE DEFAULT VAULE OF THE BUS ADDRESS DOES NOT RESPOND
164 (TIMES OUT) WHEN ADDRESSED, AN ERROR IS REPORTED.
165 AFTER THE ERROR IS REPORTED ONE OF TWO COURSES OF ACTION
166 WILL BE TAKEN:
167

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

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR TO CHANGE THE ADDRESS OF THE RH11 OR RH70 AND THE VECTOR ADDRESS.

ADDRESS SELECTION EXAMPLES

EXAMPLE #1

RMCS1=176700 / 177200.

EXAMPLE #2

RMCS1=176700 / 176300<CR>
RMVEC=254 / 260<CR>
RMPRIO=5 / 6.

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>
RMPRIO=5<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.

EACH TEST CONTAINS TWO SETS OF TRACK LIMIT PARAMETERS. PARAMETERS "LT", "FT" AND "IT" ARE USED BY RM03/2 DRIVES AND PARAMETERS "LT'", "FT'", AND "IT'" ARE USED BY RM05 DRIVES. THE PROGRAM DETERMINES WHICH DRIVE IS BEING TESTED AND SELECTS THE CORRECT SET OF TRACK LIMIT VALUES. IF THE PROGRAM IS BEING USED TO TEST A SUBSYSTEM WHICH CONTAINS BOTH RM03/2 AND RM05 DRIVES, THE OPERATOR MUST CHANGE BOTH SETS OF TRACK LIMITS IF THE TESTS ARE TO BE MODIFIED FOR ALL DRIVES TESTED.

4.3.3.1 DRIVE AND PARAMETER SELECTION DESCRIPTION

THE FOLLOWING IS A TABLE OF TERMS USED BY THE PS:

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

```

''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
''FT''' FIRST TRACK ADDRESS :RM05 PARAMETER
''LT''' LAST TRACK ADDRESS :RM05 PARAMETER
''IT''' INCREMENT TRACK :RM05 PARAMETER
''FS'' FIRST SECTOR ADDRESS
''LS'' LAST SECTOR ADDRESS
''PAT'' PATTERN (USED FOR DATA TEST)
''WDX'' WORD OF PATTERN 0 WHERE X IS 1 TO 16

*'S'' ALL SEEK TESTS (TESTS 0 - 10)
*'T'' ALL TIMING TESTS (TESTS 12 - 15)
*'A'' ALL ADDRESS TESTS (TESTS 16 - 17)
*'D'' THE DATA TEST (TEST 20)
*'E'' THE EXERCISE (TEST 21)
    
```

* USED BY THE OPERATOR TO SELECT TEST GROUPS

NOTE: ALL NUMBERS WILL BE IN DECIMAL EXCEPT FOR THE PATTERN (PAT) AND WORDS (WDX) SELECTION. 'PAT' WILL BE SELECTED BY A BIT (I.E. 001000(8)=PATTERN 9) AND 'WDX' WILL BE IN OCTAL.

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 0-10, 12-20 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=

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

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

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
```


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

FC=0 / <CR> ;DO NOT ALTER THIS LINE JUT CONTINUE
LC=822 /

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING 822 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= 822 / 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
R=1 / <CR>
FC=0 / <CR>
LC= 822 / 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

```
400
401 DRIVE=4.<CR> ;SELECT DRIVE #4, TERMINATE AND
402 ;BEGIN EXECUTION USING PREVIOUSLY ASSIGNED
403 ;PARAMETERS
404
405 EXAMPLE #2
406 -----
407
408 DRIVE=0<CR> ;SELECT DRIVE #0 AND MAKE 'CHANGES ''
409 TEST=1-5.<CR> ;RUN TEST 1 THRU 5 ONLY, USE DEFAULT
410 ;PARAMETERS AND TERMINATE AND EXECUTE.''
411
412 EXAMPLE #3
413 -----
414
415 DRIVE=2<CR> ;SELECT DRIVE #2 AND MAKE CHANGES ''
416 TEST=1-5,6/7/10/<CR> ;RUN TEST 1-5 WITH DEFAULT PARAMETERS, OPEN
417 TEST 6 ;TEST 6,7 AND 10 FOR CHANGES
418 R=1 / <CR> ;LEAVE 'R' AS IS AND MOVE TO NEXT PARAMETER
419 FC=0 / 10.<CR> ;SET 'FC' CYLINDER ADDRESS TO 10, END CHANGES
420 ;TO TEST 6.
421
422 TEST 7
423 R=1 / 50<CR> ;50 ITERATIONS, MOVE TO NEXT PARAMETER
424 FC=0 / <CR> ;DO NOT CHANGE 'FC' CYLINDER ADDRESS BUT CONTINUE
425 LC=822 / 50..<CR> ;TEST 10 IS STILL PENDING AND WILL BE
426 ;RETAIN ITS PRESENT PARAMETERS.
427
428
429 EXAMPLE #4
430 -----
431
432 DRIVE=0<CR> ;SELECT DRIVE #0 AND MAKE CHANGES
433 TEST=S,E.<CR> ;RUN ALL SEEK TESTS AND THE EXERCISER
434
435 EXAMPLE #5
436 -----
437
438 DRIVE=1<CR>
439 TEST=S/D<CR> ;RUN ALL SEEK TESTS (OPEN FOR CHANGES) AND
440 TEST 0 ;THE DATA TEST (WITH DEFAULT PARAMETERS).
441 R=10 / <CR> ;RUN WITH 10 ITERATIONS
442 FC=0 / 10..<CR> ;CHANGE FIRST CYLINDER ADDRESS
443 ;AND START EXECUTION
444 ;TESTS 1 - 10 WILL RETAIN THEIR PREVIOUSLY
445 ;ASSIGNED PARAMETERS.
446
447 EXAMPLE #6
448 -----
449
450 DRIVE=1<CR>
451 TEST=S/<CR> ;OPEN THE SEEK TESTS (TESTS 0-10)
452 TEST 0
453 R=10 / 100.<CR> ;CHANGE TO 100 ITERATIONS, TO TO THE NEXT TEST
454 TEST 1
455 R=100 / 1000.<CR> ;CHANGE 'R' TO 1000 ITERATIONS, MOVE TO NEXT TEST
456 TEST 2
```

```

457 R=1 / 10<CR> ;CHANGE 'R' TO 10 ITERATIONS, GO TO NEXT PARAMETER
458 FC=0 / 50<CR> ;CHANGE 'FC' TO 50, GO TO NEXT PARAMETER
459 LC=822 / 51.<CR> ;CHANGE 'LC' TO 51, GO TO THE NEXT TEST
460 TEST 3
461 R=1.<CR> ;MOVE TO NEXT TEST
462 TEST 4
463 R=1..<CR> ;USE TEST 4'S PARAMETERS AND START PROGRAM EXECUTION
464
465
466
467

```

EXAMPLE #7

```

468 DRIVE=1<CR>
469 TEST=D/<CR> ;SELECT AND OPEN THE DATA TEST
470 TEST 20
471 :1 / 1000<CR> ;DO 1000 ITERATION OF TEST PATTERN
472 FC=0 / 10<CR> ;#8 ON CYLINDER 10, TRACK 2, SECTOR 4
473 LC=822 / 10<CR>
474 IC=64 / 0<CR>
475 FT=0 / 2<CR>
476 LT=4 / 2<CR>
477 IT=1 / <CR>
478 FT'=0 / 2<CR> ;RM05 PARAMETER
479 LT'=18 / 2<CR> ;RM05 PARAMETER
480 IT'=1 / <CR> ;RM05 PARAMETER
481 FS=0 / 4<CR>
482 LS=22 / 4<CR>
483 PAT=177777 / 400..<CR> ;RUN WITH PATTERN #8
484
485
486
487

```

EXAMPLE #8

```

488 DRIVE=1<CR> ;USE THE SAME PARAMETERS AS IN EXAMPLE
489 TEST=D/<CR> ;#7, BUT ALSO SPECIFY A DATA PATTERN (PAT #0).
490 TEST 20
491 R=1000 / <CR>
492 FC=10 / <CR>
493 LC=10 / <CR>
494 IC=0 / <CR>
495 FT=2 / <CR>
496 LT=2 / <CR>
497 IT=1 / <CR>
498 FT'=0 / 2<CR> ;RM05 PARAMETER
499 LT'=18 / 2<CR> ;RM05 PARAMETER
500 IT'=1 / <CR> ;RM05 PARAMETER
501 FS=4 / <CR>
502 LS=4 / <CR>
503 PAT=000400 / 401<CR> ;RUN WITH PATTERNS #8 & #0 (0=OPERATOR INPUT)
504 WD1=165555 / 125252<CR> ;FIRST WORD OF PATTERN 0
505 WD2=133333 / 52525..<CR> ;SECOND WORD OF PATTERN 0
506 ;<..> START EXECUTION
507
508
509

```

EXAMPLE #9

```

510
511 DRIVE=0,1,4<CR> ;TEST DRIVES 0,1, AND 4 IN SEQUENCE
512 TEST=0-5/<CR> ;CHANGE TEST 5
513 TEST 0

```

514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570

R=10 / <CR>
FC=0 / <CR>
LC=822 / 1..<CR> ;CHANGE LAST CYLINDER FROM 822 TO 1
;START PROGRAM EXECUTION.

5. SWITCH SETTINGS

5.1 OPERATIONAL SWITCH SETTINGS

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

SW<15>=1...HALT ON ERROR
SW<14>=1...LOOP ON TEST
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<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 12-15)
SW<05>=1...REPORT ONE ERROR PER SECTOR (TESTS 16 & 17)
SW<04>=1...INHIBIT WRITES (TEST 20)
SW<03>=1...INHIBIT WRITE CHECKS (TEST 20)
SW<02>=1...INHIBIT READ AND SOFTWARE COMPARES (TEST 20)
SW<01>=1...INHIBIT SOFTWARE COMPARES (TEST 20)
SW<00>=1...PERFORM READ AFTER WRITE CHECK ERROR (TEST 20)

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34)
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 (8). 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 AT A HIGHER PRIORITY PROCESSING AN RM05/3/2 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 SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER
IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED., 'RUBOUT' AND
'ONTROL 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 16 SWITCHES IN THE
'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE
'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST
BE FOLLOWED.

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

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:

- L.SWR<15>=0...WRITE PACK BEFORE TESTING (TEST16)
- =1...INHIBIT WRITE PACK BEFORE TESTING (TEST16)
- 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
- C.SWR<08>=0...DO IMPLIED SEEKS WITH DATA TRANSFERS
- =1...DO EXPLICIT SEEKS BEFORE DATA TRANSFERS
- C.SWR<07>=0...DO READ HEADER AND DATA COMMANDS IN TESTS 0-7
- =1...DO EXPLICIT SEEK COMMANDS IN TESTS 0-7
- C.SWR<06>=0...60 HZ POWER SOURCE
- =1...50 HZ POWER SOURCE
- C.SWR<05>=0...ALLOW SOFTWARE TIMEOUTS(ENABLE WATCHDOG TIMER)
- =1...INHIBIT SOFTWARE TIMEOUTS(DISABLE WATCHDOG TIMER)
- C.SWR<00>=0...OPERATE IN 32. SECTOR (16 BIT) MODE
- =1...OPERATE IN 30. SECTOR (18 BIT) MODE

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

REFER TO 4.3.1 FOR C.SWR SELECTION

6. ERRORS

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

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

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

6.1 ERROR TYPES

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

6.1.1 DRIVER ERROR

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

6.1.2 NON-FATAL ERRORS

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

6.1.3 FATAL ERRORS

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 PROTECT FOR

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

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

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 15 MINUTES TO MAKE ONE PASS WITH RM03/2 DRIVES AND APPROXIMATELY 16.5 MINUTES TO A PASS WITH RM05 DRIVES. THIS ASSUMES THE DEFAULT TEST SEQUENCE (TESTS 0-10, 12-20) AND DEFAULT TEST PARAMETERS.

8.2 STACK POINTER

THE STACK POINTER IS INITIALLY SET TO 1100.

8.3 TIMING TESTS (TESTS 12-15) PRINTOUTS

AT THE COMPLETION OF EACH OF THE TIMING TESTS THE TIME OF THE MINIMUM SEEK, MAXIMUM SEEK, AND THE AVERAGE OF ALL OF THE 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.

NOTE: THE PROGRAM STALLS FOR 2 MILLISECONDS BETWEEN SEEK ORDERS, THIS STALL TIME IS NOT INCLUDED IN THE CALCULATED SEEK TIMES. THE SEEK TIMES SPECIFIED ARE POSITIONER MOVEMENT TIMES ONLY AND ARE NOT A MEASUREMENT OF EFFECTIVE SEEK TIME.

8.3.1 TIMING TOLERANCES

1. TEST 12 -- ROTATIONAL SPEED TIMES

--TIMES FOR RM05/3 DRIVES--

60 HZ
MINIMUM=16340 US
MAXIMUM=17000 US
NOMINAL=16670 US

50 HZ
MINIMUM=16250 US
MAXIMUM=17090 US
NOMINAL=16670 US

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

--TIMES FOR RM02 DRIVES--

60 HZ
MINIMUM=24500 US
MAXIMUM=25500 US
NOMINAL=25000 US

50 HZ
MINIMUM=24370 US
MAXIMUM=25630 US
NOMINAL=25000 US

2. TEST 13 -- ONE CYLINDER SEEK TIMES

MAXIMUM=7000 US
NOMINAL=6000 US

3. TEST 14 -- ACCESS TIME MEASUREMENT

MAXIMUM=32000 US
NOMINAL=30000 US

4. TEST 15 -- MAXIMUM SEEK TIMES

MAXIMUM=57000 US
NOMINAL=55000 US

8.3.2. TIMING TESTS PRINTOUT EXAMPLES

EXAMPLE #1

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

ONE CYLINDER SEEK TIMES
* FORWARD
MIN=5350 US
MAX=6920 US
AVG=5550 US 821 SEEKS TIMED
* REVERSE
MIN=5140 US
MAX=5960 US
AVG=5430 US 822 SEEKS TIMED

ACCESS TIME MEASUREMENTS
* FORWARD
MIN=27770 US
MAX=28640 US
AVG=28230 US 128 SEEKS TIMED
* REVERSE
MIN=27990 US
MAX=28550 US

799 AVG=28220 US 128 SEEKS TIMED
800
801 MAXIMUM SEEK TIMES
802 * FORWARD
803 MIN=49990 US
804 MAX=51980 US
805 AVG=51010 US 128 SEEKS TIMED
806 * REVERSE
807 MIN=48120 US
808 MAX=50650 US
809 AVG=49340 US 128 SEEKS TIMED
810
811
812 EXAMPLE #2
813 -----
814
815 ROTATIONAL SPEED TIMES
816 MIN=16670 US
817 MAX=16690 US
818 AVG=16680 US 10 SEEKS TIMED
819
820 ONE CYLINDER SEEK TIMES
821 * FORWARD
822 MIN=5470 US
823 MAX=7940 US 3 ABOVE THE MAXIMUM OF 7000 US
824 AVG=5830 US 821 SEEKS TIMED
825 * REVERSE
826 MIN=5040 US
827 MAX=5970 US
828 AVG=5330 US 822 SEEKS TIMED
829
830 ACCESS TIME MEASUREMENTS
831 * FORWARD
832 MIN=29730 US
833 MAX=32620 US 73 ABOVE THE MAXIMUM OF 32000 US
834 AVG=30320 US 128 SEEKS TIMED
835 * REVERSE
836 MIN=28620
837 MAX=32230 US 128 ABOVE THE MAXIMUM OF 32000 US
838 AVG=32800 US 128 SEEKS TIMED
839
840 MAXIMUM SEEK TIMES
841 * FORWARD
842 MIN=57510 US
843 MAX=57240 US 128 ABOVE THE MAXIMUM OF 57000 US
844 AVG=57020 US 128 SEEKS TIMED
845 * REVERSE
846 MIN=57050 US
847 MAX=57550 US 128 ABOVE THE MAXIMUM OF 57000 US
848 AVG=57210 US 128 SEEKS TIMED
849
850
851 8.4 END OF TEST
852
853 WITH ALL SWITCHES ON A '0' AN 'END OF PASS' MESSAGE WILL BE
854 TYPED AT THE COMPLETION OF TESTING A DRIVE AND THE 'END OF TEST'
855 TYPEOUT WILL OCCUR WHEN ALL DRIVES HAVE BEEN TESTED.

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

9. PROGRAM DESCRIPTION

THIS PROGRAM CONTAINS NINETEEN TESTS NUMBERED 0-22 IN OCTAL. TESTS 0-7 & 11 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 12-15 WILL MEASURE THE ROTATIONAL SPEED, THE ONE CYLINDER SEEK, THE ACCESS TIME, AND THE MAXIMUM SEEK TIMES TO ENSURE THEY ARE ALL WITHIN THE TOLERANCES ALLOWED. TEST 16 AND 17 ENSURES THE SECTOR AND TRACK ADDRESSING CIRCUITRY WORKS PROPERLY. TEST 20 VERIFIES THE DATA STORAGE AND RETRIEVAL CAPABILITIES ARE FUNCTIONAL. AND TEST 21 WILL STRESS AND CHECK THE READ/WRITE AND SERVO SYSTEMS.

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 0-10,12-20) 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/SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE COMMAND CYCLE AND THEN SEEK FORWARD TO CYLINDER 'LC'. AT THE COMPLETION OF BOTH COMMANDS, STATUS INDICATIONS ARE CHECKED TO ENSURE NO ERRORS OCCURRED.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	200
LC	-	822
FT	-	0
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.

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

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	100
FC	-	0
LC	-	256
IC	-	0
FT	-	0
LT	-	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	-	822
IC	-	1
FT	-	0
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
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

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

EXAMINED TO ENSURE PROPER OPERATION.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	1
FT	-	0
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	-	822
IC	-	1
FT	-	0
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	-	822
IC	-	1
FT	-	0
FT'	-	0
FS	-	0

9.8 TEST 7 - RANDOM SEEK TEST

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

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
1083

OF POSITIONING OCCURS USING EACH HEAD. TRACK ADDRESSES ARE INCREMENTED BETWEEN PARAMTERS 'FT' AND 'LT'.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	5000
FC	-	0
LC	-	822
FT	-	0
LT	-	4
FT'	-	0
LT'	-	18

9.9 TEST 10 - SERVO SETTLE DOWN TEST

THIS TEST VERIFIES THAT THE SERVO HAS SETTLED DOWN AND THAT THE DRIVE IS ON CYLINDER WHEN THE DRIVE INDICATES SEEK COMPLETE. RANDOM SEEKS ARE ISSUED BETWEEN CYLINDERS 'NC1' AND 'NC1+IC' ('NC1' STARTS AT VALUE 'FC'). AT THE COMPLETION OF 1000 (10) SEEKS, 'NC1' IS INCREMENTED BY VALUE 'IC' AND THE SEQUENCE IS REPEATED. THE TEST IS COMPLETED WHEN 'NC1' HAS BEEN INCREMENTED BEYOND 'LC'.

WHEN THE SEEK COMPLETES, THE PROGRAM READS THE DRIVE'S LOOK-AHEAD REGISTER (RMLA) TO DETERMINE THE ADDRESS OF THE SECTOR ROTATING INTO POSITION. THE PROGRAM THEN ISSUES A WRITE HEADER AND DATA COMMAND FOR THAT SECTOR. ERRORS IN THIS TEST INDICATE THAT THE SERVO SYSTEM MAY NOT BE ADJUSTED CORRECTLY, THAT THE DRIVE IS MALFUNCTIONING, OR THAT A PACK WITH MARGINAL SERVO TRACKS IS MOUNTED ON THE DRIVE.

THIS TEST IS VALID ONLY IF THE OPERATION IS STARTED WITHIN A FEW HUNDRED MICRO-SECONDS AFTER SEEK DONE OCCURS. THE NECESSARY TIME DEPENDENT PARAMETERS OCCUR WITHIN THE REQUIRED TIME RANGE FREQUENTLY ENOUGH TO PERMIT THIS TEST TO BE EFFECTIVE.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	822
IC	-	100
FT	-	0
FT'	-	0

9.10 TEST 11 - 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
---	---	---

1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140

FC	-	0
LC	-	822
IC	-	1
FT	-	0
FT'	-	0
FS	-	0

9.11 TEST 12 - ROTATIONAL SPEED TIMING TEST

THIS TEST WILL START A SEARCH TO CYLINDER 'FC', TRACK 'FT', SECTOR 'FS'. AS SOON AS THE INTERRUPT OCCURS, THE GC 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:

RM05/3:
16.67 MS/REV + OR - 2% IF 60HZ
16.67 MS/REV + OR - 2.5% IF 50HZ.

RM02:
25.00 MS/REV + OR - 2% IF 60HZ
25.00 MS/REV + OR - 2.5% IF 50HZ.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

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

9.12 TEST 13 - 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. THE TIME MUST BE LESS THAN 10MS.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	822

9.13 TEST 14 - ACCESS TIME MEASUREMENT

THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO CYLINDER 'LC', THEN A REVERSEK FROM CYLINDER 'LC' TO CYLINDER 'FC'. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE ACCESS TIME MEASUREMENT. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS). THE AVERAGE ACCESS TIME MUST BE LESS THAN 30 MS.

THE TEST USES THE FOLLOWING PARAMETERS:

1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197

R - 1
FC - 0
LC - 255

9.14 TEST 15 - 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 AND 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). THE MAXIMUM SEEK TIME MUST BE LESS THAN 54 MS.

THE TEST USES THE FOLLOWING PARAMETERS:

R - 1
FC - 0
LC - 822

9.15 TEST 16 - SECTOR ADDRESSING TEST

THIS TEST WRITES DATA INTO ALL SECTORS OF TRACK 'FT'. THE DATA WILL BE 256 WORDS OF THE SECTOR ADDRESS OF THE SECTOR BEING WRITTEN. A WRITE CHECK IS PERFORMED, THE BUFFER IS CLEARED (TO 177400) AND THE DATA IS READ AND COMPARED. THEN SECTOR 0 IS REWRITTEN AND SECTORS 0 - 31 ARE WRITE CHECKED. THEN SECTOR 1 IS REWRITTEN AND SECTORS 0 - 31 ARE WRITE CHECKED. THIS REWRITE AND WRITE CHECK PROCEDURE IS CONTINUED UP THROUGH REWRITE SECTOR 31 AND WRITE CHECK SECTORS 0-31.

THE TEST USES THE FOLLOWING PARAMETERS:

R - 1
FC - 0
FT - 0
FT' - 0

9.16 TEST 17 - TRACK ADDRESSING TEST

THIS TEST WILL WRITE DATA IN THE FORM OF TRACK ADDRESSES IN CYLINDER 'FC' SECTOR 'FS' OF EVERY TRACK WITH EACH TRACK GETTING ITS OWN TRACK ADDRESS. A WRITE CHECK IS THEN PERFORMED ON EACH TRACK TO INSURE THE DATA IS VALID. THEN TRACK 0 IS REWRITTEN AND TRACK 1 THROUGH LAST TRACK IS WRITE CHECKED. THEN TRACK 1 IS REWRITTEN AND TRACK 2 THROUGH LAST TRACK IS WRITE CHECKED. THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING NEXT TO LAST TRACK AND WRITE CHECKING LAST TRACK.

THE TEST USES THE FOLLOWING PARAMETERS:

R - 1
FC - 0
FS - 0

9.17 TEST 20 - DATA TEST

1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254

THIS TEST PERFORMS DATA STORAGE AND RETRIEVAL ON CYLINDERS 'FC' THROUGH 'LC' BY THE INCREMENT 'IC' USING THE DATA PATTERNS SPECIFIED. THE FOLLOWING SEQUENCE OCCURS FOR EACH CYLINDER:

1. SET 'NT' TO 'FT' THEN REPEAT 2-4 UNTIL 'NT' > 'LT'
2. WRITE THEN WRITE CHECK 'FS' THROUGH 'LS' OF TRACK 'NT'
3. READ THEN SOFTWARE COMPARE 'FS' THROUGH 'LS' OF TRACK 'NT'
4. INCREMENT 'NT' BY 'IT'
5. REPEAT STEPS 1-4 FOR EACH DATA PATTERN
6. REPEAT STEPS 1-5 FOR 'FC' THROUGH 'LC' ADVANCING BY 'IC'

IF A WRITE CHECK ERROR OCCURS THE ERROR IS REPORTED AND THE TRACK IN ERROR IS REWRITTEN AND CHECKED. THIS CHECK IS ACCOMPLISHED BY PERFORMING TWO(2) SUCCESSIVE ERROR FREE WRITE CHECKS. IF THE CHECK FAILS THE ERROR IS REPORTED AS FATAL AND NO READ OCCURS.
FS DEFAULTS TO 1 AND LS DEFAULTS TO 0
PAT DEFAULTS TO 17777 (ALL POSSIBLE PATTERNS)
THE POSSIBLE PATTERNS ARE:

PAT 0	PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7
165555	000001	177776	000000	000000	052525	007417	026455
133333	000003	177774	000000	010421	052525	007417	026455
165555	000007	177770	000000	021042	052525	007417	026455
133333	000017	177760	177777	031463	125252	170360	151322
165555	000037	177740	177777	042104	125252	170360	151322
133333	000077	177700	177777	052525	125252	170360	151322
165555	000177	177600	000000	063146	052525	007417	026455
133333	000377	177400	000000	073567	052525	007417	026455
165555	000777	177000	177777	104210	125252	170360	151322
133333	001777	176000	177777	114631	125252	170360	151322
165555	003777	174000	000000	125252	052525	007417	026455
133333	007777	170000	177777	135673	125252	170360	151322
165555	017777	160000	000000	146314	052525	007417	026455
133333	037777	140000	177777	156735	125252	170360	151322
165555	077777	100000	000000	167356	052525	007417	026455
133333	177777	000000	177777	177777	125252	170360	151322
PAT 8	PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
165555	000001	177776	172666	077777	153333	000000	177777
133333	000002	177775	155555	137777	066667	177777	000000
165555	000004	177773	172666	157777	153333	177777	000000
133333	000010	177767	155555	167777	066667	177777	000000
165555	000020	177757	172666	173777	153333	177777	000000
133333	000040	177737	155555	175777	066667	177777	000000
165555	000100	177677	172666	176777	153333	177777	000000
133333	000200	177577	155555	177377	066667	177777	000000
165555	000400	177377	172666	177577	153333	177777	000000
133333	001000	176777	155555	177677	066667	177777	000000
165555	002000	175777	172666	177737	153333	177777	000000
133333	004000	173777	155555	177757	066667	177777	000000
165555	010000	167777	172666	177767	153333	177777	000000
133333	020000	157777	155555	177773	066667	177777	000000
165555	040000	137777	172666	177775	153333	177777	000000
133333	100000	077777	155555	177776	066667	177777	000000

1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	822
IC	-	64
FT	-	0
LT	-	4
IT	-	1
FT'	-	0
LT'	-	18
IT'	-	1
FS	-	1
LS	-	0
PAT	-	177777

9.18 TEST 21 - RANDOM ADDRESS AND RANDOM PATTERN TEST

STARTING AT 'FC' AND GOING THROUGH 'LC' THE DISK PACK IS WRITTEN WITH A RANDOM PATTERN. THE FIRST TWO WORDS OF EACH SECTOR WILL BE THE BASE OF THE RANDOM GENERATOR FOR THAT SECTOR. THE TEST THEN PERFORMS THE FOLLOWING SEQUENCE 'R' TIMES 'R' DEFAULTS TO 20,000.

- 1) GENERATE A RANDOM ADDRESS
- 2) WRITE A RANDOM PATTERN AT THE ADDRESS GENERATED IN 1.
- 3) GENERATE A RANDOM ADDRESS
- 4) READ THE SECTOR AT THE ADDRESS GENERATED IN 3.
- 5) DO A SOFTWARE CHECK OF THE DATA READ IN 4.
- 6) DO A WRITE CHECK OF THE DATA WRITTEN IN 2
- 7) GENERATE A RANDOM ADDRESS
- 8) READ THE SECTOR AT THE ADDRESS GENERATED IN 7.
- 9) DO A SOFTWARE CHECK OF THE DATA READ IN 8
- 10) DO A WRITE CHECK OF THE DATA WRITTEN IN 2

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	20000
FC	-	0
LC	-	822

9.19 TEST 22 - ACCESS TIME ADJUSTMENT TEST

THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 255 TO ALLOW THE OPERATOR TO ADJUST THE ACCESS TIME ON AN RM05/3/2 USING THE DDU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS SO THAT THE ACCESS TIME INDICATORS ON THE DDU MAY BE OBSERVED.

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	5000
FC	-	0

1312
1313
1314
1315
1316
1317
1318

LC - 255

10. PROGRAM LISTING

@

1
44
45

46

47

48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```

:PROGRAM REVISION #001
.TITLE CZRMVAO RM05/3/2 EXT'D DR TST
*COPYRIGHT (C) 1980
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY MIKE LEAVITT
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.
*
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
SBTTL OPERATIONAL SWITCH SETTINGS
*
*      SWITCH      USE
*      -----
*      15          HALT ON ERROR
*      14          LOOP ON TEST
*      13          INHIBIT ERROR TYPEOUTS
*      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 12-15)
*      5           REPORT ONE ERROR PER SECTOR (TESTS 16 & 17)
*      4           INHIBIT WRITES (TEST 20)
*      3           INHIBIT WRITE CHECKS (TEST 20)
*      2           INHIBIT READ AND SOFTWARE COMPARES (TEST 20)
*      1           INHIBIT SOFTWARE COMPARES (TEST 20)
*      0           PERFORM READ AFTER WRITE CHECK ERROR (TEST 20)
*
SBTTL CONTROL SWITCH SETTINGS
*
*      SWITCH  STATE  USE
*      -----  -----
*      15      0      WRITE PACK BEFORE TESTING (TEST 21)
*      14      1      INHIBIT WRITING PACK BEFORE TESTING (TEST 21)
*      14      0      NO STALL BETWEEN DRIVE FUNCTIONS
*      13      1      STALL AFTER EVERY DRIVE FUNCTION
*      13      0      USE SPECIFIC STALL TIME
*      12      1      USE RANDOM STALL
*      12      0      NO INCREMENTING STALL IN TEST 4
*      8       1      DO INCREMENTING STALL IN TEST 4
*      8       0      DO IMPLIED SEEKS WITH DATA TRANSFERS
*      7       1      DO EXPLICIT SEEKS BEFORE DATA TRANSFERS
*      7       0      DO 'READ HEADER AND DATA' IN TESTS 0-11
*      7       1      DO EXPLICIT SEEKS IN TESTS 0-11

```

73
74
75
76
77
78
79
80

```

:*      6      0      60 HZ
:*      1      1      50 HZ
:*      5      0      RUN WATCHDOG TIMER
:*      1      1      INHIBIT WATCHDOG TIMER
:*      0      0      TEST DRIVE(S) IN 32. SECTOR (16 BIT) MODE
:*      1      1      TEST DRIVE(S) IN 30. SECTOR (18 BIT) MODE
    
```

.SBTTL BASIC DEFINITIONS

```

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

```

000011  ;*MISCELLANEOUS DEFINITIONS
000012  HT = 11          ;;CODE FOR HORIZONTAL TAB
000015  LF = 12          ;;CODE FOR LINE FEED
000200  CR = 15          ;;CODE FOR CARRIAGE RETURN
177776  CRLF = 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776  PS = 177776     ;;PROCESSOR STATUS WORD
177774  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
    
```

```

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

```

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

```

100000  ;*'SWITCH REGISTER' SWITCH DEFINITIONS
040000  SW15 = 100000
020000  SW14 = 40000
010000  SW13 = 20000
004000  SW12 = 10000
002000  SW11 = 4000
001000  SW10 = 2000
000400  SW09 = 1000
        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
    
```

```

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

```

; *BASIC "CPU" TRAP VECTOR ADDRESSES
000004      ERRVEC   = 4           ;; TIME OUT AND OTHER ERRORS
000010      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

81
82          .SBTTL  RM11/RM70 REGISTERS
83
84          ;CONTROL AND STATUS REGISTER 1 (RMCS1)
85
86          000100      IE      = 100          ;INTERRUPT ENABLE (BIT #6)
87          000200      RDY     = 200          ;READY (BIT #7)
88          000400      A16    = 400          ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
89          001000      A17    = 1000         ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
90          002000      PSEL   = 2000         ;PORT SELECT (BIT #10)
91          020000      MCPE   = 20000        ;MASSBUS PARITY ERROR (BIT #13)
92          040000      TRE    = 40000        ;TRANSFER ERROR (BIT #14)
93          ;SC      = 100000          ;SPECIAL CONDITION (BIT #15)
94
95          ;WORD COUNT REGISTER (RMWC)
96          ;(EACH BIT IS CALLED BY BIT NUMBER)
97
98          ;BUS ADDRESS REGISTER (RMBA)
99          ;(EACH BIT IS CALLED BY BIT NUMBER)
100
101         ;CONTROL AND STATUS REGISTER 2 (RMCS2)
102
103         000001      US1     = 1            ;UNIT SELECT (BIT #0)
104         000002      US2     = 2            ;UNIT SELECT (BIT #1)
105         000004      US4     = 4            ;UNIT SELECT (BIT #2)
106         000010      BAI     = 10           ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
107         ;PAT      = 20            ;MASSBUS PARITY TEST (BIT #4)
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  RM 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 (FMR1) (#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)
167
168      ;MAINTAINABILITY REGISTER #01 (RMMR1)(#03) - READ ONLY BITS
169
170      000001      DMD     = 1      ;DIAGNOSTIC MODE
171      000002      LSIT    = 2
172      ;LS          = 4
173      000010      WD      = 10
174      000020      EECC    = 20
175      000040      WC      = 40
176      000100      CONT    = 100
177      000200      PHA     = 200
178      000400      PDA     = 400
179      001000      ECRC    = 1000
180      002000      PLFS    = 2000
181      004000      ESRC    = 4000
182      010000      REX     = 10000
183      020000      EBL     = 20000
184      ;R/G        = 40000
185      100000      OCC     = 100000
186
187      ;MAINTAINABILITY REGISTER #01 (RMMR1) (#03) - WRITE ONLY BITS
188
189      000001      DMD     = 1      ;DIAGNOSTIC MODE BIT
190      000002      MSC     = 2
191      000004      MI      = 4
  
```


192	000010	MWP	= 10	
193	000020	DTG	= 20	
194	000040	MS	= 40	
195	000100	MDF	= 100	
196	000200	MSER	= 200	
197	000400	MOC	= 400	
198	001000	MUR	= 1000	
199	002000	MRD	= 2000	
200	004000	MCLK	= 4000	
201	010000	MSEN	= 10000	
202	020000	DT0	= 20000	
203	040000	OBEN	= 40000	
204	100000	OBCK	= 100000	
205				
206		;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)		
207				
208	000001	AT0	= 1	;DEVICE 0 (BIT #0)
209	000002	AT1	= 2	;DEVICE 1 (BIT #1)
210	000004	AT2	= 4	;DEVICE 2 (BIT #2)
211	000010	AT3	= 10	;DEVICE 3 (BIT #3)
212	000020	AT4	= 20	;DEVICE 4 (BIT #4)
213	000040	AT5	= 40	;DEVICE 5 (BIT #5)
214	000100	AT6	= 100	;DEVICE 6 (BIT #6)
215	000200	AT7	= 200	;DEVICE 7 (BIT #7)
216				
217		;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)		
218		;(EACH BIT IS CALLED BY BIT NUMBER)		
219				
220		;DRIVE TYPE REGISTER (RIIDT) (#06)		
221				
222	000001	DT00	= 1	;DRIVE TYPE NUMBER BIT 1
223	000002	DT01	= 2	;DRIVE TYPE NUMBER BIT 2
224	000004	DT02	= 4	;DRIVE TYPE NUMBER BIT 3
225	000010	DT03	= 10	;DRIVE TYPE NUMBER BIT 4
226	000020	DT04	= 20	;DRIVE TYPE NUMBER BIT 5
227	000040	DT05	= 40	;DRIVE TYPE NUMBER BIT 6
228	000100	DT06	= 100	;DRIVE TYPE NUMBER BIT 7
229	000200	DT07	= 200	;DRIVE TYPE NUMBER BIT 8
230	000400	DT08	= 400	;DRIVE TYPE NUMBER BIT 9
231	004000	DRQ	= 4000	;DRIVE REQUEST REQUIRED (BIT #11)
232	020000	MOH	= 20000	;MOVING HEAD (BIT #13)
233	040000	TAP	= 40000	;TAPE DRIVE (BIT #14)
234	100000	NBA	= 100000	;NOT BLOCK ADDRESSED (BIT #15)
235				
236		;LOOK-AHEAD REGISTER (RMLA) (#07)		
237				
238				
239	000100	SC0	= 100	;SECTOR COUNT FIELD 0 (BIT #6)
240	000200	SC1	= 200	;SECTOR COUNT FIELD 1 (BIT #7)
241	000400	SC2	= 400	;SECTOR COUNT FIELD 2 (BIT #8)
242		;SC3	= 1000	;SECTOR COUNT FIELD 3 (BIT #9)
243		;SC4	= 2000	;SECTOR COUNT FIELD 4 (BIT #10)
244				
245				
246		;RM MAINTAINABILITY REGISTER #2 (RMMR2) (#10)		
247				
248		;RM ERROR REGISTER #02 (RMMR2) (#10)		

```

249
250           ;OFFSET REGISTER (RMOF) (#11)
251
252           000200 OFD      = 200           ;OFFSET DIRECTION (BIT #07)
253           002000 HCI      = 2000          ;HEADER COMPARE INHIBIT (BIT #10)
254           004000 ECI      = 4000          ;ERROR CORRECTION CODE INHIBIT (BIT #11)
255           010000 FMT16   = 10000         ;FORMAT BIT (BIT #12)
256
257           ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
258           ;(EACH BIT IS CALLED BY BIT NUMBER)
259
260           ;CURRENT CYLINDER ADDRESS (RMHR) (#13)
261           ;(EACH BIT IS CALLED BY BIT NUMBER)
262
263           ;SERIAL NUMBER REGISTER (RMSN) (#14)
264           ;(EACH IS CALLED BY BIT NUMBER)
265
266
267           ;RM ERROR REGISTER #02 (RMER2) (#15)
268
269           020000 OPE      = 20000         ;OPERATOR PLUG ERROR (BIT #13)
270           040000 SKI      = 40000         ;SEEK INCOMPLETE (BIT #15)
271
272           ;ECC POSITION REGISTER (RMEC1) (#16)
273           ;(EACH BIT IS CALLED BY BIT NUMBER)
274
275           ;ECC PATTERN REGISTER (RMEC2) (#17)
276           ;(EACH BIT IS CALLED BY BIT NUMBER)
277
278           ;*****
279
280           ;OP CODE DEFINITIONS
281           000101 NOOP     = 101
282           000103 UNLOAD   = 103
283           000105 SEEK     = 105
284           000107 RECAL    = 107
285           000111 DRVCLR   = 111
286           000113 RELEASE  = 113
287           000115 OFFSET   = 115
288           000117 RTC      = 117
289           000121 READIN   = 121
290           000123 PACK     = 123
291           000131 SEARCH   = 131
292           000151 WRCKD    = 151
293           000153 WRCKHD   = 153
294           000161 WRITE    = 161
295           000163 WRTHD    = 163
296           000171 READ     = 171
297           000173 READHD   = 173
298           000141 GETREG   = 141
299           000143 SETFORM  = 143
300           000145 SELDRV   = 145
301
302           ;OTHER EQUATES
303
304           177400 SCTRW    = -256.         ;WORD COUNT FOR SECTOR
305

```

307
308

.SBTTL TRAP CATCHER

000000

. = 0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000174 000174
000174 000000
000176 000000

. = 174
DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)

309 000200 000137 004570
310 000204 000137 004612
311
312 000210 000137 004560
313
314 000214 000137 004602
315
316

JMP @#START1 ;:JUMP TO STARTING ADDRESS OF PROGRAM
JMP @#START2 ;:SELECT OPERATING PARAMETERS
JMP @#START3 ;:SELECT RH/RM ADDRESSES
JMP @#START4 ;:COMBINATION OF 204 AND 210

.SBTTL ACT11 HOOKS

;HOOKS REQUIRED BY ACT11

000220
000046 000046
000052 021244
000052 000052
000220 020000
000220

\$SVPC= . ;SAVE PC
. = 46
\$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
. = 52
.WORD 20000 ;:2)SET LOC.52 TO 20000
.\$SVPC ;:RESTORE PC

317
318
319

001100

. = 1100
.SBTTL APT PARAMETER BLOCK

;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

001100
000024 000024
000024 000200
000044 000044
000044 001100
001100

.\$X= . ;:SAVE CURRENT LOCATION
. = 24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;:FOR APT START UP
. = 44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
\$APTHDR ;:POINT TO APT HEADER BLOCK
.\$X ;:RESET LOCATION COUNTER

;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

001100
001100 000000
001102 001234
001104 001604
001106 001604
001110 001604
001112 000J30
001114

\$APTHD:
\$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
\$MBADR: .WORD \$MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
\$STMT: .WORD 900. ;:RUN TIM OF LONGEST TEST
\$PASTM: .WORD 900. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 900. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
.\$ETEND-\$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
TAB.XY= . ;:COMMON TAG STARTING ADDRESS

320
321

520
524

0

.SBTTL COMMON TAGS

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

001114	001114				
001114	000000	\$CMTAG:	.=TAB.XY		::START OF COMMON TAGS
001116	000	\$STSTM:	.WORD 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	\$GDDAT:	.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	\$SWR:	.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:	.ASCIZ /?/		::QUESTION MARK
001231	015	\$CRLF:	.ASCIZ <15>		::CARRIAGE RETURN
001232	012	\$LF:	.ASCIZ <12>		::LINE FEED

377

377

000

 .SBTTL APT MAILBOX-ETABLE

```

*****
.EVEN
001234          $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  $$WREG: .WORD   ASWREG  ;; APT SWITCH REGISTER
001260 000000  $USWR: .WORD   AUSWR   ;; USER SWITCHES
001262 000000  $CPUOP: .WORD   ACPUOP  ;; CPU TYPE, OPTIONS
                ;;
                ;; BITS 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 PRCESSOR
                ;; BIT 8=MEMORY MANAGEMENT
001264          $MAMS1: .BYTE   AMAMS1 ;; HIGH ADDRESS,M.S. BYTE
001265          $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          $MAMS2: .BYTE   AMAMS2 ;; HIGH ADDRESS,M.S. BYTE
001271          $MTYP2: .BYTE   AMTYP2 ;; MEM.TYPE,BLK#2
001272 000000  $MADR2: .WORD   AMADR2 ;; MEM.LAST ADDRESS,BLK#2
001274          $MAMS3: .BYTE   AMAMS3 ;; HIGH ADDRESS,M.S.BYTE
001275          $MTYP3: .BYTE   AMTYP3 ;; MEM.TYPE,BLK#3
001276 000000  $MADR3: .WORD   AMADR3 ;; MEM.LAST ADDRESS,BLK#3
001300          $MAMS4: .BYTE   AMAMS4 ;; HIGH ADDRESS,M.S.BYTE
001301          $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#2BUS PRIORITY#2
001310 000000  $BASE: .WORD   ABASE  ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
001312 000000  $DEVVM: .WORD   ADEVVM ;; DEVICE MAP
001314          $ETEND:
                .MEXIT

```

```

0      .SBTTL  USER DEFINED TAGS

001314 000000 C.SWR: .WORD 0 ;CONTROL SWITCHES
001316 000000 SAVCSW: .WORD 0 ;PREVIOUS CONTENTS OF 'C.SWR'
001320 000000 CNTRLC: .WORD 0 ;CONTROL 'C' FLAG
001322 000000 BUSADR: .WORD 0 ;GET ADDRESSES FROM THE TTY FLAG (0=NO, -1=YES)
001324 000000 LPTAVL: .WORD 0 ;LPT AVAILABLE STATUS (0=NO,1=YES)
001326 000000 DRVSEL: .WORD 0 ;DRIVES SELECTED FOR TESTING
001330 176777 000003 TSTNMS: .WORD 176777,3 ;DEFAULT IS RUN TESTS 0-10 & 12-21
001334 000000 000000 OPNFLG: .WORD 0,0 ;MODIFY TEST PARAMETER FLAGS
001340 000000 CLKSTA: .WORD 0 ;CLOCK STATUS (0=NO CLOCK,+1=KW11-P,
;AND -1=KW11-L)
;16 MILLISECONDS PER CLOCK TICK
;16666 MIRCSECONDS PER CLOCK TICK

001342 000020 TICKMS: .WORD 16.
001344 040432 TICKUS: .WORD 16666.
001346 000000 BYPASS: .WORD 0
001350 000000 CHKDRV: .WORD 0 ;DRIVE UNDER TEST
001352 000000 DRVMSK: .WORD 0 ;DRIVE MASK BIT
001354 000000 SVSTAT: .WORD 0 ;STATUS/ERROR INDICATOR IS
;SAVED HERE ON AN ERROR

001356 000000 CYL.RD: .WORD 0 ;CYLINDER READ
001360 000000 TRK.RD: .WORD 0 ;TRACK READ
001362 000000 SEC.RD: .WORD 0 ;SECTOR READ
001364 000000 CYL.DS: .WORD 0 ;CYLINDER DESIRED
001366 000000 SEC.DS: .WORD 0 ;SECTOR DESIRED
001370 000000 TRK.D_: .WORD 0 ;TRACK DESIRED
001372 000000 LSTRK: .WORD 0 ;CONTAINS THE LAST TRACK OF THE UNIT UNDER
;TEST. RM02/3 = 4., RMC5 = 18.

001374 000000 TIM.UP: .WORD 0 ;MINIMUM TIME
001376 000000 .WORD 0 ;NUMBER OF COUNTS BELOW MIN. LIMIT
001400 000000 .WORD 0 ;MAXIMUM TIME
001402 000000 .WORD 0 ;NUMBER OF COUNTS ABOVE MAX. LIMIT
001404 000000 000000 .WORD 0,0 ;TOTAL TIME OF ALL SEEKS
001410 000000 .WORD 0 ;NUMBER OF SEEKS PERFORMED
001412 000000 TIM.DN: .WORD 0 ;MINIMUM TIME
001414 000000 .WORD 0 ;NUMBER OF COUNTS BELOW MIN. LIMIT
001416 000000 .WORD 0 ;MAXIMUM TIME
001420 000000 .WORD 0 ;NUMBER OF COUNTS ABOVE MAX. LIMIT
001422 000000 000000 .WORD 0,0 ;TOTAL TIME OF ALL SEEKS
001426 000000 .WORD 0 ;NUMBER OF SEEKS PERFORMED
001430 000000 TIM.PT: .WORD 0 ;POINTS TO TABLE OF TIMES
001432 000000 WCEFLG: .WORD 0 ;FATAL WRITE CHECK ERROR FLAG
001434 000000 STALLO: .WORD 0 ;VARIABLE STALL
001436 000000 000000 SVADR: .WORD 0,0 ;SAVE DISK ADDRESS
001442 000000 SEKTMR: .WORD 0 ;SEEK TIMER
001444 000000 SEKCNT: .WORD 0 ;SEEK COUNTER
001446 000000 DELTA: .WORD 0 ;TESTING RANGE FOR SERVO SETTLE DOWN TEST
001450 160000 TRCKWC: .WORD -<256.*32.> ;WORD COUNT FOR A FULL TRACK IN 16 BIT MODE
001452 000012 STALL1: .WORD 10. ;10 MILLISECONDS STALL
001454 000012 STALL2: .WORD 10. ;10 MILLISECONDS STALL
001456 011610 STALL3: .WORD 5000. ;5 SEC STALL
001460 000031 MXSTAL: .WORD 25. ;MAX. INCREMENTING STALL ALLOWED IN TEST 4
001462 144 ERR.CT: .BYTE 100. ;NUMBER OF ERRORS ALLOWED IN TESTS 16 - 21
;BEFORE GOING TO THE NEXT TEST
;RESERVED

001463 000 .BYTE 0
001464 000J00 BASFLG: .WORD 0 ;FLAG FOR DETECTING BAD SECTOR
001466 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 RM05/3/2.

001470 176700
 001472 000254 000240
 001476 000104 000106
 001502 172540
 001504 172542
 001506 172544
 001510 000100 000102
 001514 177546
 001516 177564
 001520 177566
 001522 177514
 001524 177516

;ADDRESSES AND VECTORS
 RH.ADR: .WORD 176700
 RHVEC: .WORD 254,240
 PKV: .WORD 104,106
 PKCS: .WORD 172540
 PKB: .WORD 172542
 PKC: .WORD 172544
 LKV: .WORD 100,102
 LKS: .WORD 177546
 TPS: .WORD 177564
 TPB: .WORD 177566
 LPS: .WORD 177514
 LPB: .WORD 177516

;RH/RM UNIBUS ADDRESS
 ;VECTOR ADDRESS AND PRIORITY
 ;KW11-P VECTOR ADDRESS
 ;KW11-P CONTROL AND STATUS REG.
 ;KW11-P COUNT SET BUFFER
 ;KW11-P COUNTER
 ;KW11-L VECTOR ADDRESS
 ;KW11-L STATUS REGISTER
 ;TTY PRINTER STATUS
 ;TTY PRINTER BUFFER
 ;LINE PRINTER STATUS
 ;LINE PRINTER BUFFER

001526 000001
 001530 000002
 001532 000004
 001534 000010
 001536 000020
 001540 000040
 001542 000100
 001544 000200
 001546 000400
 001550 001000
 001552 002000
 001554 004000
 001556 010000
 001560 020000
 001562 040000
 001564 100000
 001566 000001
 001570 000002
 001572 000004
 001574 000010
 001576 000020
 001600 000040
 001602 000100
 001604 000200

;BIT TABLE
 BITS: .WORD BIT00
 .WORD BIT01
 .WORD BIT02
 .WORD BIT03
 .WORD BIT04
 .WORD BIT05
 .WORD BIT06
 .WORD BIT07
 .WORD BIT08
 .WORD BIT09
 .WORD BIT10
 .WORD BIT11
 .WORD BIT12
 .WORD BIT13
 .WORD BIT14
 .WORD BIT15
 .WORD BIT00
 .WORD BIT01
 .WORD BIT02
 .WORD BIT03
 .WORD BIT04
 .WORD BIT05
 .WORD BIT06
 .WORD BIT07

.SBTTL TIMING LIMITS

;ROTATIONAL TEST TABLES FOR RM05/3 DRIVES

;60HZ TABLE

001606 046261
 001610 000000
 001612 003142
 001614 003244

T7A: .WORD ROTATE
 .WORD 0
 .WORD 1634.
 .WORD 1700.

;LO LIMIT (16.67MS - 2%)
 ;HI LIMIT (16.67MS + 2%)

;50HZ TABLE

001616 046261
 001620 000000
 001622 003131
 001624 003255

T7B: .WORD ROTATE
 .WORD 0
 .WORD 1625.
 .WORD 1709. ;

;LO LIMIT (16.67MS - 2.5%)
 ;HI LIMIT (16.67MS + 2.5%)

;ROTATIONAL TEST TABLES FOR RM02 DRIVES

```

;60HZ TABLE
001626 046261 T7A1: .WORD ROTATE
001630 000000 .WORD 0
001632 004622 .WORD 2450. ;LO LIMIT (25.00MS - 2%)
001634 004766 .WORD 2550. ;HI LIMIT (25.00MS + 2%)
    
```

```

;50HZ TABLE
001636 046261 T7B1: .WORD ROTATE
001640 000000 .WORD 0
001642 004605 .WORD 2437. ;LO LIMIT (25.00MS - 2.5%)
001644 005003 .WORD 2563. ;HI LIMIT (25.00MS + 2.5%)
    
```

;SEEK TEST TABLES

```

001646 046312 T10: .WORD ONECYL ;FORWARD
001650 046464 .WORD REV ;REVERSE
001652 000000 .WORD 0 ;NO LO LIMIT
001654 001274 .WORD 700. ;HI LIMIT (7MS)
    
```

```

001656 046357 T11: .WORD ACCESS ;FORWARD
001660 046464 .WORD REV ;REVERSE
001662 000000 .WORD 0 ;NO LO LIMIT
001664 006200 .WORD 3200. ;HI LIMIT (32MS)
    
```

```

001666 046424 T12: .WORD MXSEEK ;FORWARD
001670 046464 .WORD REV ;REVERSE
001672 000000 .WORD 0 ;NO LO LIMIT
001674 013104 .WORD 5700. ;HI LIMIT (57MS)
    
```

;SPECS. MESSAGE TABLES FOR ROTATIONAL AND TIMING TESTS

;ROTATIONAL MESSAGE AND LO/HI LIMITS FOR RM05/3 DRIVES

```

;60HZ TABLE
001676 046662 SP7A: .WORD MSG7XA
001700 003142 .WORD 1634. ;LO LIMIT (16.67MS - 2%)
001702 003244 .WORD 1700. ;HI LIMIT (16.67MS + 2%)
    
```

```

;50HZ TABLE
001704 046662 SP7B: .WORD MSG7XA
001706 003131 .WORD 1625. ;LO LIMIT (16.67MS - 2.5%)
001710 003255 .WORD 1709. ;HI LIMIT (16.67MS + 2.5%)
    
```

;ROTATIONAL MESSAGE AND LO/HI LIMITS FOR RM02 DRIVES

```

;60HZ TABLE
001712 046737 SP7A1: .WORD MSG7XB
001714 004622 .WORD 2450. ;LO LIMIT (25.00MS - 2%)
001716 004766 .WORD 2550. ;HI LIMIT (25.00MS + 2%)
    
```

```

;50HZ TABLE
001720 046737 SP7B1: .WORD MSG7XB
001722 004605 .WORD 2437. ;LO LIMIT (25.00MS - 2.5%)
001724 005003 .WORD 2563. ;HI LIMIT (25.00MS + 2.5%)
    
```

;TIMING TESTS MESSAGES AND LO/HI LIMITS

```

001726 047012 SP10: .WORD MSG10X
001730 000000 .WORD 0 ;NO LO LIMIT
    
```

001732	001274		.WORD	700.		;HI LIMIT (7MS)
001734	047054	SP11:	.WORD	MSG11X		
001736	000000		.WORD	0		;NO LO LIMIT
001740	006200		.WORD	3200.		;HI LIMIT (32MS)
001742	047110	SP12:	.WORD	MSG12X		
001744	000000		.WORD	0		;NO LO LIMIT
001746	013104		.WORD	5700.		;HI LIMIT (57MS)

001750	050012	;STATUS/ERROR MESSAGE POINTER TABLE				
001752	050054	STATBL:	.WORD	MSGB14		;OFFLINE OR UNSAFE DRIVE REQUESTED
001754	050105		.WORD	MSGB13		;UNLOAD DRIVE REQUESTED
001756	050127		.WORD	MSGB12		;PERSISTENT UNSAFE
001760	050155		.WORD	MSGB11		;PARITY ERROR OCCURRED
001762	050200		.WORD	MSGB10		;FATAL PARITY ERROR
001764	050237		.WORD	MSGB09		;SOFTWARE TIMEOUT ON THIS DRIVE
001766	050301		.WORD	MSGB08		;SOFTWARE TIMEOUT ON ANOTHER DRIVE
001770	050345		.WORD	MSGB06		;ERROR OCCURRED DURING I/O OPERATION
001772	050415		.WORD	MSGB05		;ERROR OCCURRED DURING NON-I/O OPERATION
001774	050435		.WORD	MSGB04		;UNSAFE OCCURRED
001776	050505		.WORD	MSGB03		;AUTOMATIC RECALIBRATE SEQUENCE OCCURRED
002000	050555		.WORD	MSGB02		;DRIVE HAS NOT RESPONDED TO PORT REQUEST
			.WORD	MSGB01		;DRIVE HAS BECOME NONEXISTENT

0

.SBTTL ERROR POINTER TABLE

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

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

002002

\$ERRTB:

;*ERROR 1

;* RH/RM 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+6

EM2
 DH2
 DT2
 DF2

;*ERROR 3

;* MASSBUS PARITY ERROR (MCPE=1)
 ;* TEST ERR PC ADDRESS DATA
 ;* \$TMP0 \$ERRPC RD.ADR RD.WRD

EM3
 DH3
 DT3
 DF3

;*ERROR 4

;* MASSBUS PARITY ERROR (PAR=1)
 ;* TEST ERR PC ADDRESS GDDATA BDDATA
 ;* \$TMP0 \$ERRPC WRT.ADR WRT.WD RD.WRD

EM4
 DH4

1
 5
 6
 7
 8
 9
 10
 11 002002 047152
 12 002004 050613
 13 002006 052170
 14 002010 052622
 15
 16
 17
 18
 19
 20
 21
 22 002012 047216
 23 002014 050630
 24 002016 052174
 25 002020 052626
 26
 27
 28
 29
 30
 31
 32
 33 002022 047254
 34 002024 050715
 35 002026 052212
 36 002030 052632
 37
 38
 39
 40
 41
 42
 43
 44 002032 047311
 45 002034 050752

46	002036	052222	DT4
47	002040	052636	DF4
48			
49			;*ERROR 5
50			
51			;* ADDRESS PLUG CHANGE BIT SET
52			;* ERR PC DRIVE RMAS RMDS RMER1 RMMR2 RMER2
53			;* \$ERRPC \$REG1 \$REG3 RMERRS RMERRS+2 RMERRS+4 RMERRS+6
54			
55	002042	047345	EM5
56	002044	050630	DH2
57	002046	052174	DT2
58	002050	052626	DF2
59			
60			;*ERROR 6 -- NOT USED
61			
62	002052	000000	0
63	002054	000000	0
64	002056	000000	0
65	002060	000000	0
66			
67			;*ERROR 7 -- NOT USED
68			
69	002062	000000	0
70	002064	000000	0
71	002066	000000	0
72	002070	000000	0
73			
74			;*ERROR 10
75			
76			;* RH/RM FAILED TO RESPOND TO ADDRESSING
77			;* RMCS1 ERR PC
78			;* RH.ADR \$ERRPC
79			
80	002072	047401	EM10
81	002074	051021	DH10
82	002076	052254	DT10
83	002100	052642	DF10
84			
85			;*ERROR 11
86			
87			;* DRIVE SELECTED IS NOT ONLINE
88			;* DRIVE ERR PC
89			;* \$REG2 \$ERRPC
90			
91	002102	047447	EM11
92	002104	051040	DH11
93	002106	052260	DT11
94	002110	052646	DF11
95			
96			;*ERROR 12
97			
98			;* IMPROPER HEADER DATA
99			;* TEST ERR PC TST PC DRIVE CYLNR TRACK SECTOR
100			;* \$TMP0 \$ERRPC \$REG0 CHKDRV CYL.DS TRK.DS SEC.DS
101			;* GDCYL GDTRK GDSCR BDCYL BDRK BDSCR
102			;* CYL.DS TRK.DS SEC.DS CYL.RD TRK.RD SEC.RD

```

103          ;*   CYLNDR, TRACK, AND SECTOR ARE DECIMAL
104
105 002112   047504          EM12
106 002114   051057          DH12
107 002116   052264          DT12
108 002120   052652          DF12
109
110          ;*ERROR 13
111
112          ;*   DATA COMPARE FAILURE
113          ;*   TEST   ERR PC TST PC DRIVE  CYLNDR TRACK  SECTOR
114          ;*   $TMPO  $ERRPC $REGO  CHKDRV  CYL.DS  TRK.DS  SEC.DS
115          ;*           GDDAT  BDDAT  WRDCNT  GDADR  BDADR
116          ;*           $GDDAT $BDDAT $REG4  $GDADR $BDADR
117          ;*   CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
118
119 002122   047531          EM13
120 002124   051057          DH12
121 002126   052316          DT13
122 002130   052662          DF13
123
124          ;*ERROR 14 -- FOLLOWS #13
125
126          ;*           $GDDAT $BDDAT $REG4  $GDADR $BDADR
127
128 002132   000000          0
129 002134   000000          0
130 002136   052334          DT13A
131 002140   052672          DF14
132
133          ;*ERROR 15
134
135          ;*   DATA COMPARE FAILURE
136          ;*   TEST   ERR PC TST PC DRIVE  CYLNDR TRACK  SECTOR
137          ;*   $TMPO  $ERRPC $REGO  CHKDRV  CYL.DS  TRK.DS  SEC.DS
138          ;*           GDDAT  BDDAT  WRDCNT  GDADR  BDADR
139          ;*           $GDDAT $BDDAT $REG4  $GDADR $BDADR
140          ;*   CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
141
142 002142   047531          EM13
143 002144   051057          DH12
144 002146   052316          DT13
145 002150   052662          DF13
146
147          ;*ERROR 16 -- FOLLOWS #15
148
149          ;* $GDDAT $BDDAT $REG4  $GDADR $BDADR
150
151 002152   000000          0
152 002154   000000          0
153 002156   052334          DT13A
154 002160   052672          DF14
155
156          ;*ERROR 17
157
158          ;*   DISK ERROR IN TIMING TEST
159          ;*   TEST   ERR PC DRIVE  RMCS1  RMDS  RMER1  RMMR2  RMER2
  
```

```

160          ;*   $TMP0   $ERRPC  CHKDRV  RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
161
162 002162   047556          EM17
163 002164   051273          DH17
164 002166   052346          DT17
165 002170   052676          DF17
166
167          ;*ERROR 20
168
169          ;*   CLOCK (KW11-P) OVERFLOW IN TIMING TEST
170          ;*   TEST     ERR PC  DRIVE  RMCS1  RMDS      RMER1      RMMR2      RMER2
171          ;*   $TMP0   $ERRPC  CHKDRV  RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
172
173 002172   047610          EM20
174 002174   051273          DH17
175 002176   052346          DT17
176 002200   052676          DF17
177
178          ;*ERROR 21
179
180          ;*   DATA COMPARE FAILURE
181          ;*   TEST     ERR PC  TST PC  DRIVE  CYLNDR  TRACK
182          ;*   $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS
183          ;*   GDDAT   BDDAT   WRDCNT  SECTOR
184          ;*   $REG1   $BDDAT  $REG4   $REG1
185          ;*   CYLINDR, TRACK, WRDCNT, AND SECTOR ARE DECIMAL
186
187 002202   047531          EM13
188 002204   051370          DH21
189 002206   052366          DT21
190 002210   052702          DF21
191
192          ;*ERROR 22--FOLLOWS #21
193
194          ;*           $REG1   $BDDAT   $REG4   $REG1
195
196 002212   000000          0
197 002214   000000          0
198 002216   052402          DT21A
199 002220   052712          DF22
200
201          ;*ERROR 23
202
203          ;*   DISK ERROR DURING SEEK
204          ;*   TEST     ERR PC  DRIVE  CYLNDR  RMCS1  RMCS2  RMDS
205          ;*   $TMP0   $ERRPC  CHKDRV  CYL.DS  RM.REG  RM.REG+10 RM.REG+12
206          ;*           RMER1      RMMR2      RMER2      RMDC      RMRH
207          ;*           RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
208
209 002222   047657          EM23
210 002224   051505          DH23
211 002226   052412          DT23
212 002230   052716          DF23
213
214          ;*ERROR 24
215
216          ;*   SEEK NOT COMPLETE WITHIN 120 MS

```



```

217      ;*      TEST      ERR PC  DRIVE  CYLNDR  RMCS1  RMCS2  RMD5
218      ;*      $TMP0     $ERRPC  CHKDRV  CYL.DS  RM.REG  RM.REG+10 RM.REG+12
219      ;*      RMR1      RMMR2   RMR2     RMDC    RMHR
220      ;*      RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
221
222 002232 047706      EM24
223 002234 051505      DH23
224 002236 052412      DT23
225 002240 052716      DF23
226
227      ;*****
228      ;*      ERRORS 23-40 NOT USED
229      ;*      ERRORS 41-46 WILL HAVE AN EM THAT
230
231      ;*      VARIES DEPENDING ON THE ERROR, IT WILL BE IN THE FORM:
232      ;*      RH/RM ERROR (MESSAGE)
233      ;*      WHERE MESSAGE WILL BE ONE OR MORE OF THE FOLLOWING:
234      ;*      1) OFFLINE OR UNSAFE DRIVE REQUESTED
235      ;*      2) UNLOADED DRIVE REQUESTED
236      ;*      3) PERSISTENT UNSAFE
237      ;*      4) PARITY ERROR OCCURRED
238      ;*      5) FATAL PARITY ERROR
239      ;*      6) SOFTWARE TIMEOUT ON THIS DRIVE
240      ;*      7) SOFTWARE TIMEOUT ON ANOTHER DRIVE
241      ;*      8) ERROR OCCURRED DURING I/O OPERATION
242      ;*      9) ERROR OCCURRED DURING NON-I/O OPERATION
243      ;*      10) UNSAFE OCCURRED
244      ;*      11) AUTOMATIC RECALIBRATE SEQUENCE OCCURRED
245      ;*****
246
247 002242      ITEM41:
248
249      ;*ERROR 41
250
251      ;*      RH/RM ERROR (MESSAGE)
252      ;*      TEST      ERR PC  TST PC  DRIVE
253      ;*      $TMP0     $ERRPC  $REGO   CHKDRV
254
255 002242 047746      EM41
256 002244 051637      DH41
257 002246 052442      DT41
258 002250 052726      DF41
259
260      ;*ERROR 42
261
262      ;*      RH/RM ERROR (MESSAGE)
263      ;*      TEST      ERR PC  TST PC  DRIVE  RMCS1  RMCS2  RMD5
264      ;*      $TMP0     $ERRPC  $REGO   CHKDRV  RM.REG  RM.REG+10 RM.REG+12
265
266 002252 047746      EM41
267 002254 051675      DH42
268 002256 052452      DT42
269 002260 052732      DF42
270
271      ;*ERROR 43
272
273      ;*      RH/RM ERROR (MESSAGE)
  
```

```

274      :*      TEST      ERR PC  TST PC  DRIVE  RMCS1  RMCS2  RMDS
275      :*      $TMP0    $ERRPC  $REGO  CHKDRV  RM.REG  RM.REG+10 RM.REG+12
276      :*              RMER1      RMMR2      RMER2
277      :*              RM.REG+14  RM.REG+40  RM.REG+42
278
279 002262 047746      EM41
280 002264 051675      DH42
281 002266 052470      DT43
282 002270 052736      DF43
283
284      :*ERROR 44
285
286      :*      RH/RM ERROR (MESSAGE)
287      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK      SECTOR
288      :*      $TMP0    $ERRPC  $REGO  CHKDRV  CYL.DS  TRK.DS    SEC.DS
289      :*              RMCS1      RMCS2      RMDS      RMHR      RMDC      RMDA
290      :*              RM.REG      RM.REG+10  RM.REG+12  RM.REG+36  RM.REG+34  RM.REG+06
291      :*      RMER1      RMMR2      RMER2
292      :*      RM.REG+14  RM.REG+40  RM.REG+42
293      :*      CYLNDR,TRACK, AND SECTOR ARE DECIMAL
294
295 002272 047746      EM41
296 002274 051057      DH12
297 002276 052514      DT44
298 002300 052746      DF44
299
300      :*ERROR 45
301
302      :*      RH/RM ERROR (MESSAGE)
303      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK      SECTOR
304      :*      $TMP0    $ERRPC  $REGO  CHKDRV  CYL.DS  TRK.DS    SEC.DS
305      :*              RMCS1      RMCS2      RMDS      RMHR      RMDC      RMDA
306      :*              RM.REG      RM.REG+10  RM.REG+12  RM.REG+36  RM.REG+34  RM.REG+06
307      :*      RMER1      RMMR2      RMER2      RMWC      RMBA      RMDB
308      :*      RM.REG+14  RM.REG+40  RM.REG+42  RM.REG+2   RM.REG+4   RM.REG+22
309      :*      CYLNDR,TRACK, AND SECTOR ARE DECIMAL
310
311 002302 047746      EM41
312 002304 051057      DH12
313 002306 052554      DT45
314 002310 052762      DF45
315
316      :*ERROR 46
317
318      :*      FATAL WRITE CHECK ERROR (MESSAGE)
319      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK      SECTOR
320      :*      $TMP0    $ERRPC  $REGO  CHKDRV  CYL.DS  TRK.DS    SEC.DS
321      :*              RMCS1      RMCS2      RMDS      RMHR      RMDC      RMDA
322      :*              RM.REG      RM.REG+10  RM.REG+12  RM.REG+36  RM.REG+34  RM.REG+06
323      :*      RMER1      RMMR2      RMER2      RMWC      RMBA      RMDB
324      :*      RM.REG+14  RM.REG+40  RM.REG+42  RM.REG+2   RM.REG+4   RM.REG+22
325      :*      CYLNDR,TRACK, AND SECTOR ARE DECIMAL
326
327 002312 047762      EM46
328 002314 051057      DH12
329 002316 052554      DT45
330 002320 052762      DF45
  
```



```

1          .SBTTL TEST PARAMETER POINTERS AND TABLES
2
3          ;COMMON STORAGE FOR TEST PARAMETERS
4 002322 000000 PRM: .WORD 0 ;THIS WORD TELLS WHICH OF THE
5                                     ;FOLLOWING PARAMETERS ARE TO BE USED
6 002324 000000 RPT: .WORD 0 ;REPEAT COUNTS FOR ALL TESTS
7 002326 000000 FC: .WORD 0 ;FIRST CYLINDER
8 002330 000000 LC: .WORD 0 ;LAST CYLINDER
9 002332 000000 IC: .WORD 0 ;INCREMENT CYLINDER
10 002334 000000 FT: .WORD 0 ;FIRST TRACK
11 002336 000000 LT: .WORD 0 ;LAST TRACK
12 002340 000000 IT: .WORD 0 ;INCREMENT TRACK
13 002342 000000 FS: .WORD 0 ;FIRST SECTOR
14 002344 000000 LS: .WORD 0 ;LAST SECTOR
15 002346 000000 PAT: .WORD 0 ;PATTERN CODE
16 002350 000000 000000 000000 .WORD 0,0,0 ;FILLER WORDS FOR COMMON TABLE USED BY THE
17                                     ;'OPNTST' ROUTINE.
18
19 002356 000000 NC1: .WORD 0 ;NEW CYLINDER ADDRESS
20 002360 000000 NC2: .WORD 0 ;NEW CYLINDER ADDRESS
21
22          ;TABLE OF PARAMETER POINTERS
23 002362 003116 PRMPT: .WORD PRM0
24 002364 003132 .WORD PRM1
25 002366 003160 .WORD PRM2
26 002370 003200 .WORD PRM3
27 002372 003220 .WORD PRM4
28 002374 003240 .WORD PRM5
29 002376 003260 .WORD PRM6
30 002400 003300 .WORD PRM7
31 002402 003320 .WORD PRM10
32 002404 003336 .WORD PRM11
33 002406 003356 .WORD PRM12
34 002410 003372 .WORD PRM13
35 002412 003402 .WORD PRM14
36 002414 003412 .WORD PRM15
37 002416 003422 .WORD PRM16
38 002420 003434 .WORD PRM17
39 002422 003444 .WORD PRM20
40 002424 003500 .WORD PRM21
41 002426 003510 .WORD PRM22
42 002430 000000 .WORD 0 ;TERMINATOR

          ;TABLE OF PARAMETER UPPER LIMITS
30 002432 032767 PRMLMT: .WORD 32767 ;'R''
31 002434 001466 .WORD 822. ;'FC''
32 002436 001466 .WORD 822. ;'LC''
33 002440 001466 .WORD 822. ;'IC''
34 002442 000004 .WORD 4 ;'FT''
35 002444 000004 .WORD 4 ;'LT''
36 002446 000004 .WORD 4 ;'IT''
37 002450 000022 .WORD 18. ;'FT'''
38 002452 000022 .WORD 18. ;'LT'''
39 002454 000022 .WORD 18. ;'IT'''
40 002456 000037 .WORD 31. ;'FS''
41 002460 000037 .WORD 31. ;'LS''
42 002462 177777 .WORD 177777 ;'PAT''
    
```

43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

002464 045626
002466 045630
002470 045633
002472 045636
002474 045641
002476 045644
002500 045647
002502 045652
002504 045656
002506 045662
002510 045666
002512 045671

002514 002225 000310 001466
002530 006677 000144 000000
002556 002237 000001 000000
002576 002237 000010 000000
002616 002237 000001 000000
002636 002237 000001 000000
002656 002237 000001 000000
002676 000667 011610 000000
002716 000237 000001 000000
002734 002237 000001 000000
002754 002223 000001 000000
002770 000007 000001 000000
003000 000007 000001 000000
003010 000007 000001 000000
003020 000223 000001 000000
003032 002003 000001 000000
003042 017777 000001 000000
003076 000007 001750 000000
003106 000007 011610 000000

003116 002225
003120 000310
003122 001466
003124 000000
003126 000000
003130 000000

003132 006677
003134 000144
003136 000000
003140 000400
003142 000000
003144 000000
003146 000000
003150 000000

:TABLE OF PARAMETER 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 MES.FT
.WORD MES.LT
.WORD MES.IT
.WORD MSG.FS
.WORD MSG.LS

:STATUS/ERPOR INDICATOR MESSAGES POINTER TABLE
:DEFAULT VALUES OF TEST PARAMETERS

DFLT: .WORD 2225,200.,822.,0,0,0 :RECAL/SEEK (T0)
.WORD 6677,100.,0,256.,0,0,0,0,0,0 :SEEK/SEEK (T1)
.WORD 2237,1,0,822.,1,0,0,0 :INCREMENT SEEK (T2)
.WORD 2237,10,0,512.,1,0,0,0 :STEPPING SEEK (T3)
.WORD 2237,1,0,822.,1,0,0,0 :OSCILLATING SEEK (T4)
.WORD 2237,1,0,822.,1,0,0,0 :CONVERGING/DIVERGING SEEK (T5)
.WORD 2237,1,0,822.,1,0,0,0 :SERVO ADDRESSING LOGIC NOISE (T6)
.WORD 667,5000.,0,822.,0,4,0,18. :RANDOM SEEK TEST (T7)
.WORD 237,1,0,822.,100.,0,0 :SERVO SETTLE DOWN TEST (T10)
.WORD 2237,1,0,822.,1,0,0,0 :ALL SEEKS TEST (T11)
.WORD 2223,1,0,0,0,0 :ROTATIONAL SPEED TIMING TEST (T12)
.WORD 7,1,0,822. :ONE CYLINDER SEEK TIMING TEST (T13)
.WORD 7,1,0,255. :ACCESS TIME MEASUREMENT TEST (T14)
.WORD 7,1,0,822. :MAXIMUM SEEK TIMING TEST (T15)
.WORD 223,1,0,0,0 :SECTOR ADDRESSING TEST (T16)
.WORD 2003,1,0,0 :TRACK ADDRESSING TEST (T17)
.WORD 17777,1,0,821.,64.,0,4,1,0,18.,1,1,0,17777 :DATA TEST (T20)
.WORD 7,1000.,0,821. :EXERCISER (T21)
.WORD 7,5000.,0,255. :ACCESS TIME ADJUSTMENT TEST (T22)

:PARAMETER TABLES

:RECAL/SEEK (T0)

PRM0: .WORD 2225
.WORD 200.
.WORD 822.
.WORD 0
.WORD 0
.WORD 0

:SEEK/SEEK (T1)

PRM1: .WORD 6677
.WORD 100.
.WORD 0
.WORD 256.
.WORD 0
.WORD 0
.WORD 0
.WORD 0

```
100 003152 000000          .WORD 0
101 003154 000000          .WORD 0
102 003156 000000          .WORD 0
103
104          ;INCREMENT SEEK (T2)
105 003160 002237          PRM2: .WORD 2237
106 003162 000001          .WORD 1
107 003164 000000          .WORD 0
108 003166 001466          .WORD 822.
109 003170 000001          .WORD 1
110 003172 000000          .WORD 0
111 003174 000000          .WORD 0
112 003176 000000          .WORD 0
113
114          ;STEPPING SEEK (T3)
115 003200 002237          PRM3: .WORD 2237
116 003202 000001          .WORD 1
117 003204 000000          .WORD 0
118 003206 001000          .WORD 512.
119 003210 000001          .WORD 1
120 003212 000000          .WORD 0
121 003214 000000          .WORD 0
122 003216 000000          .WORD 0
123
124          ;OSCILLATING SEEK (T4)
125 003220 002237          PRM4: .WORD 2237
126 003222 000001          .WORD 1
127 003224 000000          .WORD 0
128 003226 001466          .WORD 822.
129 003230 000001          .WORD 1
130 003232 000000          .WORD 0
131 003234 000000          .WORD 0
132 003236 000000          .WORD 0
133
134          ;CONVERGING/DIVERGING SEEK (T5)
135 003240 002237          PRM5: .WORD 2237
136 003242 000001          .WORD 1
137 003244 000000          .WORD 0
138 003246 001466          .WORD 822.
139 003250 000001          .WORD 1
140 003252 000000          .WORD 0
141 003254 000000          .WORD 0
142 003256 000000          .WORD 0
143
144          ;SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)
145 003260 002237          PRM6: .WORD 2237
146 003262 000001          .WORD 1
147 003264 000000          .WORD 0
148 003266 001466          .WORD 822.
149 003270 000001          .WORD 1
150 003272 000000          .WORD 0
151 003274 000000          .WORD 0
152 003276 000000          .WORD 0
153
154          ;RANDOM SEEK TEST (T7)
155 003300 000667          PRM7: .WORD 667
156 003302 011610          .WORD 5000.
```

157	003304	000000	.WORD	0
158	003306	001466	.WORD	822.
159	003310	000000	.WORD	0
160	003312	000004	.WORD	4
161	003314	000000	.WORD	0
162	003316	000022	.WORD	18.
163				
164			;SERVO SETTLE DOWN TEST (T10)	
165	003320	000237	PRM10: .WORD	237
166	003322	000001	.WORD	1
167	003324	000000	.WORD	0
168	003326	001466	.WORD	822.
169	003330	000144	.WORD	100.
170	003332	000000	.WORD	0
171	003334	000000	.WORD	0
172				
173			;ALL SEEKS TEST (T11)	
174	003336	002237	PRM11: .WORD	2237
175	003340	000001	.WORD	1
176	003342	000000	.WORD	0
177	003344	001466	.WORD	822.
178	003346	000001	.WORD	1
179	003350	000000	.WORD	0
180	003352	000000	.WORD	0
181	003354	000000	.WORD	0
182				
183			;ROTATIONAL SPEED TIMING TEST (T12)	
184	003356	002223	PRM12: .WORD	2223
185	003360	000001	.WORD	1
186	003362	000000	.WORD	0
187	003364	000000	.WORD	0
188	003366	000000	.WORD	0
189	003370	000000	.WORD	0
190				
191			;ONE CYLINDER SEEK TIMING TEST (T13)	
192	003372	000007	PRM13: .WORD	7
193	003374	000001	.WORD	1
194	003376	000000	.WORD	0
195	003400	001466	.WORD	822.
196				
197			;ACCESS TIME MEASUREMENT TEST (T14)	
198	003402	000007	PRM14: .WORD	7
199	003404	000001	.WORD	1
200	003406	000000	.WORD	0
201	003410	000377	.WORD	255.
202				
203			;MAXIMUM SEEK TIMING TEST (T15)	
204	003412	000007	PRM15: .WORD	7
205	003414	000001	.WORD	1
206	003416	000000	.WORD	0
207	003420	001466	.WORD	822.
208				
209			;SECTOR ADDRESSING TEST (T16)	
210	003422	000223	PRM16: .WORD	223
211	003424	000001	.WORD	1
212	003426	000000	.WORD	0
213	003430	000000	.WORD	0


```
214 003432 000000 .WORD 0
215
216 ;TRACK ADDRESSING TEST (T17)
217 003434 002003 PRM17: .WORD 2003
218 003436 000001 .WORD 1
219 003440 000000 .WORD 0
220 003442 000000 .WORD 0
221
222 ;DATA TEST (T20)
223 003444 017777 PRM20: .WORD 17777
224 003446 000001 .WORD 1
225 003450 000000 .WORD 0
226 003452 001465 .WORD 821.
227 003454 000100 .WORD 64.
228 003456 000000 .WORD 0
229 003460 000004 .WORD 4
230 003462 000001 .WORD 1
231 003464 000000 .WORD 0
232 003466 000022 .WORD 18.
233 003470 000001 .WORD 1
234 003472 000001 .WORD 1
235 003474 000000 .WORD 0
236 003476 177777 PTRN15: .WORD 177777
237
238 ;EXERCISER (T21)
239 003500 000007 PRM21: .WORD 7
240 003502 001750 .WORD 1000.
241 003504 000000 .WORD 0
242 003506 001465 .WORD 821.
243
244 ;ACCESS TIME ADJUSTMENT TEST (T22)
245 003510 000007 PRM22: .WORD 7
246 003512 011610 .WORD 5000.
247 003514 000000 .WORD 0
248 003516 000377 .WORD 255.
249
```

```

1          .SBTTL DATA PATTERN POINTERS AND TABLES
2
3 003520 003560 PAT.PT: .WORD PAT0          ;DATA PATTERN 0
6 003522 003620 .WORD PAT1          ;DATA PATTERN 1
  003524 003660 .WORD PAT2          ;DATA PATTERN 2
  003526 003720 .WORD PAT3          ;DATA PATTERN 3
  003530 003760 .WORD PAT4          ;DATA PATTERN 4
  003532 004020 .WORD PAT5          ;DATA PATTERN 5
  003534 004060 .WORD PAT6          ;DATA PATTERN 6
  003536 004120 .WORD PAT7          ;DATA PATTERN 7
  003540 004160 .WORD PAT8          ;DATA PATTERN 8
  003542 004220 .WORD PAT9          ;DATA PATTERN 9
  003544 004260 .WORD PAT10         ;DATA PATTERN 10
  003546 004320 .WORD PAT11         ;DATA PATTERN 11
  003550 004360 .WORD PAT12         ;DATA PATTERN 12
  003552 004420 .WORD PAT13         ;DATA PATTERN 13
  003554 004460 .WORD PAT14         ;DATA PATTERN 14
  003556 004520 .WORD PAT15         ;DATA PATTERN 15
7
8 003560 066666 PAT0: .WORD 066666          ;PATTERN 0
9 003562 066666 .WORD 066666
10 003564 066666 .WORD 066666
11 003566 066666 .WORD 066666
12 003570 066666 .WORD 066666
13 003572 066666 .WORD 066666
14 003574 066666 .WORD 066666
15 003576 066666 .WORD 066666
16 003600 066666 .WORD 066666
17 003602 066666 .WORD 066666
18 003604 066666 .WORD 066666
19 003606 066666 .WORD 066666
20 003610 066666 .WORD 066666
21 003612 066666 .WORD 066666
22 003614 066666 .WORD 066666
23 003616 066666 .WORD 066666
24
25 003620 000001 PAT1: .WORD 000001          ;PATTERN 1
26 003622 000003 .WORD 000003
27 003624 000007 .WORD 000007
28 003626 000017 .WORD 000017
29 003630 000037 .WORD 000037
30 003632 000077 .WORD 000077
31 003634 000177 .WORD 000177
32 003636 000377 .WORD 000377
33 003640 000777 .WORD 000777
34 003642 001777 .WORD 001777
35 003644 003777 .WORD 003777
36 003646 007777 .WORD 007777
37 003650 017777 .WORD 017777
38 003652 037777 .WORD 037777
39 003654 077777 .WORD 077777
40 003656 177777 .WORD 177777
41
42 003660 177776 PAT2: .WORD 177776          ;PATTERN 2
43 003662 177774 .WORD 177774
44 003664 177770 .WORD 177770
45 003666 177760 .WORD 177760
    
```

46	003670	177740	.WORD	177740	
47	003672	177700	.WORD	177700	
48	003674	177600	.WORD	177600	
49	003676	177400	.WORD	177400	
50	003700	177000	.WORD	177000	
51	003702	176000	.WORD	176000	
52	003704	174000	.WORD	174000	
53	003706	170000	.WORD	170000	
54	003710	160000	.WORD	160000	
55	003712	140000	.WORD	140000	
56	003714	100000	.WORD	100000	
57	003716	000000	.WORD	000000	
58					
59	003720	000000	PAT3: .WORD	000000	;PATTERN 3
60	003722	000000	.WORD	000000	
61	003724	000000	.WORD	000000	
62	003726	177777	.WORD	177777	
63	003730	177777	.WORD	177777	
64	003732	177777	.WORD	177777	
65	003734	000000	.WORD	000000	
66	003736	000000	.WORD	000000	
67	003740	177777	.WORD	177777	
68	003742	177777	.WORD	177777	
69	003744	000000	.WORD	000000	
70	003746	177777	.WORD	177777	
71	003750	000000	.WORD	000000	
72	003752	177777	.WORD	177777	
73	003754	000000	.WORD	000000	
74	003756	177777	.WORD	177777	
75					
76	003760	000000	PAT4: .WORD	000000	;PATTERN 4
77	003762	010421	.WORD	010421	
78	003764	021042	.WORD	021042	
79	003766	031463	.WORD	031463	
80	003770	042104	.WORD	042104	
81	003772	052525	.WORD	052525	
82	003774	063146	.WORD	063146	
83	003776	073567	.WORD	073567	
84	004000	104210	.WORD	104210	
85	004002	114631	.WORD	114631	
86	004004	125252	.WORD	125252	
87	004006	135673	.WORD	135673	
88	004010	146314	.WORD	146314	
89	004012	156735	.WORD	156735	
90	004014	167356	.WORD	167356	
91	004016	177777	.WORD	177777	
92					
93	004020	052525	PAT5: .WORD	052525	;PATTERN 5
94	004022	052525	.WORD	052525	
95	004024	052525	.WORD	052525	
96	004026	125252	.WORD	125252	
97	004030	125252	.WORD	125252	
98	004032	125252	.WORD	125252	
99	004034	052525	.WORD	052525	
100	004036	052525	.WORD	052525	
101	004040	125252	.WORD	125252	
102	004042	125252	.WORD	125252	

103	004044	052525	.WORD	052525	
104	004046	125252	.WORD	125252	
105	004050	052525	.WORD	052525	
106	004052	125252	.WORD	125252	
107	004054	052525	.WORD	052525	
108	004056	125252	.WORD	125252	
109					
110	004060	007417	PAT6: .WORD	007417	;PATTERN 6
111	004062	007417	.WORD	007417	
112	004064	007417	.WORD	007417	
113	004066	170360	.WORD	170360	
114	004070	170360	.WORD	170360	
115	004072	170360	.WORD	170360	
116	004074	007417	.WORD	007417	
117	004076	007417	.WORD	007417	
118	004100	170360	.WORD	170360	
119	004102	170360	.WORD	170360	
120	004104	007417	.WORD	007417	
121	004106	170360	.WORD	170360	
122	004110	007417	.WORD	007417	
123	004112	170360	.WORD	170360	
124	004114	007417	.WORD	007417	
125	004116	170360	.WORD	170360	
126					
127	004120	026455	PAT7: .WORD	026455	;PATTERN 7
128	004122	026455	.WORD	026455	
129	004124	026455	.WORD	026455	
130	004126	151322	.WORD	151322	
131	004130	151322	.WORD	151322	
132	004132	151322	.WORD	151322	
133	004134	026455	.WORD	026455	
134	004136	026455	.WORD	026455	
135	004140	151322	.WORD	151322	
136	004142	151322	.WORD	151322	
137	004144	026455	.WORD	026455	
138	004146	151322	.WORD	151322	
139	004150	026455	.WORD	026455	
140	004152	151322	.WORD	151322	
141	004154	026455	.WORD	026455	
142	004156	151322	.WORD	151322	
143					
144	004160	165555	PAT8: .WORD	165555	;PATTERN 8
145	004162	133333	.WORD	133333	
146	004164	165555	.WORD	165555	
147	004166	133333	.WORD	133333	
148	004170	165555	.WORD	165555	
149	004172	133333	.WORD	133333	
150	004174	165555	.WORD	165555	
151	004176	133333	.WORD	133333	
152	004200	165555	.WORD	165555	
153	004202	133333	.WORD	133333	
154	004204	165555	.WORD	165555	
155	004206	133333	.WORD	133333	
156	004210	165555	.WORD	165555	
157	004212	133333	.WORD	133333	
158	004214	165555	.WORD	165555	
159	004216	133333	.WORD	133333	

160					
161	004220	000001	PAT9:	.WORD	000001 ;PATTERN 9
162	004222	000002		.WORD	000002
163	004224	000004		.WORD	000004
164	004226	000010		.WORD	000010
165	004230	000020		.WORD	000020
166	004232	000040		.WORD	000040
167	004234	000100		.WORD	000100
168	004236	000200		.WORD	000200
169	004240	000400		.WORD	000400
170	004242	001000		.WORD	001000
171	004244	002000		.WORD	002000
172	004246	004000		.WORD	004000
173	004250	010000		.WORD	010000
174	004252	020000		.WORD	020000
175	004254	040000		.WORD	040000
176	004256	100000		.WORD	100000
177					
178	004260	177776	PAT10:	.WORD	177776 ;PATTERN 10
179	004262	177775		.WORD	177775
180	004264	177773		.WORD	177773
181	004266	177767		.WORD	177767
182	004270	177757		.WORD	177757
183	004272	177737		.WORD	177737
184	004274	177677		.WORD	177677
185	004276	177577		.WORD	177577
186	004300	177377		.WORD	177377
187	004302	176777		.WORD	176777
188	004304	175777		.WORD	175777
189	004306	173777		.WORD	173777
190	004310	167777		.WORD	167777
191	004312	157777		.WORD	157777
192	004314	137777		.WORD	137777
193	004316	077777		.WORD	077777
194					
195	004320	172666	PAT11:	.WORD	172666 ;PATTERN 11
196	004322	155555		.WORD	155555
197	004324	172666		.WORD	172666
198	004326	155555		.WORD	155555
199	004330	172666		.WORD	172666
200	004332	155555		.WORD	155555
201	004334	172666		.WORD	172666
202	004336	155555		.WORD	155555
203	004340	172666		.WORD	172666
204	004342	155555		.WORD	155555
205	004344	172666		.WORD	172666
206	004346	155555		.WORD	155555
207	004350	172666		.WORD	172666
208	004352	155555		.WORD	155555
209	004354	172666		.WORD	172666
210	004356	155555		.WORD	155555
211					
212	004360	077777	PAT12:	.WORD	077777 ;PATTERN 12
213	004362	137777		.WORD	137777
214	004364	157777		.WORD	157777
215	004366	167777		.WORD	167777
216	004370	173777		.WORD	173777

217	004372	175777	.WORD	175777	
218	004374	176777	.WORD	176777	
219	004376	177377	.WORD	177377	
220	004400	177577	.WORD	177577	
221	004402	177677	.WORD	177677	
222	004404	177737	.WORD	177737	
223	004406	177757	.WORD	177757	
224	004410	177767	.WORD	177767	
225	004412	177773	.WORD	177773	
226	004414	177775	.WORD	177775	
227	004416	177776	.WORD	177776	
228					
229	004420	153333	PAT13: .WORD	153333	;PATTERN 13
230	004422	066667	.WORD	066667	
231	004424	153333	.WORD	153333	
232	004426	066667	.WORD	066667	
233	004430	153333	.WORD	153333	
234	004432	066667	.WORD	066667	
235	004434	153333	.WORD	153333	
236	004436	066667	.WORD	066667	
237	004440	153333	.WORD	153333	
238	004442	066667	.WORD	066667	
239	004444	153333	.WORD	153333	
240	004446	066667	.WORD	066667	
241	004450	153333	.WORD	153333	
242	004452	066667	.WORD	066667	
243	004454	153333	.WORD	153333	
244	004456	066667	.WORD	066667	
245					
246	004460	000000	PAT14: .WORD	000000	;PATTERN 14
247	004462	177777	.WORD	177777	
248	004464	177777	.WORD	177777	
249	004466	177777	.WORD	177777	
250	004470	177777	.WORD	177777	
251	004472	177777	.WORD	177777	
252	004474	177777	.WORD	177777	
253	004476	177777	.WORD	177777	
254	004500	177777	.WORD	177777	
255	004502	177777	.WORD	177777	
256	004504	177777	.WORD	177777	
257	004506	177777	.WORD	177777	
258	004510	177777	.WORD	177777	
259	004512	177777	.WORD	177777	
260	004514	177777	.WORD	177777	
261	004516	177777	.WORD	177777	
262					
263	004520	177777	PAT15: .WORD	177777	;PATTERN 15
264	004522	000000	.WORD	000000	
265	004524	000000	.WORD	000000	
266	004526	000000	.WORD	000000	
267	004530	000000	.WORD	000000	
268	004532	000000	.WORD	000000	
269	004534	000000	.WORD	000000	
270	004536	000000	.WORD	000000	
271	004540	000000	.WORD	000000	
272	004542	000000	.WORD	000000	
273	004544	000000	.WORD	000000	

274	004546	000000	.WORD	000000
275	004550	000000	.WORD	000000
276	004552	000000	.WORD	000000
277	004554	000000	.WORD	000000
278	004556	000000	.WORD	000000
279				
460				

```

1      .SBTTL START OF PROGRAM
2
3 004560 012737 177777 001322 START3: MOV    #-1,BUSADR    ;GET BUSADR FLAG
4 004566 000402                BR      STRT1A
5
6 004570 005037 001322        START1: CLR    BUSADR    ;CLR BUSADR FLAG
7 004574 005037 001320        STRT1A: CLR   CNTRLC    ;NO CONTROL "C"
8 004600 000411                BR      START
9
10 004602 012737 177777 001322 START4: MOV    #-1,BUSADR   ;SET BUSADR FLAG
11 004610 000402                BR      STRT2A
12
13 004612 005037 001322        START2: CLR    BUSADR    ;CLR BUSADR FLAG
14 004616 012737 177777 001320 STRT2A: MOV    #-1,CNTRLC ;SET CONTROL "C" FLAG
15
16 004624 000240                START:  NOP
17 004626 005227 000000        INC     #0          ;:TY LOOP, WAIT FOR INCREMENT
18 004632 001375                BNE    -4          ;:OF WORD
19 004634 000005                RESET   -4          ;:CLEAR THE WORLD
20
21      .SBTTL INITIALIZE THE COMMON TAGS
      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
      MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
      CLR    (R6)+           ;;CLEAR MEMORY LOCATION
      CMP    #SWR,R6         ;;DONE?
      BNE    -6              ;;LOOP BACK IF NO
      MOV    #STACK,SP      ;;SETUP THE STACK POINTER
      ;;INITIALIZE A FEW VECTORS
      MOV    # $SCOPE,@#IOTVEC ;;:IOT VECTOR FOR SCOPE ROUTINE
      MOV    #340,@#IOTVEC+2 ;;:LEVEL 7
      MOV    # $ERROR,@#EMTVEC ;;:EMT VECTOR FOR ERROR ROUTINE
      MOV    #340,@#EMTVEC+2 ;;:LEVEL 7
      MOV    # $TRAP,@#TRAPVEC ;;:TRAP VECTOR FOR TRAP CALLS
      MOV    #340,@#TRAPVEC+2;:LEVEL 7
      MOV    $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
      MOV    #176543,$SHNUM  ;;PRIME THE RANDOM NUMBER GENERATOR
      MOV    #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
      CLR    $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
      CLR    $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
      MOV    #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
      MOV    #.,$LPADR     ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
      MOV    #.,$LPERR     ;;SETUP THE ERROR LOOP ADDRESS
      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
      MOV    @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
      MOV    #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
      MOV    #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
      MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
      CMP    #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
      BNE    66$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
      ;;AND THE HARDWARE SWR IS NOT = -1
      BR    65$           ;;BRANCH IF NO TIMEOUT
      64$: MOV    #65$,(SP) ;;SET UP FOR TRAP RETURN
      65$: MOV    #SWREG,SWR ;;POINT TO SOFTWARE SWR
      66$: MOV    #DISPREG,DISPLAY
      MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
    
```



```

005064 005037 001242          CLR      $PASS          ;;CLEAR PASS COUNT
005070 132737 000200 001255  BITB     #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
005076 001403                BEQ      67$            ;;YES,USE NON-APT SWITCH
005100 012737 001256 001154  MOV      #$$SWREG,$SWR  ;;NO,USE APT SWITCH REGISTER
005106
22 67$:
.SBTTL  TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005106 005227 177777          INC      #-1            ;;FIRST TIME?
005112 001054                BNE     68$            ;;BRANCH IF NO
005114 022737 021244 000042  CMP     #SENDAD,@#42   ;;ACT-11?
005122 001450                BEQ     68$            ;;BRANCH IF YES
005124 104401 005172          TYPE     ,69$          ;;TYPE ASCIZ STRING
.SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
005130 005737 000042          TST     @#42           ;;ARE WE RUNNING UNDER XXDP/ACT?
005134 001012                BNE     70$            ;;BRANCH IF YES
005136 123727 001254 000001  CMPB   $ENV,#1        ;;ARE WE RUNNING UNDER APT?
005144 001406                BEQ     70$            ;;BRANCH IF YES
005146 023727 001154 000176  CMP     $SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
005154 001005                BNE     71$            ;;BRANCH IF NO
005156 104406                GTSWR                    ;;GET SOFT-SWR SETTINGS
005160 000403                BR      71$
005162 112737 000001 001150 70$:  MOVB   #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
005170 005170 000425          71$:  BR      68$
;;69$: .ASCIZ <CRLF>@CZRMVAO - RM05/3/2 EXTENDED DRIVE TEST@<CRLF>
68$:
23 005244
24 005244 012700 001174          MOV     #REGAD,RO      ;FIRST ADDRESS
25 005250 005020          1$:  CLR     (RO)+          ;CLEAR VARIABLE STORAGE
26 005252 022700 001224          CMP     #BELL,RO      ;DONE?
27 005256 001374          BNE     1$            ;NO--BRANCH
28 005260 013737 001516 001164  MOV     TPS,$TPS      ;SETUP THE STATUS AND BUFFER REG'S
29 005266 013737 001520 001166  MOV     TPB,$TPB      ;FOR THE TYPE ROUTINE
30
31 ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
32 ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
33
34 005274 005037 001466          CLR     XXDP          ;CLEAR 'XXDP' LOAD DEVICE STORAGE
35 005300 122737 000016 000041  CMPB   #16,@#41      ;LOADED FROM AN RM05/3/2 ?
36 005306 001160          BNE     4$            ;BR IF NOT
37 005310 013737 000040 001466  MOV     @#40,XXDP     ;GET DEVICE INDICATOR AND NUMBER
38 005316 122737 000007 001466  CMPB   #7,XXDP       ;IS IT A VALID NUMBER ?
39 005324 103002          BHIS   2$            ;YES
40 005326 105037 001466          CLRB   XXDP          ;NO, DEFAULT TO DRIVE 0
41 005332 005737 000042          2$:  TST     @#42         ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
42 005336 001425          BEQ     3$            ;BR IF NEITHER
43 005340 104401 005346          TYPE   ,73$         ;;TYPE ASCIZ STRING
005344 000412          BR     72$          ;;GET OVER THE ASCIZ
;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
72$:
44 005372 005046          CLR     -(SP)         ;CLEAR WORD ON STACK
45 005374 113716 001466          MOVB   XXDP,(SP)     ;GET DRIVE ADDRESS
46 005400 104403          TYPOS                    ;TYPE THE ADDRESS
47 005402      J01          .BYTE  1            ;ONLY 1 CHARACTER
48 005403      000          .BYTE  0            ;SUPRESS LEADING ZEROS
49 005404 104401 001231          TYPE   ,SCRLF        ;CR-LF

```

```

50 005410 000517          BR      4$          ;GET NUMBER OF DRIVES
51
52 005412 005227 177777  3$:      INC      #-1          ;FIRST TIME THRU HERE ?
53 005416 001114          BNE     4$          ;NO
54 005420 104401 005426  TYPE     75$         ;:TYPE ASCIZ STRING
   005424 000410          BR      74$         ;:GET OVER THE ASCIZ
   ;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
   74$:
55 005446 005046          CLR     -(SP)        ;CLEAR WORD ON STACK
56 005450 113716 001466  MOVB   XXDP,(SP)    ;GET DRIVE ADDRESS
57 005454 104403          TYPOS                    ;TYPE DRIVE ADDRESS
58 005456      001        .BYTE   1           ;ONLY 1 CHARACTER
59 005457      000        .BYTE   0           ;SUPRESS LEADING ZEROS
60 005460 104401 005466  TYPE     77$         ;:TYPE ASCIZ STRING
   005464 000431          BR      76$         ;:GET OVER THE ASCIZ
   ;:77$: .ASCIZ /, HALT PROGRAM, REMOVE RRDPAK AND REPLACE IT/<CRLF>
   76$:
61 005550 104401 005556  TYPE     78$         ;:TYPE ASCIZ STRING
   005554 000435          BR      4$          ;:GET OVER THE ASCIZ
   ;:78$: .ASCIZ /WITH A WORK PAK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
   4$:
65 005650 004737 023404  JSR    PC,$TKINT    ;TURN ON THE TTY KEYBOARD INTERRUPT
66 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
   005654 005737 000042  TST    @#42         ;:ARE WE RUNNING UNDER XXDP/ACT?
   005660 001012          BNE     79$         ;:BRANCH IF YES
   005662 123727 001254 000001  CMPB   $ENV,#1      ;:ARE WE RUNNING UNDER APT?
   005670 001406          BEQ    79$         ;:BRANCH IF YES
   005672 023727 001154 000176  CMP    SWR,#SWREG   ;:SOFTWARE SWITCH REG SELECTED?
   005700 001005          BNE     80$         ;:BRANCH IF NO
   005702 104406          GTSWR                    ;:GET SOFT-SWR SETTINGS
   005704 000403          BR      80$
   005706 112737 000001 001150 79$:   MOVB   #1,$AUTOB    ;:SET AUTO-MODE INDICATOR
   005714 80$:
67 005714 005227 177777  INC     #-1          ;SEE IF FIRST START
68 005720 001002          BNE     SRTINT       ;BR IF NOT
69 005722 004737 045260  JSR    PC,GETADR    ;GET OR CHECK THE RH/RM ADDRESS
70
71 005726 104401 001231  SRTINT: TYPE     ,SCLF ;CR-LF
72 005732 004737 026114  JSR    PC,LP.AVL    ;CHECK FOR A LINE PRINTER
73 005736 005037 177776  CLR     PS          ;ENSURE THE PRIORITY = 0
74 005742 012737 000001 001120  MOV    #1,$ICNT     ;SET ITERATION COUNT TO 1
75 005750 004737 033776  JSR    PC,GETSWR    ;GO CHECK FOR CONTROL SWITCHES
76 005754 004737 026156  JSR    PC,ST.CLK    ;INITIALIZE THE CLOCK
77
78 005760 004737 037104  SETVEC: JSR    PC,RMINIT ;CHECK THE DRIVE STATUS
79 005764 012737 177777 037026  MOV    #-1,$SAVEFG ;SET THE SAVE REGISTERS FLAG
80 005772 005227 177777  INC     #-1          ;FIRST TIME THRU HERE ?
81 005776 001403          BEQ    1$           ;BR IF NO
82 006000 005737 001320  TST    CNTRLC       ;CONTROL 'C' SWITCH SET ?
83 006004 001112          BNE     SRTDRV       ;CONTINUE IF YES
84 006006 012737 000340 177776 1$:   MOV    #PR7,PS      ;SET PRIORITY TO 7
85 006014 005004          CLR     R4          ;DRIVE TABLE POINTER
86 006016 104401 045731  TYPE     ,UNSTAT     ;TYPE 'UNIT STATUS'
87 006022 104401 001231 2$:   TYPE     ,SCLF      ;CR-LF
88 006026 010446          MOV    R4,-(SP)     ;SAVE R4 FOR TYPEOUT
   ;:TYPE DRIVE NUMBER
   ;:GO TYPE--OCTAL ASCII
   006030 104403          TYPOS
  
```

	006032	002		.BYTE	2		::TYPE 2 DIGIT(S)
	006033	000		.BYTE	0		::SUPPRESS LEADING ZEROS
89	006034	104401	046655	TYPE	,BLNKS4		:TYPE 4 SPACES
90	006040	105764	036740	TSTB	DRVSTA(R4)		:CHECK DRIVE'S STATUS
91	006044	100416		BMI	5\$:BR IF UNSAFE
92	006046	001020		BNE	6\$:BR IF ONLINE
93	006050	105764	036750	TSTB	DRVTYP(R4)		:SEE IF OFFLINE OR NONEXISTENT
94	006054	001404		BEQ	3\$:BR IF NONEXISTENT
95	006056	100006		BPL	4\$:BR IF OFFLINE
96	006060	104401	046015	TYPE	,NOTRM		:DRIVE NOT AN RM05/3/2
97	006064	000452		BR	11\$:CHECK NEXT DRIVE
98							
99	006066	104401	045770	3\$: TYPE	,NOTPRS		:DRIVE NOT PRESENT
100	006072	000447		BR	11\$:CHECK NEXT DRIVE
101							
102	006074	104401	045747	4\$: TYPE	,UNTOFF		:DRIVE OFFLINE
103	006100	000416		BR	8\$:PRINT DRIVE TYPE
104							
105	006102	104401	046005	5\$: TYPE	,NOTSAF		:DRIVE UNSAFE
106	006106	000413		BR	8\$:PRINT DRIVE TYPE
107							
108	006110	005737	001466	6\$: TST	XXDP		:LOADED FROM THIS DEVICE ?
109	006114	001406		BEQ	7\$:BR IF NO
110	006116	123704	001466	CMPB	XXDP,R4		:LOADED FROM THIS DRIVE ?
111	006122	001003		BNE	7\$:BR IF NO
112	006124	104401	046036	TYPE	,LODEV		:DRIVE IS LOAD DEVICE
113	006130	000430		BR	11\$		
114	006132	104401	045760	7\$: TYPE	,UNTON		:DRIVE ONLINE
115	006136	104401	046657	8\$: TYPE	,BLNKS2		:TYPE 2 SPACES
116	006142	005000		CLR	RO		
117	006144	116400	036750	MOV	DRVTYP(R4),RO		:GET DRIVE TYPE
118	006150	012737	046063	MOV	#\$RM03,10\$	006210	:ASSUME ADDRESS OF RM03 MESSAGE
119	006156	122700	000004	CMPB	#4,RO		:IS DEVICE AN RM03 ?
120	006162	001411		BEQ	9\$:TYPE IT IF YES
121	006164	012737	046056	MOV	#\$RM02,10\$	006210	:ADDRESS OF RM02 MESSAGE
122	006172	122700	000005	CMPB	#5,RO		:IS DEVICE AN RM02 ?
123	006176	001403		BEQ	9\$:BR IF YES
124	006200	012737	046070	MOV	#\$RM05,10\$	006210	:ADDRESS OF RM05 MESSAGE
125							
126	006206	104401		9\$: TYPE			:TYPE THE DRIVE TYPE MESSAGE
127	006210	000000		10\$: .WORD	0		:MESSAGE ADDRESS HERE
128							
129	006212	005204		11\$: INC	R4		:INCREMENT DRIVE NUMBER/TABLE POINTER
130	006214	020427	000010	CMP	R4,#8.		:FINISHED ?
131	006220	001300		BNE	2\$:BR IF NOT
132	006222	104401	001231	TYPE	,\$CRLF		:CR-LF
133							
134	006226	005037	177776	CLR	PS		:SET PRIORITY BACK TO '0'
135							
136	006232	005737	001320	SRTDRV: TST	CNTRLC		:CONTROL 'C' START/RESTART?
137	006236	001417		BEQ	2\$:NO--BRANCH
138	006240	013746	001316	MOV	SAVCSW,-(SP)		:GET THE PREVIOUS 'C.SWR' CONTENTS
139	006244	063716	001314	ADD	C.SWR,(SP)		:SET UP TO SEE IF 'BIT00' IS DIFFERENT
140	006250	032726	000001	BIT	#BIT00,(SP)+		:IS 'BIT00' DIFFERENT ?
141	006254	001405		BEQ	1\$:BR IF NOT
142	006256	013737	001314	MOV	C.SWR,SAVCSW	001316	:STORE PRESENT 'C.SWR' VALUE
143	006264	004737	026434	JSR	PC,LODFLT		:RESET PARAMETERS TO THEIR DEFAULT VALUES

```

144 006270 004737 034226 1$: JSR PC,GT.PRM ;GET PARAMETERS
145 006274 000426 BR 6$
146 006276 004737 026434 2$: JSR PC,LODFLT ;SETUP DEFAULT PARAMETERS
147 006302 005037 001326 CLR DRVSEL ;NO DRIVES SELECTED
148 006306 005000 CLR R0 ;DETERMINE THE DRIVES THAT
149 006310 012701 000001 MOV #1,R1 ;ARE AVAILABLE FOR TESTING
150
151 006314 105760 036740 3$: TSTB DRVSTA(R0) ;IS DRIVE ON-LINE ?
152 006320 003411 BLE 5$ ;BR IF NO
153 006322 005737 001466 TST XXDP ;LOADED FROM THIS DEVICE ?
154 006326 001403 BEQ 4$ ;BR IF NO
155 006330 123700 001466 CMPB XXDP,R0 ;LOADED FROM THIS DRIVE ?
156 006334 001403 BEQ 5$ ;BR IF YES
157 006336 156037 037054 001326 4$: BISB ATABIT(R0),DRVSEL ;YES, SELECT DRIVE FOR TESTING
158 006344 005200 5$: INC R0 ;TRY NEXT DRIVE
159 006346 106301 ASLB R1 ;ANY MORE DRIVES TO CHECK ?
160 006350 001361 BNE 3$ ;BR IF YES
161
162 006352 005037 037030 6$: CLR SEEKFG ;CLEAR SEEK FLAG
163 006356 032737 000400 001314 BIT #SW08,C.SWR ;DO SEEK BEFORE DATA TRANSFER?
164 006364 001002 BNE 7$ ;YES--BRANCH
165 006366 005137 037030 COM SEEKFG ;NO
166 006372 7$:
167 006372 104401 046075 TYPE ,DRIVES ;'DRIVES(S) TO BE TESTED'
168 006376 005037 021200 CLR $ENDCT ;DETERMINE PASSES TO MAKE AND
169 006402 005000 CLR R0 ;THE DRIVES TO BE TESTED
170 006404 013701 001326 MOV DRVSEL,R1 ;ANY DRIVES SELECTED?
171 006410 001004 BNE 8$ ;YES--BRANCH
172 006412 104401 046125 TYPE ,NONE ;'NONE'
173 006416 000137 021020 JMP $EOP ;JUMP TO END OF PROGRAM
174
175 006422 006201 8$: ASR R1 ;REPORT THE DRIVES TO BE TESTED
176 006424 103011 BCC 9$
177 006426 005237 021200 INC $ENDCT ;GIVE THIS DRIVE A PASS
178 006432 010046 MOV RO,-(SP) ;SAVE RO FOR TYPEOUT
006434 104403 TYPOS ;GO TYPE--OCTAL ASCII
006436 001 .BYTE 1 ;TYPE 1 DIGIT(S)
006437 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
179 006440 005701 TST R1 ;MORE DRIVES?
180 006442 001404 BEQ 10$ ;NO--BRANCH
181 006444 104401 046132 TYPE ,COMMA ;','
182 006450 005200 9$: INC R0 ;FORM DRIVE NUMBER
183 006452 000763 BR 8$
184
185 006454 013737 021200 021172 10$: MOV $ENDCT,$EOPCT
186 006462 005737 001340 TST CLKSTA ;IS KW11-P AVAILABLE ?
187 006466 003006 BGT RSTR1 ;BR IF YES
188 006470 032737 036000 001330 BIT #36000,TSTNMS ;ANY TIMING TESTS TO BE PERFORMED ?
189 006476 001402 BEQ RSTR1 ;BR IF NO
190 006500 104401 046135 TYPE ,NOCLOCK ;TYPE NO KW11-P CLOCK MESSAGE
191
192 006504 005737 001326 RSTR1: TST DRVSEL ;ANY DRIVES SELECTED ?
193 006510 001002 BNE 1$ ;BR IF YES
194 006512 000137 004612 JMP START2 ;GET DRIVE SELECTION ENTRY
195 ;RESTART AT BEGINNING
196 006516 005037 001350 1$: CLR CHKDRV ;INIT. THE CHECK DRIVE KEY
197 006522 012737 000001 001352 MOV #1,DRVMSK ;START TO CHECK DESIRED DRIVES
  
```

```

198
199 006530 033737 001352 001326 RSTR2: BIT      DRVMSK,DRVSEL  ;IS THIS DRIVE SELECTED?
200 006536 001010                BNE      DRVOK      ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
201
202 006540 012706 001100                RESTART: MOV     #STACK,SP  ;SETUP THE STACK POINTER
203 006544 005237 001350                INC      CHKDRV      ;MOVE TO NEXT DRIVE NUMBER
204 006550 106337 001352                ASLB    DRVMSK      ;POSITION THE MASK
205 006554 103753                BCS     RSTR1       ;BRANCH IF THE DRIVE NUMBER NEEDS INITIALIZED
206 006556 000764                BR      RSTR2
207
208 006560 013702 001350                DRVOK:  MOV     CHKDRV,R2   ;PICKUP THE DRIVE NUMBER
209 006564 105762 036740                TSTB   DRVSTA(R2)     ;IS DESIRED DRIVE ON-LINE?
210 006570 003005                BGT    1$           ;YES, BRANCH
211 006572 104011                EMT    11
212 006574 043737 001352 001326                BIC    DRVMSK,DRVSEL  ;CLEAR DRIVE'S SELECTION BIT
213 006602 000756                BR     RESTART      ;RETURN
214 006604 010237 045456                1$:   MOV     R2,DPB.A   ;SET THE DRIVE NUMBER INTO THE DPB'S
215 006610 010237 045476                MOV     R2,DPB.B
216 006614 010237 045516                MOV     R2,DPB.C
217 006620 010237 045536                MOV     R2,DTADPB
218 006624 004737 027066                JSR    PC,LDCMD      ;LOAD COMMAND INTO DPB.B AND DPB.C
219 006630 012737 021020 001346                MOV     #SEOP,CYPASS  ;IF ERROR GO TO END OF PROGRAM
220 006636 112737 000020 045457                MOVB   #20,DPB.A+1   ;ASSUME 16 BIT FORMAT
221 006644 032737 000001 001314                BIT    #BIT00,C.SWR  ;16 BIT FORMAT REQUESTED ?
222 006652 001402                BEQ    2$           ;BR IF YES
223 006654 105037 045457                CLRB   DPB.A+1      ;CLEAR THE 'FMT16' BIT
224 006660 112737 000143 045460                2$:   MOVB   #SETFORM,DPB.A+2 ;SET THE FORMAT BIT PER DPB.A+1
225 006666 004037 027132                JSR    RO,CALL.A     ;GO EXECUTE THE COMMAND
226 006672 004737 026072                JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
227 006676 112737 000107 045460                MOVB   #RECAL,DPB.A+2 ;RECAL=COMMAND
228 006704 004037 027132                JSR    RO,CALL.A     ;GO EXECUTE THE COMMAND
229 006710 104401 046222                TYPE   ,TSTDRV      ;'TESTING DRIVE '
230 006714 010246                MOV     R2,-(SP)     ;;SAVE R2 FOR TYPEOUT
    006716 104403                TYPOS  ;GO TYPE--OCTAL ASCII
    006720 002                .BYTE 2             ;;TYPE 2 DIGIT(S)
    006721 000                .BYTE 0             ;;SUPPRESS LEADING ZEROS
231 006722 104401 046132                TYPE   ,COMMA       ;TYPE ','
232 006726 104401 046242                TYPE   ,SERIAL      ;TYPE 'MBA SN# '
233 006732 012700 000004                MOV     #4,RO        ;FOUR DIGITS TO TYPE
234 006736 013701 045606                MOV     RM,REG+30,R1 ;SERIAL NUMBER
235 006742 005002                3$:   CLR     R2          ;ZERO
236 006744 006101                ROL    R1            ;PUT THE NEXT DIGIT
237 006746 006102                ROL    R2            ;INTO R2
238 006750 006101                ROL    R1
239 006752 006102                ROL    R2
240 006754 006101                ROL    R1
241 006756 006102                ROL    R2
242 006760 006101                ROL    R1
243 006762 006102                ROL    R2
244 006764 062702 000060                ADD    #'0,R2        ;MAKE IT ASCII
245 006770 010227                MOV     R2,(PC)+     ;SAVE IT
246 006772 000000                4$:   .WORD 0
247 006774 104401 006772                TYPE   ,4$          ;TYPE
248 007000 005300                DEC    RO            ;ALL DIGITS TYPED?
249 007002 003557                BGT    3$           ;NO -- BRANCH
250 007004 104401 001231                TYPE   ,$CRLF       ;CR-LF
251 007010 113737 001462 001131                MOVB   ERR.CT,$ERMAX ;SETUP MAX ERROR COUNT
  
```


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            822         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            4 OR 18    LAST TRACK ADDRESS
:*IT             1          INCREMENT VALUE
:*NT            FT+IT      NEW OR MODIFIED TRACK ADDRESS
*
:*FS             0          FIRST SECTOR ADDRESS
:*LS            31         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/SEEK TEST
:*THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE
:*COMMAND CYCLE AND THEN SEEK FORWARD TO CYLINDER 'LC' AT
:*THE COMPLETION OF BOTH COMMANDS STATUS INDICATIONS ARE
:*CHECKED TO ENSURE NO ERRORS OCCURRED.
:*****
TST0:
    
```

```

007016
007016 000240
007020 033737 001526 001330
007026 001002
007030 000137 007204

007034 012737 000000 001116
007042 004737 026672
007046 012737 007166 001124
007054 013737 002324 001220
007062 112737 000031 001131
007070 012737 000000 001240
    
```

```

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

MOV #0,$TSTNM ;SET TEST #0 AND CLEAR ($SERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TEST0,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #0,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
    
```

45

```

007076 032777 010000 172050 BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
007104 001406 BEQ .+16 ;BR IF YES
007106 104401 046253 TYPE ,MSGTST ;TYPE 'TEST'
007112 013746 001240 MOV $TESTN,-(SP) ;;SAVE $TESTN FOR TYPEOUT
007116 104403 TYPOS ;;GO TYPE--OCTAL ASCII
007120 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
007121 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS

46 007122 112737 000107 045460 MOVB #RECAL,DPB A+2 ;RECAL=COMMAND
47 007130 113737 002342 045506 MOVB FS,DPB.B+10 ;FS
48 007136 113737 002334 045507 MOVB FT,DPB.B+11 ;FT
49 007144 013737 002330 045510 MOV LC,DPB.B+12 ;LC
58 007152 012737 007202 001346 MOV #EXIT0,BYPASS ;GO TO EXIT0 ON ERROR
007160 012737 007166 001122 MOV #TEST0,$LPADR ;SETUP LOOP ADDRESS
007166 TEST0:
007166 012706 001100 MOV #STACK,SP ;SET UP STACK POINTER
007172 004037 027132 JSR RO,CALL.A ;GO EXECUTE THE COMMAND
007176 004037 027264 JSR RO,CALL.B ;GO EXECUTE THE COMMAND
007202 000004 EXIT0: SCOPE ;CALL SCOPE ROUTINE

59
66
67
*****
;*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.
*****
TST1:
007204 NOP
007204 000240 BIT BITS+<1*2>,TSTNMS ;DO THIS TEST?
007206 033737 001530 001330 BNE .+6 ;BR IF YES
007214 001002 JMP TST2 ;NO--JUMP TO TEST2
007216 000137 007412

007222 012737 000001 001116 MOV #1,$TSTNM ;SET TEST #1 AND CLEAR ($ERFLG)
007230 004737 026672 JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
007234 012737 007374 001124 MOV #TEST1,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
007242 013737 002324 001220 MOV RPT,$TIMES ;GET THE ITERATION COUNT
007250 112737 000031 001131 MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
007256 012737 000001 001240 MOV #1,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

68
007264 032777 010000 171662 BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
007272 001406 BEQ .+16 ;BR IF YES
007274 104401 046253 TYPE ,MSGTST ;TYPE 'TEST'
007300 013746 001240 MOV $TESTN,-(SP) ;;SAVE $TESTN FOR TYPEOUT
007304 104403 TYPOS ;;GO TYPE--OCTAL ASCII
007306 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
007307 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS

69 007310 005037 001464 CLR BASFLG ;CLEAR BAD SECTOR ENCOUNTER FOR THE DRIVE
70 007314 113737 002342 045506 MOVB FS,DPB.B+10 ;FS
71 007322 113737 002344 045526 MOVB LS,DPB.C+10 ;LS
72 007330 113737 002334 045507 MOVB FT,DPB.B+11 ;FT
73 007336 113737 002336 045527 MOVB LT,DPB.C+11 ;LT
74 007344 013737 002326 045510 MOV FC,DPB.B+12 ;FC
75 007352 013737 002330 045530 MOV LC,DPB.C+12 ;LC
80 007360 012737 007410 001346 MOV #EXIT1,BYPASS ;GO TO EXIT1 ON ERROR
    
```



```

007366 012737 007374 001122      MOV      #TEST1,$LPADR      ;SETUP LOOP ADDRESS
007374                                TEST1:
81 007374 012706 001100      MOV      #STACK,SP        ;SET THE STACK POINTER
82 007400 004037 027466      JSR      RO,CALL.C        ;GO EXECUTE THE COMMAND
83 007400 004037 027264      JSR      RO,CALL.B        ;GO EXECUTE THE COMMAND
84 007410 000004                                EXIT1: SCOPE                ;CALL SCOPE ROUTINE
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
007412                                ;*****
007412                                ;*TEST 2      INCREMENT/SEEK TEST
007414 000240                                ;*THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
007414 033737 001532 001330      ;*CYLINDER ADDRESS FROM 'FC' TO 'LC' BY THE INCREMENT 'IC'.
007422 001002                                ;*WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS
007424 000137 007654                                ;*'LC' REVERSE SEEK CYCLES ARE INITIATED; STARTING
                                ;*AT THE LAST LEGAL 'NC' AND DECREMENTING BY 'IC'
                                ;*UNTIL 'NC' IS LESS THAN 'FC'. AT THE COMPLETION OF EACH
                                ;*SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
                                ;*ENSURE PROPER OPERATION.
                                ;*****
007412                                TST2:
007412                                NOP
007414 033737 001532 001330      BIT      BITS+<2*2>,$TSTNMS ;DO THIS TEST?
007422 001002                                BNE     .+6                ;BR IF YES
007424 000137 007654                                JMP     TST3                ;NO--JUMP TO TEST3

007430 012737 000002 001116      MOV      #2,$TSTNM        ;SET TEST #2 AND CLEAR ($ERFLG)
007436 004737 026672                                JSR      PC,LODPRM        ;LOAD THE PARMETERS FOR THE TEST
007442 012737 007546 001124      MOV      #TEST2,$LPERR    ;SETUP THE LOOP ON ERROR ADDRESS
007450 013737 002324 001220      MOV      RPT,$TIMES       ;GET THE ITERATION COUNT
007456 112737 000031 001131      MOV      #25,$ERMAX       ;MAX ERRORS ALLOWED FOR TEST
007464 012737 000002 001240      MOV      #2,$TESTN        ;SET TEST NUMBER IN APT MAIL BOX

007472 032777 010000 171454      BIT      #SW12,$SWR        ;INHIBIT TYPING TEST NUMBER ?
007500 001406                                BEQ     .+16                ;BR IF YES
007502 104401 046253                                TYPE   ,MSGTST            ;TYPE 'TEST'
007506 013746 001240                                MOV     $TESTN,-(SP)       ;SAVE $TESTN FOR TYPEOUT
007512 104403                                TYPOS  ;GO TYPE--OCTAL ASCII
007514 002                                .BYTE  2                   ;TYPE 2 DIGIT(S)
007515 000                                .BYTE  0                   ;SUPPRESS LEADING ZEROS

007516 012737 007524 001122      MOV      #18,$LPADR       ;SETUP LOOP ADDRESS
007524 113737 002342 045506      MOV      FS,DPB.B+10      ;FS
007532 113737 002334 045507      MOV      FT,DPB.B+11      ;FT
007540 012737 007652 001346      MOV      #EXIT2,BYPASS    ;GO TO EXIT2 ON ERROR

007546                                TEST2:
007546 013737 002326 045510      MOV      FC,DPB.B+12      ;FC
007554 012737 007554 001124      MOV      #,$LPERR         ;SETUP THE ERROR LOOP ADDRESS
007562 012706 001100      MOV      #STACK,SP        ;LOAD THE STACK POINTER

007566                                INCSK:
007566 004037 027264                                JSR      RO,CALL.B        ;GO EXECUTE THE COMMAND
007572 063737 002332 045510      ADD     IC,DPB.B+12       ;MOVE TO NEXT CYLINDER
007600 023737 002330 045510      CMP     LC,DPB.B+12       ;OUT OF CYLINDERS?
007606 002367                                BGE     INCSK             ;NO--BRANCH
007610 013737 002330 045510      MOV     LC,DPB.B+12
007616 012737 007616 001124      MOV     #,$LPERR         ;SETUP THE ERROR LOOP ADDRESS
007624 012706 001100      MOV     #STACK,SP        ;LOAD THE STACK POINTER

007630                                DECSK:

```

```

116 007630 004037 027264      JSR    RO,CALL.B      ;GO EXECUTE THE COMMAND
117 007634 163737 002332 045510  SUB    IC,DPB.B+12
118 007642 023737 002326 045510  CMP    FC,DPB.B+12
119 007650 003767      BLE    DECSK
120 007652 000004      EXIT2: SCOPE        ;CALL SCOPE ROUTINE
121
128
129
  
```

```

:*****
:*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.
:*****
  
```

```

007654
007654 000240
007656 033737 001534 001330      NOP
007664 001002      BIT    BITS+<3*2>,TSTNMS ;DO THIS TEST?
007666 000137 010074      BNE    .+6             ;BR IF YES
                                JMP    TST4             ;NO--JUMP TO TEST4

007672 012737 000003 001116      MOV    #3,$TSTNM      ;SET TEST #3 AND CLEAR ($ERFLG)
007700 004737 026672      JSR    PC,LODPRM     ;LOAD THE PARMETERS FOR THE TEST
007704 012737 010010 001124      MOV    #TEST3,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
007712 013737 002324 001220      MOV    RPT,$TIMES    ;GET THE ITERATION COUNT
007720 112737 000031 001131      MOV    #25,$SERMAX   ;MAX ERRORS ALLOWED FOR TEST
007726 012737 000003 001240      MOV    #3,$TESTN     ;:SET TEST NUMBER IN APT MAIL BOX

130 007734 032777 010000 171212      BIT    #SW12,@SWF    ;INHIBIT TYPING TEST NUMBER ?
007742 001406      BEQ    .+16          ;BR IF YES
007744 104401 046253      TYPE   ,MSGTST      ;TYPE 'TEST'
007750 013746 001240      MOV    $TESTN,-(SP)  ;:SAVE $TESTN FOR TYPEOUT
007754 104403      TYPOS ;:GO TYPE--OCTAL ASCII
007756      .BYTE 2           ;:TYPE 2 DIGIT(S)
007757      .BYTE 0           ;:SUPPRESS LEADING ZEROS

131 007760 012737 007766 001122      MOV    #1$, $LPADR   ;SETUP TEST LOOP ADDRESS
132 007766 113737 002342 045506 1$:  MOV    FS,DPB.B+10   ;FS
133 007774 113737 002334 045507      MOV    FT,DPB.B+11   ;FT
137 010002 012737 010072 001346      MOV    #EXIT3,BYPASS ;GO TO BYPASS ON ERROR

TEST3:
138 010010 013737 002326 045510      MOV    FC,DPB.B+12   ;FC
139 010016 012737 010016 001124      MOV    #,$LPERR     ;SETUP THE ERROR LOOP ADDRESS
                                MOV    #STACK,SP      ;LOAD THE STACK POINTER
140 010030 004037 027264      JSR    RO,CALL.B     ;GO EXECUTE THE COMMAND
141 010034 013701 002332      MOV    IC,R1         ;CYLINDER 1
142 010040 012737 010040 001124      MOV    #,$LPERR     ;SETUP THE ERROR LOOP ADDRESS
                                MOV    #S,K,SP       ;LOAD THE STACK POINTER
143 010052 010137 045510 1$:  MOV    R1,DPB.B+12   ;DESIRED CYLINDER
144 010056 004037 027264      JSR    RO,CALL.B     ;GO EXECUTE THE COMMAND
145 010062 006301      ASL    R1            ;MOVE TO NEXT CYLINDER
146 010064 020137 002330      CMP    R1,LC        ;DONE?
147 010070 003770      BLE    1$           ;NO--LOOP
148 010072 000004      EXIT3: SCOPE        ;CALL SCOPE ROUTINE
149
157
158
  
```

```

:*****
:*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:

010074	010074	000240			NOP		
010076	033737	001536	001330		BIT	BITS+<4*2>,TSTNMS	;DO THIS TEST?
010104	001002				BNE	.+6	;BR IF YES
010106	000137	010506			JMP	TST5	;NO--JUMP TO TEST5
010112	012737	000004	001116		MOV	#4,\$TSTNM	;SET TEST #4 AND CLEAR (\$ERFLG)
010120	004737	026672			JSR	PC,LODPRM	;LOAD THE PARMETERS FOR THE TEST
010124	012737	010244	001124		MOV	#TEST4,\$LPERR	;SETUP THE LOOP ON ERROR ADDRESS
010132	013737	002324	001220		MOV	RPT,\$TIMES	;GET THE ITERATION COUNT
010140	112737	000031	001131		MOVB	#25,\$ERMAX	;MAX ERRORS ALLOWED FOR TEST
159 010146	012737	000004	001240		MOV	#4,\$TESTN	;SET TEST NUMBER IN APT MAIL BOX
010154	032777	010000	170772		BIT	#SW12,@SWR	;INHIBIT TYPING TEST NUMBER ?
010162	001406				BEQ	.+16	;BR IF YES
010164	104401	046253			TYPE	,MSG,ST	;TYPE 'TEST'
010170	013746	001240			MOV	\$TESTN,-(SP)	;SAVE \$TESTN FOR TYPEOUT
010174	104403				TYPOS		;GO TYPE--OCTAL ASCII
010176	002				.BYTE	2	;TYPE 2 DIGIT(S)
010177	000				.BYTE	0	;SUPPRESS LEADING ZEROS
160 010200	012737	010206	001122		MOV	#1\$, \$LPADR	;SETUP LOOP ADDRESS
161 010206	113737	002342	045506	1\$:	MOVB	FS,DPB.B+10	;FS
162 010214	113737	002334	045507		MOVB	FT,DPB.B+11	;FT
170 010222	012737	010504	001346		MOV	#EXIT4,BYPASS	;GO TO EXIT4 ON ERROR
010230	005002				CLR	R2	;CLEAR STALL SWITCH (NO STALL)
010232	032737	010000	001314		BIT	#SW12,C.SWR	;STALL REQUIRED?
010240	001401				BEQ	TEST4	;NO--BRANCH
010242	005102				COM	R2	;YES--SET SWITCH
010244					TEST4:		
171 010244	013701	002326			MOV	FC,R1	;SET NC TO FC
172 010250	005037	001434			CLR	STALLO	;START AT ZERO IF STALLS REQUIRED
173 010254	012737	010254	001124		MOV	#, \$LPERR	;SETUP THE ERROR LOOP ADDRESS
010262	012706	001100			MOV	#STACK,SP	;LOAD THE STACK POINTER
174 010266	010137	045510		1\$:	MOV	R1,DPB.B+12	;NC
175 010272	004037	027264			JSR	RO,CALL.B	;GO EXECUTE THE COMMAND
176 010276	005702				TST	R2	;STALL?
177 010300	001403				BEQ	2\$;NO--BRANCH
178 010302	004037	030664			JSR	RO,STALL	;YES--GO TO STALL ROUTINE
179 010306	001434				.WORD	STALLO	;TIME POINTER
180 010310	013737	002326	045510	2\$:	MOV	FC,DPB.B+12	;FC
181 010316	004037	027264			JSR	RO,CALL.B	;GO EXECUTE THE COMMAND
182 010322	005702				TST	R2	;STALL?
183 010324	001413				BEQ	3\$;NO--BRANCH
184 010326	004037	030664			JSR	RO,STALL	;YES--GO TO STALL ROUTINE
185 010332	001434				.WORD	STALLO	;TIME POINTER
186 010334	005237	001434			INC	STALLO	;UPDATE THE TIME
187 010340	023737	001460	001434		CMP	MXSTAL,STALLO	;TIME TO BIG?
188 010346	003547				BGT	1\$;NO--BRANCH
189 010350	005037	001434			CLR	STALLO	;YES--START OVER AT ZERO
190 010354	063701	002332		3\$:	ADD	IC,R1	;MOVE TO NEXT CYLINDER

```

191 010360 020137 002330      CMP      R1,LC      ;LAST CYLINDER COMPLETED?
192 010364 003740      BLE      1$         ;NO--BRANCH
193 010366 013701 002330      MOV      LC,R1      ;SET NC TO LC
194 010372 012737 010372 (01124  MOV      #,$LPERR   ;SETUP THE ERROR LOOP ADDRESS
      010400 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
195 010404 010137 045510 4$:  MOV      R1,DPB.B+12 ;NC
196 010410 004037 027264      JSR      RO,CALL.B  ;GO EXECUTE THE COMMAND
197 010414 005702      TST      R2         ;STALL?
198 010416 001403      BEQ      5$         ;NO--BRANCH
199 010420 004037 030664      JSR      RO,STALL   ;YES--GO TO STALL ROUTINE
200 010424 001434      .WORD    STALLO     ;TIME POINTER
201 010426 013737 002330 045510 5$:  MOV      LC,DPB.B+12 ;LC
202 010434 004037 027264      JSR      RO,CALL.B  ;GO EXECUTE THE COMMAND
203 010440 005702      TST      R2         ;STALL?
204 010442 001413      BEQ      6$         ;NO--BRANCH
205 010444 004037 030664      JSR      RO,STALL   ;YES--GO TO STALL ROUTINE
206 010450 001434      .WORD    STALLO     ;TIME POINTER
207 010452 005237 001434      INC      STALLO     ;UPDATE STALL TIME
208 010456 023737 001460 001434      CMP      MXSTAL,STALLO ;TIME TOO BIG?
209 010464 003347      BGT      4$         ;NO--BRANCH
210 010466 005037 001434      CLR      STALLO     ;YES--SET STALL TIME BACK TO ZERO
211 010472 163701 002332 6$:  SUB      LC,R1      ;NEXT CYLINDER
212 010476 020137 002326      CMP      R1,FC      ;DONE?
213 010502 002340      BGE      4$         ;NO--BRANCH
214 010504 000004      EXIT4:  SCOPE      ;CALL SCOPE ROUTINE
215
226
227

```

```

*****
;*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.
*****

```

```

010506
010506 000240
010510 033737 001540 001330      NOP
010516 001002      BIT      BITS+<5*2>,TSTNMS ;DO THIS TEST?
010520 000137 010732      BNE      +6         ;BR IF YES
      JMP      TST6     ;NO--JUMP TO TEST6

010524 012737 000005 001116      MOV      #5,$TSTNM  ;SET TEST #5 AND CLEAR ($ERFLG)
010532 004737 026672      JSR      PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
010536 012737 010642 001124      MOV      #TEST5,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
010544 013737 002324 001220      MOV      RPT,$TIMES ;GET THE ITERATION COUNT
010552 112737 000031 001131      MOV      #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
010560 012737 000005 001240      MOV      #5,$TESTN  ;SET TEST NUMBER IN APT MAIL BOX

228 010566 032777 010000 170360      BIT      #SW12,$SWR  ;INHIBIT TYPING TEST NUMBER ?
010574 001406      BEQ      +16        ;BR IF YES
010576 104401 046253      TYPE    $MSGTST     ;TYPE 'TEST'
010602 013746 001240      MOV      $TESTN,-(SP) ;SAVE $TESTN FOR TYPEOUT
010606 104403      TYPOS   ;GO TYPE--OCTAL ASCII
010610 002      .BYTE 2 ;TYPE 2 DIGIT(S)

```

```

010611      000                .BYTE 0                ;;SUPPRESS LEADING ZEROS
229 010612  012737  010620  001122      MOV      #1$, $LPADR      ;SETUP LOOP ADDRESS
230 010620  113737  002342  045506  1$:   MOVB     FS,DPB.B+10      ;FS
231 010626  113737  002334  045507      MOVB     FT,DPB.B+11      ;FT
235 010634  012737  010730  001346      MOV      #EXIT5,BYPASS    ;GO TO EXIT5 ON ERROR
010642
236 010642  013701  002326      MOV      FC,R1           ;START NC1 AT FC
237 010646  013702  002330      MOV      LC,R2           ;START NC2 AT LC
238 010652  012737  010652  001124      MOV      #., $LPERR      ;SETUP THE ERROR LOOP ADDRESS
010660  012706  001100      MOV      #STACK,SP       ;LOAD THE STACK POINTER
239 010664  010137  045510  1$:   MOV      R1,DPB.B+12      ;NC1
240 010670  004037  027264      JSR      RO,CALL.B        ;GO EXECUTE THE COMMAND
241 010674  010237  045510      MOV      R2,DPB.B+12      ;NC2
242 010700  004037  027264      JSR      RO,CALL.B        ;GO EXECUTE THE COMMAND
243 010704  063701  002332      ADD      IC,R1           ;NEXT NC1
244 010710  163702  002332      SUB      IC,R2           ;NEXT NC2
245 010714  020137  002330      CMP      R1,LC           ;DONE?
248 010720  003003      BGT      EXIT5           ;YES--BRANCH
249 010722  020237  002326      CMP      R2,FC           ;?
250 010726  002356      BGE      1$             ;NO--BRANCH
251 010730  000004      EXIT5:  SCOPE           ;CALL SCOPE ROUTINE
252
261
262

```

```

*****
;*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.
*****

```

```

010732
010732  000240
010734  033737  001542  001330      NOP
010742  001002      BIT      BITS+<6*2>,TSTNMS ;DO THIS TEST?
010744  000137  011222      BNE     .+6             ;BR IF YES
                                JMP      TST7             ;NO--JUMP TO TEST7
010750  012737  000006  001116      MOV      #6,$TSTNM       ;SET TEST #6 AND CLEAR ($ERFLG)
010756  004737  026672      JSR      PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
010762  012737  011066  001124      MOV      #TEST6,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
010770  013737  002324  001220      MOV      RPT,$TIMES      ;GET THE ITERATION COUNT
010776  112737  000031  001131      MOVB     #25,$ERMAX      ;MAX ERRORS ALLOWED FOR TEST
263 011004  012737  000006  001240      MOV      #6,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
011012  032777  010000  170134      BIT      #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
011020  001406      BEQ     .+16            ;BR IF YES
011022  104401  046253      TYPE    ,MSGTST         ;TYPE 'TEST'
011026  013746  001240      MOV      $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
011032  104403      TYPOS   ;GO TYPE--OCTAL ASCII
011034      002      .BYTE 2              ;;TYPE 2 DIGIT(S)
011035      000      .BYTE 0              ;;SUPPRESS LEADING ZEROS
264 011036  012737  011044  001122      MOV      #1$, $LPADR     ;SETUP LOOP ADDRESS
265 011044  113737  002342  045506  1$:   MOVB     FS,DPB.B+10      ;FS
266 011052  113737  002334  045507      MOVB     FT,DPB.B+11      ;FT

```

```

270 011060 012737 011220 001346      MOV      #EXIT6,BYPASS      ;GO TO EXIT6 ON ERROR
                                TEST6:
271 011066 013701 002326      MOV      FC,R1              ;PICKUP 'FC'
272 011072 013702 002330      MOV      LC,R2              ;FORM LAST CYLINDER THAT
273 011076 162702 000005      SUB      #5,R2              ;IS AVAILABLE FOR TESTING
274 011102 012737 011102 001124      MOV      #.,$LPERR         ;SETUP THE ERROR LOOP ADDRESS
                                MOV      #STACK,SP         ;LOAD THE STACK POINTER
275 011114 020102 001100      1$:    CMP      R1,R2         ;LAST CYLINDER
276 011116 003040      BGT      EXIT6             ;YES--BRANCH
277 011120 010137 045510      MOV      R1,DPB.B+12       ;NC
278 011124 004037 027264      JSR      RO,CALL.B         ;GO EXECUTE THE COMMAND
279 011130 062737 000004 045510      ADD      #4,DPB.B+12       ;NC+4
280 011136 004037 027264      JSR      RO,CALL.B         ;GO EXECUTE THE COMMAND
281 011142 162737 000003 045510      SUB      #3,DPB.B+12       ;NC+1
282 011150 004037 027264      JSR      RO,CALL.B         ;GO EXECUTE THE COMMAND
283 011154 062737 000002 045510      ADD      #2,DPB.B+12       ;NC+3
284 011162 004037 027264      JSR      RO,CALL.B         ;GO EXECUTE THE COMMAND
285 011166 162737 000001 045510      SUB      #1,DPB.B+12       ;NC+2
286 011174 004037 027264      JSR      RO,CALL.B         ;GO EXECUTE THE COMMAND
287 011200 062737 000003 045510      ADD      #3,DPB.B+12       ;NC+5
288 011206 004037 027264      JSR      RO,CALL.B         ;GO EXECUTE THE COMMAND
289 011212 063701 002332      ADD      IC,R1
290 011216 000736      BR      1$
291 011220 000004      EXIT6: SCOPE              ;CALL SCOPE ROUTINE
292
293
294
303
304

```

```

:*****
:*TEST 7      RANDOM SEEK TEST
:*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 PARAMTERS 'FT' AND 'LT'.
:*****
TST7:

```

```

011222 000240
011224 033737 001544 001330      NOP
011232 001002      BIT      BITS+<7*2>,TSTNMS ;DO THIS TEST?
011234 000137 011620      BNE      .+6              ;BR IF YES
                                JMP      TST10             ;NO--JUMP TO TEST10

011240 012737 000007 001116      MOV      #7,$TSTNM        ;SET TEST #7 AND CLEAR ($ERFLG)
011246 004737 026672      JSR      PC,LODPRM        ;LOAD THE PARMETERS FOR THE TEST
011252 012737 011362 001124      MOV      #TEST7,$LPERR    ;SETUP THE LOOP ON ERROR ADDRESS
011260 013737 002324 001220      MOV      RPT,$TIMES       ;GET THE ITERATION COUNT
011266 112737 000031 001131      MOV      #25,$ERMAX       ;MAX ERRORS ALLOWED FOR TEST
011274 012737 000007 001240      MOV      #7,$TESTN        ;SET TEST NUMBER IN APT MAIL BOX

305 011302 032777 010000 167644      BIT      #SW12,@SWR       ;INHIBIT TYPING TEST NUMBER ?
011310 001406      BEQ      .+16             ;BR IF YES
011312 104401 046253      TYPE     ,MSGTST          ;TYPE 'TEST'
011316 013746 001240      MOV      $TESTN,-(SP)     ;SAVE $TESTN FOR TYPEOUT
011322 104403      TYPOS    ;GO TYPE--OCTAL ASCII!
011324 002      .BYTE   2               ;TYPE 2 DIGIT(S)
011325 000      .BYTE   0               ;SUPPRESS LEADING ZEROS

306 011326 113737 002334 045507      MOV      FT,DPB.B+11      ;LOAD STARTING TRACK ADDRESS

```

```

307 011334 112737 000105 045460      MOVB  #SEEK,DPB.A+2      ;SEEK=COMMAND
308 011342 013704 037066              MOV   RMADR,R4          ;UNIBUS ADDRESS OF THE RH/RM
313 011346 012737 011616 001346      MOV   #EXIT7,BYPASS    ;ERROR TERMINATION ADDRESS
      011354 012737 011362 001122      MOV   #TEST7,$LPADR    ;SETUP THE LOOP ON TEST ADDRESS
      011362                                TEST7:
314 011362 012706 001100              MOV   #STACK,SP        ;SETUP THE STACK POINTER
315 011366 013737 002326 045510      MOV   FC,DPB.B+12     ;INITIAL CYLINDER ADDRESS
316 011374 023737 002326 002330      LMP   FC,LC           ;CYLINDER LIMITS THE SAME ?
317 011402 001422                      BEQ   1$              ;BR IF THEY ARE
318 011404 004737 025550              JSR   PC,$RAND         ;CYCLE THE RANDOM NUMBER GENERATOR
319 011410 013746 025646              MOV   $MINUM,-(SP)    ;USE THE HIGH RANDOM NUMBER
320 011414 005046                      CLR   -(SP)           ;UPPER DIVIDEND
321 011416 013746 002330              MOV   LC,-(SP)        ;FORM THE DIVISOR
322 011422 005216                      INC   (SP)            ;INCREMENT
323 011424 163716 002326              SUB   FC,(SP)         ;SUBTRACT THE LOWER LIMIT
324 011430 004737 025652              JSR   PC,$DIV         ;DIVIDE
325 011434 062637 045510              ADD   (SP)+,DPB.B+12  ;ADD THE REMAINDER TO THE INITIAL CYLINDER
326 011440 005726                      TST   (SP)+           ;DISCARD THE QUOTIENT
327 011442 013737 045510 045470      MOV   DPB.B+12,DPB.A+12 ;COPY NEW CYLINDER ADDRESS
328 011450                                1$:
      011450 012737 011450 001124      MOV   #,$LPERR        ;SETUP THE ERROR LOOP ADDRESS
      011456 012706 001100              MOV   #STACK,SP      ;LOAD THE STACK POINTER
329 011462 004037 027132              JSR   RO,CALL.A       ;GO EXECUTE THE COMMAND
330 011466 012737 011466 001124      MOV   #,$LPERR        ;SETUP THE ERROR LOOP ADDRESS
      011474 012706 001100              MOV   #STACK,SP      ;LOAD THE STACK POINTER
331 011500 113764 045456 000010      MOVB  DPB.A,RMCS2(R4) ;SELECT THE DRIVE
332 011506 016446 000020              MOV   RMLA(R4),-(SP) ;GET THE LOOK AHEAD REGISTER
333 011512 006316                      ASL   (SP)            ;ALIGN THE SECTOR ADDRESS
334 011514 006316                      ASL   (SP)            ;ALIGN THE SECTOR ADDRESS
335 011516 000316                      SWAB  (SP)            ;PUT ADDRESS IN LOWER BYTE
336 011520 105766 000001              TSTB  1(SP)           ;IN THE 1ST 20% OF SECTOR ?
337 011524 001401                      BEQ   2$              ;BR IF YES
338 011526 105216                      INCB  (SP)            ;INCREMENT THE SECTOR ADDRESS
339 011530 105216                                2$:
      011530 105216                      INCB  (SP)            ;INCREMENT THE SECTOR ADDRESS
340 011532 112637 045546              MOVB  (SP)+,DTADPB+10 ;LOAD THE DPB
341 011536 013746 002456              MOV   PRMLT+24,-(SP) ;PUT LAST SECTOR ADDRESS ON THE STACK
342 011542 005216                      INC   (SP)            ;INCREMENT IT
343 011544 122637 045546              CMPB  (SP)+,DTADPB+10 ;NEW SECTOR ADDRESS TOO LARGE ?
344 011550 103007                      BHIS  4$              ;BR IF NOT
345 011552 103403                      BLO   3$              ;BR IF ADDRESS IS 2 GREATER
346 011554 105037 045546              CLRB  DTADPB+10      ;RESET TO SECTOR ADDRESS 0
347 011560 000403                      BR    4$              ;CONTINUE
348 011562 112737 000001 045546      3$:
349 011570                                4$:
      011570 004037 027264              JSR   RO,CALL.B       ;GO EXECUTE THE COMMAND
350 011574 105237 045507              INCB  DPB.B+11        ;INCREMENT THE TRACK ADDRESS
351 011600 123737 045507 002336      CMPB  DPB.B+11,LT    ;MAXIMUM ?
354 011606 101403                      BLOS  EXIT7          ;BR IF NOT
355 011610 113737 002334 045507      MOVB  FT,DPB.B+11    ;RELOAD STARTING TRACK ADDRESS
356 011616 000004                                EXIT7:
      SCOPE                                ;CALL SCOPE ROUTINE
357
379
380

```

```

:*****
:*TEST 10      SERVO SETTLE DOWN TEST
:*THIS TEST VERIFIES THAT THE SERVO HAS SETTLED DOWN AND THAT
:*THE DRIVE IS ON CYLINDER WHEN THE DRIVE INDICATES SEEK COMPLETE.
:*RANDOM SEEKS ARE ISSUED BETWEEN CYLINDERS 'NC1' AND 'NC1+IC'

```

```

;*( 'NC1' STARTS AT VALUE 'FC'). AT THE COMPLETION OF 1000 (10) SEEKS,
;*'NC1' IS INCREMENTED BY VALUE 'IC' AND THE SEQUENCE IS REPEATED.
;*THE TEST IS COMPLETED WHEN 'NC1' HAS BEEN INCREMENTED BEYOND 'LC'.
;*
;*WHEN THE SEEK COMPLETES, THE PROGRAM READS THE DRIVE'S LOOK-AHEAD
;*REGISTER (RMLA) TO DETERMINE THE ADDRESS OF THE SECTOR ROTATING INTO
;*POSITION. THE PROGRAM THEN ISSUES A WRITE HEADER AND DATA COMMAND
;*FOR THAT SECTOR.
;*ERRORS IN THIS TEST INDICATE THAT THE SERVO SYSTEM MAY NOT BE ADJUSTED
;*CORRECTLY, THAT THE DRIVE IS MALFUNCTIONING, OR THAT A PACK WITH
;*MARGINAL SERVO TRACKS IS MOUNTED ON THE DRIVE.
;*
;*THIS TEST IS VALID ONLY IF THE OPERATION IS STARTED WITHIN A FEW
;*HUNDRED MICRO-SECONDS AFTER SEEK DONE OCCURS. THE NECESSARY
;*TIME DEPENDENT PARAMETERS OCCUR FREQUENTLY ENOUGH WITHIN THE REQUIRED
;*RANGE TO PERMIT THIS TEST TO BE EFFECTIVE.
;*****

```

TST10:

```

011620
011620 000240
011622 033737 001546 001330
011630 001002
011632 000137 012750
; NOP
; BIT BITS+<10*2>,TSTNMS ;DO THIS TEST?
; BNE .+6 ;BR IF YES
; JMP TST11 ;NO--JUMP TO TEST11

011636 012737 000010 001116
011644 004737 026672
011650 012737 012052 001124
011656 013737 002324 001220
011664 112737 000031 001131
011672 012737 000010 001240
; MOV #10,$TSTNM ;SET TEST #10 AND CLEAR ($ERFLG)
; JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
; MOV #TST10,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
; MOV RPT,$TIMES ;GET THE ITERATION COUNT
; MOVB #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
; MOV #10,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

381 011700 032777 010000 167246
011706 001406
011710 104401 046253
011714 013746 001240
011720 104403
011722 003
011723 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 3 ;;TYPE 3 DIGIT(S)
; .BYTE 0 ;;SUPPRESS LEADING ZEROS

382 011724 012737 011732 001122
383 011732
; MOV #18,$LPADR ;SETUP THE LOOP ADDRESS

011732 112737 000105 045460
011740 112737 000161 045540
011746 113737 002334 045547
011754 013737 002326 045470
011762 013737 002326 045550
011770 013737 002326 002356
011776 013737 002332 001446
012004 012737 176000 045542
012012 012737 052776 045544
012020 005000
012022 004737 032732
012026 005001
012030 113701 045456
012034 013704 037066
012040 004737 044772
012044 012737 012746 001346
012052
; MOVB #SEEK,DPB.A+2 ;SEEK=COMMAND
; MOVB #WRITE,DTADPB+2 ;COMMAND
; MOVB FT,DTADPB+11 ;TRACK ADDRESS FOR THE WRITE
; MOV FC,DPB.A+12 ;CYLINDER ADDRESS FOR THE SEEK
; MOV FC,DTADPB+12 ;CYLINDER ADDRESS FOR THE WRITE
; MOV FC,NC1 ;STARTING CYLINDER
; MOV IC,DELTA ;CYLINDER INCREMENT VALUE
; MOV #-<256.*4.>,DTADPB+4 ;WORD COUNT
; MOV #BUFFER,DTADPB+6 ;BUFFER ADDRESS
; CLR R0 ;PATTERN POINTER (WC PATTERN)
; JSR PC,SETBUF ;LOAD THE WRITE BUFFER
; CLR R1 ;CLEAR REGISTER
; MOVB DPB.A,R1 ;LOAD DRIVE ADDRESS
; MOV RMADR,R4 ;UNIBUS ADDRESS OF THE RH/RM
; JSR PC,CLRQUE ;CLEAR THE OPERATION QUEUES
; MOV #EXIT10,BYPASS ;ERROR EXIT FROM TEST

TEST10:

```



```

012052 012737 012052 001124      MOV      #.,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
012060 012706 001100      MOV      #STACK,SP    ;LOAD THE STACK POINTER
012064 012737 000340 177776      MOV      #PR7,PS      ;SET PRIORITY TO 7
012072 005737 001340      TST     CLKSTA        ;SEE WHICH CLOCK ON SYSTEM
012076 001415      BEQ     3$            ;BR IF NO CLOCK
012100 100405      BMI     1$            ;BR IF KW11-L CLOCK
012102 017746 167370      MOV      @PKV,-(SP)    ;SAVE THE VECTOR
012106 013746 001476      MOV      PKV,-(SP)    ;SAVE THE VECTOR ADDRESS
012112 000404      BR      2$            ;CONTINUE
012114 017746 167370      1$:     MOV      @LKV,-(SP) ;SAVE THE 'L' CLOCK VECTOR
012120 013746 001510      MOV      LKV,-(SP)    ;SAVE THE VECTOR ADDRESS
012124 012776 012702 000000 2$:     MOV      #TST10B,@(SP) ;CHANGE THE VECTOR
012132 012777 031272 024730 3$:     MOV      #DORTI,@RMVEC ;CHANGE THE RM VECTOR
012140 012737 000010 001442      MOV      #8.,SEKTMR   ;LOAD THE SEEK TIMER
012146 012764 000040 000010      MOV      #CLR,RMCS2(R4) ;INIT THE MASSBUS
012154 110164 000010      MOV     R1,RMCS2(R4) ;RESELECT THE DRIVE
012160 013764 045470 000034      MOV     DPB.A+12,RMDC(R4) ;LOAD THE CYLINDER ADDRESS
012166 013737 045470 001364      MOV     DPB.A+12,CYL.DS ;CYLINDER ADDRESS FOR ERROR MESSAGE
012174 112764 000105 000000      MOV     #SEEK,RMCS1(R4) ;START THE SEEK
012202 005037 177776      CLR     PS            ;CLEAR THE PRIORITY
012206 105764 000012      4$:     TST     RMDS(R4)    ;HAS THE DRIVE FINISHED ?
012212 100402      BMI     5$            ;BR IF IT HAS
012214 000001      WAIT                    ;WAIT FOR THE OPERATION TO COMPLETE
012216 000773      BR      4$            ;CONTINUE
012220 012737 000340 177776 5$:     MOV      #PR7,PS      ;CHANGE PRIORITY TO MAX
012226 032764 040000 000012      BIT     #ERR,RMDS(R4) ;ERROR ?
012234 001412      BEQ     6$            ;BR IF NOT
012236 012702 045456      MOV     #DPB.A,R2     ;DPB POINTER
012242 004737 044310      JSR     PC,SVRH70     ;SAVE THE REGISTERS
012246 104023      EMT     23
012250 012764 000040 000010      MOV     #CLR,RMCS2(R4) ;INIT THE MASSBUS
012256 110164 000010      MOV     R1,RMCS2(R4) ;RESELECT THE DRIVE
012262 012777 041740 024600 6$:     MOV      #ISR,@RMVEC  ;SETUP THE RM VECTOR
012270 005737 001340      TST     CLKSTA        ;WHICH CLOCK
012274 001405      BEQ     TST10A        ;BR IF NONE
012276 016676 000002 000000      MOV     2(SP),@(SP)   ;RELOAD THE CLOCK VECTOR
012304 062706 000004      ADD     #4,SP         ;CORRECT THE STACK POINTER
012310      TST10A:
439 012310 012737 012310 001124      MOV      #.,$LPERR    ;SETUP THE ERROR LOOP ADDRESS
012316 012706 001100      MOV      #STACK,SP   ;LOAD THE STACK POINTER
440 012322 110164 000010      MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
441 012326 016446 000020      MOV     RMLA(R4),-(SP) ;GET THE LOOK AHEAD REGISTER
442 012332 006316      ASL     (SP)          ;ALIGN THE SECTOR ADDRESS
443 012334 006316      ASL     (SP)          ;ALIGN THE SECTOR ADDRESS
444 012336 000316      SWAB   (SP)          ;PUT ADDRESS IN LOWER BYTE
445 012340 122766 000300 000001      CMP     #300,1(SP)    ;IN THE LAST 20% OR SECTOR ?
446 012346 001001      BNE     2$            ;BR IF NOT
447 012350 105216      INCB   (SP)          ;INCREMENT THE SECTOR ADDRESS
448 012352 105216      2$:     INCB   (SP)          ;INCREMENT THE SECTOR ADDRESS
449 012354 112637 045546      MOV     (SP)+,DTADPB+10 ;LOAD THE DPB
450 012360 013746 002456      MOV     PRMLM+24,-(SP) ;PUT MAXIMUM SECTOR ADDRESS ON THE STACK
451 012364 005216      INC    (SP)          ;INCREMENT PAST THE MAXIMUM ADDRESS
452 012366 122637 045546      CMP     (SP)+,DTADPB+10 ;NEW SECTOR ADDRESS TOO LARGE ?
453 012372 101007      BHI     4$            ;BR IF NOT
454 012374 103403      BLO     3$            ;BR IF ADDRESS IS 2 GREATER THAN MAXIMUM
455 012376 105037 045546      CLRB   DTADPB+10     ;RESET TO SECTOR ADDRESS 0
456 012402 000403      BR      4$            ;CONTINUE

```

```

457 012404 112737 000001 045546 3$:   MOVB  #1,DTADPB+10 ;RESET ADDRESS TO SECTOR 1
458 012412 012703 045542          4$:   MOV  #DTADPB+4,R3 ;POINTER
459 012416 012764 000111 000000   MOV  #DRVCLR,RMCS1(R4) ;CLEAR THE DRIVE
460 012424 012364 000002   MOV  (R3)+,RMWC(R4) ;LOAD THE WORD COUNT
461 012430 012364 000004   MOV  (R3)+,RMBA(R4) ;LOAD THE BUFFER ADDRESS
462 012434 012364 000006   MOV  (R3)+,RMDA(R4) ;LOAD THE TRACK/SECTOR ADDR
463 012440 005037 045554   CLR  DTADPB+16 ;RESET 'DONE' INDICATOR
464 012444 012737 045536 037000   MOV  #DTADPB,TRNSWT ;LOAD 'TRANSFER' DPB ADDRESS
465 012452 010137 037052   MOV  R1,DTUW ;ADDRESS OF DRIVE TRANSFERRING
466 012456 112761 000001 036730   MOVB #1,DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR
467 012464 006301          ASL  R1 ;SHIFT DRIVE ADDRESS
468 012466 012761 001750 037032   MOV  #1000.,TIMER(R1) ;SETUP THE OPERATION TIMER
469 012474 006201          ASR  R1 ;RESTORE R1
470 012476 013764 045540 000000   MOV  DTADPB+2,RMCS1(R4) ;START THE OPERATION
471 012504 005037 177776          CLR  PS ;CLEAR THE PRIORITY
472 012510 004037 027710          JSR  RO,DRVCL1 ;WAIT FOR OPERATION TO COMPLETE
473 012514 023727 001444 001750 5$:   CMP  SEKCNT,#1000. ;FINISHED SEEKS ?
474 012522 001026          BNE  6$ ;BR IF NOT
475 012524 005037 001444          CLR  SEKCNT ;CLEAR THE SEEK COUNT
476 012530 063737 002332 002356   ADD  IC,NC1 ;ADD THE INCREMENT
477 012536 023737 002356 002330   CMP  NC1,LC ;EXCEEDED THE CYLINDER LIMIT ?
512 012544 103100          BHIS EXIT10 ;BR IF IT HAS
      012546 013737 002330 001446   MOV  LC,DELTA ;GET THE NEXT 'ZONE' ADDRESS
      012554 163737 002356 001446   SUB  NC1,DELTA ;CHECK THE DIFFERENCE
      012562 023737 002332 001446   CMP  IC,DELTA ;DIFFERENCE GREATER THAN THE INCREMENT ?
      012570 101003          BHI  6$ ;BR IF IT IS
      012572 013737 002332 001446   MOV  IC,DELTA ;USE THE INCREMENT PARAMETER
      012600 005237 001444          INC  SEKCNT ;COUNT THE NEXT SEEK
      012604 023737 002326 002330 6$:   CMP  FC,LC ;BEGINNING AND ENDING CYLINDERS THE SAME ?
      012612 001002          BNE  7$ ;BR IF NOT
      012614 000137 012052          JMP  TEST10 ;BR IF THEY ARE
      012620 013737 002356 045470 7$:   MOV  NC1,DPB.A+12 ;RESET THE CYLINDER ADDRESS
      012626 004737 025550          JSR  PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
      012632 013746 025646          MOV  $HINUM,-(SP) ;USE THE HIGH RANDOM NUMBER
      012636 005046          CLR  -(SP) ;CLEAR THE UPPER DIVIDEND
      012640 013746 001446          MOV  DELTA,-(SP) ;FORM THE DIVISOR
      012644 005216          INC  (SP) ;INCREMENT
      012646 004737 025652          JSR  PC,$DIV ;DIVIDE
      012652 062637 045470          ADD  (SP)+,DPB.A+12 ;ADD THE REMAINDER TO THE INITIAL CYLINDER
      012656 005726          TST  (SP)+ ;DISCARD THE QUOTIENT
      012660 023737 045470 045550   CMP  DPB.A+12,DTADPB+12 ;SAME CYLINDER SELECTED AS LAST TIME ?
      012666 001754          BEQ  7$ ;BR IF IT WAS
      012670 013737 045470 045550   MOV  DPB.A+12,DTADPB+12 ;COPY NEW CYLINDER ADDRESS
      012676 000137 012052          JMP  TEST10 ;CONTINUE
      012702          TST10B:
      012702 005337 001442          DEC  SEKTMR ;DECREMENT THE SEEK TIMER
      012706 001016          BNE  1$ ;CONTINUE IF NOT DONE
      012710 012702 045456          MOV  #DPB.A,R2 ;DPB ADDRESS
      012714 004737 044310          JSR  PC,SVRH70 ;SAVE THE REGISTERS
      012720 104024          EMT  24
      012722 012764 000040 000010   MOV  #CLR,RMCS2(R4) ;INIT THE MASSBUS
      012730 110164 000010          MOVB R1,RMCS2(R4) ;RESELECT THE DRIVE
      012734 016676 000002 000000   MOV  2(SP),@(SP) ;RESTORE THE CLOCK VECTOR ADDRESS
      012742 000401          BR   EXIT10 ;ABORT THE TEST
513 012744 000002          1$: RTI ;RETURN
514 012746 000004          EXIT10: SCOPE ;CALL SCOPE ROUTINE
515

```

526
527

```

*****
*TEST 11      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.
*****

```

```

012750
012750 000240
012752 033737 001550 001330
012760 001002
012762 000137 013214

012766 012737 000011 001116
012774 004737 026672
013000 012737 013134 001124
013006 013737 002324 001220
013014 112737 000031 001131
013022 012737 000011 001240

528 013030 032777 010000 166116
013036 001406
013040 104401 046253
013044 013746 001240
013050 104403
013052 003
013053 000

529 013054 012737 013062 001122
530 013062 113737 002342 045506
531 013070 113737 002342 045526
532 013076 113737 002334 045507
533 013104 113737 002334 045527
534 013112 013737 002326 045510
535 013120 013737 002326 045530
539 013126 012737 013212 001346

013134
540 013134 012706 001100
541 013140
013140 004037 027466
542 013144 004037 027264
543 013150 063737 002332 045530
544 013156 023737 002330 045530
545 013164 002365
546 013166 013737 002326 045530
547 013174 063737 002332 045510
548 013202 023737 002330 045510
549 013210 002353
550 013212 000004
551

TST11:
NOP
BIT BITS+<11*2>,TSTNMS ;DO THIS TEST?
BNE .+6 ;BR IF YES
JMP TST12 ;NO--JUMP TO TEST12

MOV #11,$STNM ;SET TEST #11 AND CLEAR ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TEST11,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #11,$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

MOV #1$, $LPADR ;SETUP THE LOOP ADDRESS
1$: MOVB FS,DPB.B+10 ;SECTOR ADDRESS
MOVB FS,DPB.C+10 ;SECTOR ADDRESS
MOVB FT,DPB.B+11 ;TRACK ADDRESS
MOVB FT,DPB.C+11 ;TRACK ADDRESS
MOV FC,DPB.B+12 ;STARTING CYLINDER ADDRESS
MOV FC,DPB.C+12 ;STARTING CYLINDER ADDRESS
MOV #EXIT11,BYPASS ;TEST ABORT EXIT

TEST11:
MOV #STACK,SP ;SETUP THE STACK POINTER
1$: JSR RO,CALL.C ;GO EXECUTE THE COMMAND
JSR RO,CALL.B ;GO EXECUTE THE COMMAND
ADD IC,DPB.C+12 ;INCREMENT THE ENDING CYLINDER ADDRESS
CMP LC,DPB.C+12 ;CHECK IF EXCEEDING MAXIMUM
BGE 1$ ;BR IF NOT
MOV FC,DPB.C+12 ;RESET ENDING CYLINDER ADDRESS
ADD IC,DPB.B+12 ;INCREMENT THE STARTING ADDRESS
CMP LC,DPB.B+12 ;EXCEEDING MAXIMUM ?
BGE 1$ ;BR IF NOT
EXIT11: SCOPE ;CALL SCOPE ROUTINE

```

1
2
3
4
5
6
7
8
9
10
11
12
28
29

.SBTTL TIMING TESTS

```

:;////////////////////////////////////
:*THE TIMING TESTS WILL ENSURE THAT THOSE FUNCTIONS BEING
:*TIMED ARE WITHIN THE TOLERANCES SPECIFIED IN THE 'RM05/3/2
:*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 12      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:
:*
:*      RM05/3:
:*          16.67 MS/REV + OR - 2% IF 60HZ
:*          16.67 MS/REV + OR - 2.5% IF 50HZ.
:*
:*      RM02:
:*          25.00 MS/REV + OR - 2% IF 60HZ
:*          25.00 MS/REV + OR - 2.5% IF 50HZ.
:*****
    
```

013214
013214 000240
013216 033737 001552 001330
013224 001002
013226 000137 014136

013232 012737 000012 001116
013240 004737 026672
013244 012737 014136 001124
013252 013737 002324 001220
013260 112737 000031 001131
013266 012737 000012 001240

30 013274 032777 010000 165652
013302 001406
013304 104401 046253
013310 013746 001240
013314 104403
013316 003
013317 000

31 013320 005737 001340
32 013324 003002
35 013326 000137 014134
36 013332 012737 013332 001122 1\$:
37 013340 004037 031110
38 013344 000402
60 013346 000137 014134

```

TST12:
NOP
BIT     BITS+<12*2>,TSTNMS      ;DO THIS TEST?
BNE     .+6                      ;BR IF YES
JMP     TST13                    ;NO--JUMP TO TEST13

MOV     #12,$TSTNM              ;SET TEST #12 AND CLEAR ($ERFLG)
JSR     PC,LODPRM              ;LOAD THE PARMETERS FOR THE TEST
MOV     #TST13,$LPERR          ;SETUP THE ERROR LOOP ADDRESS
MOV     RPT,$TIMES             ;GET THE ITERATION COUNT
MOVB    #25,$ERMAX             ;MAX ERRORS ALLOWED FOR TEST
MOV     #12,$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     EXIT12                 ;NO--JUMP TO EXIT12
MOV     #.,$LPADR              ;SETUP LOOP ADDRESS
JSR     RO,SRCH00              ;DO A MASSBUS INIT & RECAL
BR      2$                     ;RETURN HERE IF NO ERROR
JMP     EXIT12                 ;RETURN HERE IF ERROR
    
```

```

013352 012737 013352 001122 2$: MOV #,$LPADR ;ERROR LOOP ADDRESS
013360 112737 000105 045460 MOVB #SEEK,DPB.A+2 ;SEEK=COMMAND
013366 005037 045466 CLR DPB.A+10 ;USE TRACK 0 & SECTOR 0
013372 013737 002326 045470 MOV FC,DPB.A+12 ;STARTING CYLINDER
013400 012737 014134 001346 MOV #EXIT12,BYPASS ;GO TO EXIT12 IF ERROR
013406 004037 027132 JSR RO,CALL.A ;GO EXECUTE THE COMMAND
013412 013764 002326 000034 MOV FC,RMDC(R4) ;FC
013420 013766 002342 MOV FS,-(SP) ;FS
013424 113766 002334 000001 MOVB FT,1(SP) ;FT
013432 012664 000006 MOV SP)+,RMDA(R4) ;LOAD FT/FS
013436 012737 014134 001222 MOV #EXIT12,$ESCAPE ;ESCAPE TO EXIT12 ON ERROR
013444 005005 CLR R5 ;COUNT UP

61 ;SETUP PARAMETER TABLE FOR RM05/RM03/RM02
62 013446 010046 MOV RO,-(SP) ;SAVE RO
63 013450 113700 045536 MOVB DTADPB,R0 ;DRIVE ADDRESS
64 013454 032737 000100 001314 BIT #SW06,C.SWR ;CHECK CONTROL SWR FOR 60 HZ.
65 013462 001425 BEQ 3$ ;BR IF YES
66 013464 012703 001636 MOV #T7B1,R3 ;LOAD 50 HZ. TABLE FOR RM02
67 013470 012737 001636 014124 MOV #T7B1,TP50 ;
68 013476 012737 001720 014132 MOV #SP7B1,TP50 ;
69 013504 122760 000005 036750 CMPB #5,DRV1YP(RO) ;AN RM02 DRIVE ?
70 013512 001435 BEQ 4$ ;BRANCH IF SO
71 013514 012703 001616 MOV #T7B,R3 ;LOAD 50 HZ. TABLE FOR RM05/3
72 013520 012737 001616 014124 MOV #T7B,TP50 ;
73 013526 012737 001704 014132 MOV #SP7B,TP50 ;
74 013534 000424 BR 4$ ;EXIT
75 013536 012737 001626 014106 3$: MOV #T7A1,TP60 ;LOAD 60 HZ. TABLE FOR RM02
76 013544 012737 001712 014114 MOV #SP7A1,TP60 ;
77 013552 012703 001626 MOV #T7A1,R3 ;
78 013556 122760 000005 036750 CMPB #5,DRV1YP(RO) ;AN RM02 DRIVE ?
79 013564 001410 BEQ 4$ ;BRANCH IF SO
80 013566 012737 001606 014106 MOV #T7A,TP60 ;LOAD 60 HZ. TABLE FOR RM05/3
81 013574 012703 001606 MOV #T7A,R3 ;
82 013600 012737 001676 014114 MOV #SP7A,TP60 ;
83 013606 012600 4$: MOV (SP)+,RO ;RESTORE RO
84 013610 000240 NOP ;EXIT
87 013612 TEST12:
88 013612 012706 001100 MOV #STACK,SP ;SETUP STACK
89 013616 012701 000012 MOV #10,R1 ;TIME 10 SEARCHES
90 013622 004737 031274 JSR PC,STRTMR ;INITIALIZE THE TIMERS
91 013626 012777 014014 165642 MOV #7$,@PKV ;SETUP VECTOR IN CASE OF OVERFLOW
92 013634 012777 031272 023226 MOV #DORT1,@RMVEC ;SETUP RM VECTOR
93 013642 005077 165636 1$: CLR @PKB ;START COUNTING AT ZERO
94 013646 012777 000131 165626 MOV #131,@PKCS ;INT.EN., COUNT UP AT 100KHZ
95 013654 012714 000131 MOV #SEARCH,(R4) ;START A SEARCH
96 013660 000001 WAIT ;WAIT ON INTERRUPT
97 013662 042777 000101 165612 BIC #101,@PKCS ;STOP THE CLOCK
98 013670 032764 040000 000012 BIT #ERR,RMDS(R4) ;ERROR?
99 013676 001411 BEQ 2$ ;NO--BRANCH
100 013700 104412 SAVREG ;SAVE R0-R5
013702 012702 045536 MOV #DTADPB,R2 ;DPB POINTER
013706 004737 044310 JSR PC,SVRH70 ;SAVE ALL THE RM/RM REGISTERS
013712 004737 026072 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
013716 104413 RESREG ;RESTORE R0-R5
101 013720 104017 EMT 17
102 013722 005077 165556 2$: CLR @PKB ;START THE COUNT AT ZERO
  
```

```

103 013726 012714 000131      MOV      #SEARCH,(R4)      ;START A SEARCH
104 013732 012777 000131 165542  MOV      #131,@PKCS      ;START THE CLOCK
105 013740 000001      WAIT      ;WAIT ON INTERRUPT
106 013742 042777 000101 165532  BIC      #101,@PKCS      ;STOP THE CLOCK
107 013750 032764 040000 000012  BIT      #ERR,RMDS(R4)    ;IS 'ERR=1'?
108 013756 001411      BEQ      3$              ;NO--BRANCH
109 013760 104412      SAVREG      ;SAVE R0-R5
      013762 012702 045536      MOV      #DTADPB,R2      ;DPB POINTER
      013766 004737 044310      JSR      PC,SVRH70      ;SAVE ALL THE RH/RM REGISTERS
      013772 004737 026072      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
      013776 104413      RESREG      ;RESTORE R0-R5
110 014000 104017      EMT      17
111 014002 004737 031336      3$:      JSR      PC,COUNT      ;UPDATE THE COUNT
112 014006 005301      DEC      R1              ;DONE?
113 014010 003314      BGT      1$              ;NO--BRANCH
114 014012 000420      BR      8$              ;YES--GO TO THE EXIT
115 014014 042777 000101 165460  7$:      BIC      #101,@PKCS      ;STOP THE CLOCK
116 014022 005037 177776      CLR      PS              ;DROP THE PRIORITY
117 014026 012600      MOV      (SP)+,R0        ;PC OF WAIT+2
118 014030 005726      TST      (SP)+          ;POP THE PS FROM THE STACK
119 014032 104412      SAVREG      ;SAVE R0-R5
      014034 012702 045536      MOV      #DTADPB,R2      ;DPB POINTER
      014040 004737 044310      JSR      PC,SVRH70      ;SAVE ALL THE RH/RM REGISTERS
      014044 004737 026072      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
      014050 104413      RESREG      ;RESTORE R0-R5
120 014052 104020      EMT      20
121 014054 004737 026072      8$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
122 014060 004737 026156      JSR      PC,ST.CLK      ;INITIALIZE THE CLOCK
123 014064 012777 041740 022776  MOV      #ISR,@RMVEC      ;RESTORE RH/RM INT. VECTOR
124 014072 032737 000100 001314  BIT      #SW06,C.SWR      ;60 HZ?
138 014100 001007      BNE      EXIT.A          ;NO-BRANCH
      014102 004037 031576      JSR      RO,TYPTIM      ;TYPE THE TIMING
      014106 001606      TP60:   T7A              ;TABLE ADDRESS
      014110 004037 031470      JSR      RO,SPTYP      ;TYPE THE SPEC
      014114 001676      TPS60:  SP7A
      014116 000406      BR      EXIT12          ;EXIT
139 014120 004037 031576      EXIT.A: JSR      RO,TYPTIM      ;TYPE THE TIMING
140 014124 001616      TP50:   T7B              ;TABLE ADDRESS
141 014126 004037 031470      JSR      RO,SPTYP
142 014132 001704      TPS50:  SP7B
143 014134 000004      EXIT12: SCOPE          ;CALL SCOPE ROUTINE

```

144
153
154

```

:*****
:*TEST 13      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.
:*THE TIME MUST BE LESS THAN 10MS.
:*****

```

```

TST13:
014136      NOP
014136 000240      BIT      BITS+<13*2>,TSTNMS ;DO THIS TEST?
014140 033737 001554 001330      BNE      +6              ;BR IF YES
014146 001002      JMP      TST14          ;NO--JUMP TO TEST14
014150 000137 014540

```

```

014154 012737 000013 001116      MOV      #13,$STSTM      ;SET TEST #13 AND CLEAR ($ERFLG)
014162 004737 026672                JSR      PC,LODPRM      ;LOAD THE PARMETERS FOR THE TEST
014166 012737 014640 001124      MOV      #TST14,$LPERR  ;SETUP THE ERROR LOOP ADDRESS
014174 013737 002324 001220      MOV      RPT,$TIMES     ;GET THE ITERATION COUNT
014202 112737 000031 001131      MOV      #25,$SERMAX    ;MAX ERRORS ALLOWED FOR TEST
014210 012737 000013 001240      MOV      #13,$TESTN     ;:SET TEST NUMBER IN APT MAIL BOX
155
014216 032777 010000 164730      BIT      #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
014224 001406                BEQ      .+16            ;BR IF YES
014226 104401 046253                TYPE     ,MSGTST        ;TYPE 'TEST'
014232 013746 001240                MOV      $TESTN,-(SP)   ;:SAVE $TESTN FOR TYPEOUT
014236 104403                TYPOS    ;:GO TYPE--OCTAL ASCII
014240          003                .BYTE   3               ;:TYPE 3 DIGIT(S)
014241          000                .BYTE   0               ;:SUPPRESS LEADING ZEROS
156 014242 005737 001340      TST      CLKSTA         ;KW11-P CLOCK?
157 014246 003002                BGT      1$             ;YES--START TEST
160 014250 000137 014636      JMP      EXIT13         ;NO--JUMP TO EXIT13
161 014254 012737 014254 001122 1$:  MOV      #,$LPADR       ;SETUP THE LOOP ADDRESS
162 014262 004037 031110      JSR      R0,SRCH00      ;DO A MASSBUS INIT. AND RECAL
163 014266 000402                BR       2$             ;NO ERROR RETURN
175 014270 000137 014636      JMP      EXIT13         ;ERROR RETURN--SCOPE LOOP CALL
014274 012737 014274 001122 2$:  MOV      #,$LPADR       ;ERROR LOOP ADDRESS
014302 112737 000105 045460      MOV      #SEEK,DPB.A+2  ;SEEK=COMMAND
014310 005037 045466      CLR      DPB.A+10       ;USE TRACK 0 & SECTOR 0
014314 013737 002326 045470      MOV      FC,DPB.A+12    ;STARTING CYLINDER
014322 012737 014636 001346      MOV      #EXIT13,BYPASS ;GO TO EXIT13 IF ERROR
014330 004037 027132      JSR      R0,CALL.A      ;GO EXECUTE THE COMMAND
014334 012703 001646      MOV      #T10,R3        ;PARAMETER POINTER
014340 012737 014636 001222      MOV      #EXIT13,$ESCAPE ;:ESCAPE TO EXIT13 ON ERROR
014346
TEST13:
176 014346 012706 001100      MOV      #STACK,SP      ;SETUP STACK
177 014352 013737 002326 045550      MOV      FC,DTADPB+12   ;START WITH BEGINNING CYLINDER
178 014360 005237 045550      INC      DTADPB+12      ;INCREMENT THE BEGINNING CYLINDER
179 014364 005005                CLR      R5             ;SET THE UP/DOWN SWITCH TO UP
180 014366 004737 031274      JSR      PC,STRMR       ;INITIALIZE THE TIMERS
181 014372 012777 014544 165076      MOV      #7$,@PKV       ;SETUP INCASE OF OVERFLOW
182 014400 012777 031272 022462      MOV      #DORTI,@RMVEC  ;SET RM VECTOR
183 014406 005077 165072      CLR      @PKB           ;START THE COUNTER AT ZERO
184 014412 013764 045550 000034 1$:  MOV      DTADPB+12,RMDC(R4) ;LOAD DESIRED CYLINDER
185 014420 012714 000105      MOV      #SEEK,(R4)     ;START A SEEK
186 014424 012777 000131 165050      MOV      #131,@PKCS     ;START THE CLOCK
187 014432 000001                WAIT    ;WAIT ON INTERRUPT
188 014434 042777 000101 165040      BIC      #101,@PKCS     ;STOP THE CLOCK
189 014442 032764 040000 000012      BIT      #ERR,RMDS(R4)  ;ANY DISK ERRORS?
190 014450 001411                BEQ      2$             ;NO--BRANCH
191 014452 104412                SAVREG  ;SAVE R0-R5
014454 012702 045536      MOV      #DTADPB,R2     ;DPB POINTER
014460 004737 044310      JSR      PC,SVRH70      ;SAVE ALL THE RH/RM REGISTERS
014464 004737 026072      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
014470 104413                RESREG  ;RESTORE R0-R5
192 014472 104017                EMT      17
193 014474 004737 031336      JSR      PC,COUNT       ;COUNT THIS SEEKS TIME
194 014500 005705                TST      R5             ;UP OR DOWN?
195 014502 001011                BNE     4$             ;DOWN--BRANCH
196 014504 005237 045550      3$:  INC      DTADPB+12      ;MOVE TO NEXT CYLINDER

```

```

197 014510 023737 045550 002330      CMP      DTADPB+12,LC      ;OUT OF CYLINDERS?
198 014516 002733                    BLT      1$              ;NO--GO DO THE NEXT SEEK
199 014520 012705 177777              MOV      #-1,R5          ;SET UP/DOWN SWITCH TO DOWN
200 014524 000730                    BR       1$              ;GO DO THE NEXT SEEK
201 014526 005337 045550 002326 4$:  DEC      DTADPB+12        ;MOVE TO NEXT CYLINDER
202 014532 023737 045550 002326      CMP      DTADPB+12,FC      ;OUT OF CYLINDERS?
203 014540 003322                    BGT      1$              ;NO--GO DO THE NEXT SEEK
204 014542 000420                    BR       8$              ;GO TO THE EXIT
205 014544 042777 000101 164730 7$:  BIC      #101,@PKCS       ;STOP THE CLOCK
206 014552 005037 177776              CLR      PS              ;DROP THE PRIORITY
207 014556 012600                    MOV      (SP)+,RO        ;PC OF WAIT+2
208 014560 005726                    TST      (SP)+          ;POP THE PS FROM THE STACK
209 014562 104412                    SAVREG                    ;SAVE R0-R5
    014564 012702 045536                    MOV      #DTADPB,R2      ;DPB POINTER
    014570 004737 044310                    JSR      PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
    014574 004737 026072                    JSR      PC,CNTCLR       ;GO CLEAR CONTROLLER
    014600 104413                    RESREG                    ;RESTORE R0-R5
210 014602 104020                    EMT      20
211 014604 004737 026072 8$:      JSR      PC,CNTCLR       ;GO CLEAR CONTROLLER
212 014610 004737 026156                    JSR      PC,ST.CLK       ;INITIALIZE THE CLOCK
213 014614 012777 041740 022246      MOV      #ISR,@RMVEC     ;RESTORE RH/RM INT. VECTOR
214 014622 004037 031576                    JSR      RO,TYPTIM       ;GO TYPE THE TIMES
    014626 001646                    TIO
215 014630 004037 031470                    JSR      RO,SPTYP
216 014634 001726                    SP10
217 014636 000004                    EXIT13: SCOPE           ;CALL .COPE ROUTINE
218
227
228

```

```

:*****
:*TEST 14      ACCESS TIME MEASUREMENT 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 ACCESS TIME MEASUREMENT.
:*THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS).
:*THE AVERAGE ACCESS TIME MUST BE LESS THAN 30 MS.
:*****

```

```

014640
014640 000240
014642 033737 001556 001330      NOP
014650 001002                    BIT      BITS+<14*2>,TSTNMS ;DO THIS TEST?
014652 000137 015400                    BNE     .+6              ;BR IF YES
                                JMP      TST15           ;NO--JUMP TO TEST15

014656 012737 000014 001116      MOV      #14,$TSTNM      ;SET TEST #14 AND CLEAR ($ERFLG)
014664 004737 026672                    JSR      PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
014670 012737 015400 001124      MOV      #TST15,$LPERR   ;SETUP THE ERROR LOOP ADDRESS
014676 013737 002324 001220      MOV      RPT,$TIMES      ;GET THE ITERATION COUNT
014704 112737 000031 001131      MOV      #25,,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
014712 012737 000014 001240      MOV      #14,$TESTN      ;SET TEST NUMBER IN APT MAIL BOX

229 014720 032777 010000 164226      BIT      #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
014726 001406                    BEQ     .+16             ;BR IF YES
014730 104401 046253                    TYPE    $MSGTST          ;TYPE 'TEST'
014734 013746 001240                    MOV     $TESTN,-(SP)     ;SAVE $TESTN FOR TYPEOUT
014740 104403                    TYPOS                    ;GO TYPE--OCTAL ASCII
014742 003                          .BYTE  3                 ;TYPE 3 DIGIT(S)
014743 000                          .EYTE  0                 ;SUPPRESS LEADING ZEROS

```


230	014744	005737	001340		TST	CLKSTA	;KW11-P CLOCK?	
231	014750	003002			BGT	1\$;YES--START TEST	
234	014752	000137	015376		JMP	EXIT14	;NO--JUMP TO EXIT14	
235	014756	012737	014756	001122	1\$:	MOV	#,\$LPADR	;SET THE LOOP ADDRESS
236	014764	004037	031110		JSR	RO,SRCH00	;DO A MASSBUS INIT & RECAL	
237	014770	000402			BR	2\$;RETURN HERE IF NO ERROR	
249	014772	000137	015376		JMP	EXIT14	;RETURN HERE ON ERROR	
	014776	012737	014776	001122	2\$:	MOV	#,\$LPADR	;ERROR LOOP ADDRESS
	015004	112737	000105	045460		MOVB	#SEEK,DPB.A+2	;SEEK=COMMAND
	015012	005037	045466		CLR	DPB.A+10	;USE TRACK 0 & SECTOR 0	
	015016	013737	002326	045470		MOV	FC,DPB.A+12	;STARTING CYLINDER
	015024	012737	015376	001346		MOV	#EXIT14,BYPASS	;GO TO EXIT14 IF ERROR
	015032	004037	027132		JSR	RO,CALL.A	;GO EXECUTE THE COMMAND	
	015036	012703	001656		MOV	#T11,R3	;PARAMETER POINTER	
	015042	012737	015376	001222		MOV	#EXIT14,\$ESCAPE	;ESCAPE TO EXIT14 ON ERROR
	015050				TEST14:			
250	015050	012706	001100		MOV	#STACK,SP	;SETUP STACK	
251	015054	012701	000200		MOV	#128.,R1	;REPEAT "'FC'-'LC'-'FC'" 128 TIMES	
252	015060	004737	031274		JSR	PC,STRMR	;INIT. THE COUNTERS	
253	015064	012777	015304	164404		MOV	#7\$,@PKV	;SET UP VECTOR IN CASE OF OVERFLOW
254	015072	012777	031272	021770		MOV	#DORT1,@RMVEC	;SETUP RM VECTOR
255	015100	005077	164400		1\$:	CLR	@PKB	;START COUNT AT ZERO
256	015104	013764	002330	000034		MOV	LC,RMDC(R4)	; 'MIDDLE' CYLINDER
257	015112	012764	000105	000000		MOV	#SEEK,RMCS1(R4)	;START A SEEK
258	015120	012777	000131	164354		MOV	#131,@PKCS	;START THE CLOCK
259	015126	000001				WAIT		;WAIT ON INTERRUPT
260	015130	042777	000101	164344		BIC	#101,@PKCS	;STOP CLOCK
261	015136	032764	040000	000012		BIT	#ERR,RMDS(R4)	;ERR=1?
262	015144	001411				BEQ	2\$;NO--BRANCH
263	015146	104412				SAVREG		;SAVE R0-R5
	015150	012702	045536			MOV	#DTADPB,R2	;DPB POINTER
	015154	004737	044310			JSR	PC,SVRH70	;SAVE ALL THE RH/RM REGISTERS
	015160	004737	026072			JSR	PC,CNTCLR	;GO CLEAR CONTROLLER
	015164	104413				RESREG		;RESTORE R0-R5
264	015166	104017				EMT	17	
265	015170	005005			2\$:	CLR	R5	;SET UP/DOWN SWITCH TO UP
266	015172	004737	031336			JSR	PC,COUNT	;UPDATE THE COUNT
267	015176	005077	164302			CLR	@PKB	;START THE COUNT AT ZERO
268	015202	013764	002326	000034		MOV	FC,RMDC(R4)	;BEGINNING CYLINDER
269	015210	012764	000105	000000		MOV	#SEEK,RMCS1(R4)	;START A SEEK
270	015216	012777	000131	164256		MOV	#131,@PKCS	;START THE CLOCK
271	015224	000001				WAIT		;WAIT ON INTERRUPT
272	015226	042777	000101	164246		BIC	#101,@PKCS	;STOP THE CLOCK
273	015234	032764	040000	000012		BIT	#ERR,RMDS(R4)	;ERR=1?
274	015242	001411				BEQ	3\$;NO--BRANCH
275	015244	104412				SAVREG		;SAVE R0-R5
	015246	012702	045536			MOV	#DTADPB,R2	;DPB POINTER
	015252	004737	044310			JSR	PC,SVRH70	;SAVE ALL THE RH/RM REGISTERS
	015256	004737	026072			JSR	PC,CNTCLR	;GO CLEAR CONTROLLER
	015262	104413				RESREG		;RESTORE R0-R5
276	015264	104017				EMT	17	
277	015266	012705	177777		3\$:	MOV	#-1,R5	;SET UP/DOWN SWITCH TO DOWN
278	015272	004737	031336			JSR	PC,COUNT	;UPDATE THE COUNT
279	015276	005501				DEC	R1	;DONE?
280	015300	003277				BGT	1\$;NO--BRANCH
281	015302	000420				BR	8\$;YES--EXIT

```

282 015304 042777 000101 164170 7$: BIC #101,@PKCS ;STOP THE CLOCK
283 015312 005037 177776 CLR PS ;DROP THE PRIORITY
284 015316 012600 MOV (SP)+,RO ;PC OF WAIT+2
285 015320 005726 TST (SP)+ ;POP THE PS FROM THE STACK
286 015322 104412 SAVREG ;SAVE R0-R5
015324 012702 045536 MOV #DTADPB,R2 ;DPB POINTER
015330 004737 044310 JSR PC,SVRH70 ;SAVE ALL THE RH/RM REGISTERS
015334 004737 045072 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
015340 104413 RESREG ;RESTORE R0-R5
287 015342 104020 EMT 20
288 015344 004737 026072 8$: JSR PC,CNTCLR ;GO CLEAR CONTROLLER
289 015350 004737 026156 JSR PC,ST.CLK ;INITIALIZE THE CLOCK
290 015354 012777 041740 021506 MOV #ISR,@RMVEC ;RESTORE RH/RM INT. VECTOR
291 015362 004037 031576 JSR RO,TYPTIM ;GO TYPE THE TIMES
015366 001656 T11 ;POINTER
292 015370 004037 031470 JSR RO,SPTYP
293 015374 001734 SP11
294 015376 000004 EXIT14: SCOPE ;CALL SCOPE ROUTINE
295
304
305

```

```

:*****
:*TEST 15 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 AND 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). THE MAXIMUM SEEK TIME MUST BE LESS THAN 54 MS.
:*****
TST15:

```

```

015400
015400 000240
015402 033737 001560 001330 NGP
015410 001002 BIT BITS+<15*2>,TSTNMS ;DO THIS TEST?
015412 000137 016140 BNE .+6 ;BR IF YES
JMP TST16 ;NO--JUMP TO TEST16

015416 012737 000015 001116 MOV #15,$TSTNM ;SET TEST #15 AND CLEAR ($ERFLG)
015424 004737 026672 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
015430 012737 016140 001124 MOV #TST16,$LPERR ;SETUP THE ERROR LOOP ADDRESS
015436 013737 002324 001220 MOV RPT,$TIMES ;GET THE ITERATION COUNT
015444 112737 000031 001131 MOVB #25, $ERMAX ;MAX ERRORS ALLOWED FOR TEST
015452 012737 000015 001240 MOV #15,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

306 015460 032777 010000 163466 BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
015466 001406 BEQ .+16 ;BR IF YES
015470 104401 046253 TYPE ,MSGTST ;TYPE 'TEST'
015474 013746 001240 MOV $TESTN,-(SP) ;SAVE $TESTN FOR TYPEOUT
015500 104403 TYPOS ;GO TYPE--OCTAL ASCII
015502 003 .BYTE 3 ;TYPE 3 DIGIT(S)
015503 000 .BYTE 0 ;SUPPRESS LEADING ZEROS

307 015504 005737 001340 TST CLKSTA ;KW11-P CLOCK
308 015510 003002 BGT 1$ ;YES--START TEST
311 015512 000137 016136 JMP EXIT15 ;NO--JUMP TO EXIT15
312 015516 012737 015516 001122 1$: MOV #, $LPADR ;SETUP THE LOOP ADDRESS
313 015524 004037 031110 JSR RO,SRCH00 ;DO A MASSBUS INIT & RECAL
314 015530 000402 BR 2$ ;RETURN HERE IF NO ERROR
326 015532 000137 016136 JMP EXIT15 ;RETURN HERE ON ERROR

```

```

015536 012737 015536 001122 2$: MOV #,$LPADR ;ERROR LOOP ADDRESS
015544 112737 000105 045460 MOVB #SEEK,DPB.A+2 ;SEEK=COMMAND
015552 005037 045466 CLR DPB.A+10 ;USE TRACK 0 & SECTOR 0
015556 013737 002326 045470 MOV FC,DPB.A+12 ;STARTING CYLINDER
015564 012737 016136 001346 MOV #EXIT15,BYPASS ;GO TO EXIT15 IF ERROR
015572 004037 027132 JSR RO,CALL.A ;GO EXECUTE THE COMMAND
015576 012703 001666 MOV #T12,R3 ;PARAMETER POINTER
015602 012737 016136 001222 MCV #EXIT15,$ESCAPE ;:ESCAPE TO EXIT15 ON ERROR
015610 TEST15:
327 015610 012706 001100 MOV #STACK,SP ;SETUP STACK
328 015614 012701 000200 MOV #128.,R1 ;REPEAT "'FC'-'LC'-'FC'" 128 TIMES
329 015620 004737 031274 JSR PC,STRMR ;INIT. THE TIMERS
330 015624 012777 016044 163644 MOV #7$,@PKV ;SETUP VECTOR IN CASE OF OVERFLOW
331 015632 012777 031272 021230 MOV #DORT1,@RMVEC ;SETUP RM VECTOR
332 015640 005077 163640 1$: CLR @PKB ;START COUNTING FROM ZERO
333 015644 013764 002330 000034 MOV LC,RMDC(R4) ;MAXIMUM CYLINDER
334 015652 012764 000105 000000 MOV #SEEK,RMCS1(R4) ;START A SEEK
335 015660 012777 000131 163614 MOV #131,@PKCS ;START THE CLOCK
336 015666 000001 WAIT ;WAIT ON INTERRUPT
337 015670 042777 000101 163604 BIC #101,@PKCS ;STOP THE CLOCK
338 015676 032764 040000 000012 BIT #ERR,RMDS(R4) ;ERR=1?
339 015704 001411 BEQ 2$ ;NO--BRANCH
340 015706 104412 SAVREG ;SAVE R0-R5
015710 012702 045536 MOV #DTADPB,R2 ;DPB POINTER
015714 004737 044310 JSR PC,SVRH70 ;SAVE ALL THE RH/RM REGISTERS
015720 004737 026072 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
015724 104413 RESREG ;RESTORE R0-R5
341 015726 104017 EMT 17
342 015730 005005 2$: CLR R5 ;SET THE UP/DOWN SWITCH TO UP
343 015732 004737 031336 JSR PC,COUNT ;UP THE COUNT
344 015736 005077 163542 CLR @PKB ;START COUNT AT ZERO
345 015742 013764 002326 000034 MOV FC,RMDC(R4) ;BEGINNING CYLINDER
346 015750 012764 000105 000000 MOV #SEEK,RMCS1(R4) ;START A SEEK
347 015756 012777 000131 163516 MOV #131,@PKCS ;START THE CLOCK
348 015764 000001 WAIT ;WAIT ON INTERRUPT
349 015766 042777 000101 163506 BIC #101,@PKCS ;STOP THE CLOCK
350 015774 032764 040000 000012 BIT #ERR,RMDS(R4) ;'ERR'=1?
351 016002 001411 BEQ 3$ ;NO--BRANCH
352 016004 104412 SAVREG ;SAVE R0-R5
016006 012702 045536 MOV #DTADPB,R2 ;DPB POINTER
016012 004737 044310 JSR PC,SVRH70 ;SAVE ALL THE RH/RM REGISTERS
016016 004737 026072 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
016022 104413 RESREG ;RESTORE R0-R5
353 016024 104017 EMT 17
354 016026 012705 177777 3$: MOV #-1,R5 ;SET THE UP/DOWN SWITCH TO DOWN
355 016032 004737 031336 JSR PC,COUNT ;UPDATE THE COUNT
356 016036 005301 DEC R1 ;DONE?
357 016040 003277 BGT 1$ ;NO--BRANCH
358 016042 000420 BR 8$ ;YES--EXIT
359 016044 042777 000101 163430 7$: BIC #101,@PKCS ;STOP THE CLOCK
360 016052 005037 177776 CLR PS ;DROP THE PRIORITY
361 016056 012600 MOV (SP)+,RO ;PC OF WAIT+2
362 016060 005726 TST (SP)+ ;POP THE PS FROM THE STACK
363 016062 104412 SAVREG ;SAVE R0-R5
016064 012702 045536 MOV #DTADPB,R2 ;DPB POINTER
016070 004737 044310 JSR PC,SVRH70 ;SAVE ALL THE RH/RM REGISTERS
016074 004737 026072 JSR PC,CNTCLR ;GO CLEAR CONTROLLER

```

```
016100 104413 RESREG ;RESTORE R0-R5
364 016102 104020 EMT 20
365 016104 004737 026072 8$: JSR PC,CNTCLR ;GO CLEAR CONTROLLER
366 016110 004737 026156 JSR PC,ST.CLK ;INITIALIZE THE CLOCK
367 016114 012777 041740 020746 MOV #ISR,@RMVEC ;RESTORE RH/RM INT. VECTOR
368 016122 004037 031576 JSR R0,TYPTIM ;GO TYPE THE TIMES
016126 001666 T12 ;POINTER
369 016130 004037 031470 JSR R0,SPTYP ;
370 016134 001742 SP12
371 016136 000004 EXIT15: SCOPE ;CALL SCOPE ROUTINE
372
```

1
2
3
4
5
6
7
8
19
20

.SBTTL ADDRESSING TESTS

```

:;////////////////////////////////////
:*THE ADDRESSING TESTS ENSURES PROPER OPERATION OF THE TRACK
:*AND SECTOR ADDRESS CIRCUITRY. BOTH ADDRESSING TESTS WILL
:*BE PERFORMED ON CYLINDER FC.
:;////////////////////////////////////

```

```

:;*****
:*TEST 16 SECTOR ADDRESSING TEST
:*THIS TEST WRITES DATA INTO ALL SECTORS OF TRACK 'FT'. THE
:*DATA WILL BE 256 WORDS OF THE SECTOR ADDRESS OF THE SECTOR
:*BEING WRITTEN. A WRITE CHECK IS PERFORMED, THE BUFFER IS
:*CLEARED (TO 177400) AND THE DATA IS READ AND COMPARED. THEN
:*SECTOR 0 IS REWRITTEN AND SECTORS 0 - 31 ARE WRITE CHECKED.
:*THEN SECTOR 1 IS REWRITTEN AND SECTORS 0 - 31 ARE WRITE CHECKED.
:*THIS REWRITE AND WRITE CHECK PROCEDURE IS CONTINUED UP THROUGH
:*REWRITE SECTOR 31 AND WRITE CHECK SECTORS 0-31.
:;*****

```

TST16:

016140	016140	000240			NOP		
016142	033737	001562	001330		BIT	BITS+<16*2>,TSTNMS	:DO THIS TEST?
016150	001002				BNE	:+6	:BR IF YES
016152	000137	016562			JMP	TST17	:NO--JUMP TO TEST17
016156	012737	000016	001116		MOV	#16,\$TSTNM	:SET TEST #16 AND CLEAR (\$ERFLG)
016164	004737	026672			JSR	PC,LODPRM	:LOAD THE PARMETERS FOR THE TEST
016170	012737	016260	001124		MOV	#TEST16,\$LPERR	:SETUP THE LOOP ON ERROR ADDRESS
016176	013737	002324	001220		MOV	RPT,\$TIMES	:GET THE ITERATION COUNT
016204	113737	001462	001131		MOV	ERR.CT,\$ERMAX	:MAX ERRORS ALLOWED FOR TEST
016212	012737	000016	001240		MOV	#16,\$TESTN	:;SET TEST NUMBER IN APT MAIL BOX
21 016220	032777	010000	162726		BIT	#SW12,@SWR	:INHIBIT TYPING TEST NUMBER ?
016226	001406				BEQ	:+16	:BR IF YES
016230	104401	046253			TYPE	,MSGTST	:TYPE 'TEST'
016234	013746	001240			MOV	\$TESTN,-(SP)	:;SAVE \$TESTN FOR TYPEOUT
016240	104403				TYPOS		:;GO TYPE--OCTAL ASCII
016242	003				.BYTE	3	:;TYPE 3 DIGIT(S)
016243	000				.BYTE	0	:;SUPPRESS LEADING ZEROS
26 016244	012737	016560	001346		MOV	#EXIT16,BYPASS	
016252	012737	016260	001122		MOV	#TEST16,\$LPADR	:SETUP THE LOOP ADDRESS
016260							
27 016260	012706	001100			MOV	#STACK,SP	:SET THE STACK POINTER
28 016264	004737	032256			JSR	PC,FILBUF	:FILL THE BUFFER WITH SECTOR ADDRESSES
29 016270	013737	002326	045550		MOV	FC,DTADPB+12	:CYLINDER
30 016276	113737	002334	045547		MOV	FT,DTADPB+11	:TRACK
31 016304	105037	045546			CLRB	DTADPB+10	:SECTOR
32 016310	013737	001450	045542		MOV	TRCKWC,DTADPB+4	:WORD COUNT
33 016316	012737	052776	045544		MOV	#BUFFER,DTADPB+6	:BUFFER ADDRESS
34 016324	012737	016324	001124		MOV	#, \$LPERR	:SETUP THE ERROR LOOP ADDRESS
016332	012706	001100			MOV	#STACK,SP	:LOAD THE STACK POINTER
35 016336	012737	000161	045540		MOV	#WRITE,DTADPB+2	:COMMAND=WRITE DATA
36 016344	004037	027670			JSR	RO,DRVCAL	:START A DATA TRANSFER
37 016350	012737	000151	045540		MOV	#WRCKD,DTADPB+2	:COMMAND=WRITE CHECK DATA
38 016356	012737	016356	001124		MOV	#, \$LPERR	:SETUP THE ERROR LOOP ADDRESS

```

39 016364 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
40 016370 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
40 016374 012737 016374 001124  MOV      #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
41 016402 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
41 016406 004037 032314      JSR      RO,CLRBUF      ;CLEAR BUFFER
42 016412 012737 000171 045540  MOV      #READ,DTADPB+2 ;COMMAND = READ
43 016420 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
44 016424 004037 032362      JSR      RO,CKSCTR      ;CHECK THE SECTOR DATA READ
45 016430 012700 052776      MOV      #BUFFER,RO     ;BUFFER ADDRESS
46 016434 005001      CLR      R1             ;FIRST SECTOR
47 016436 012737 016436 001124  MOV      #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
48 016444 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
48 016450 012737 000161 045540 1$: MOV      #WRITE,DTADPB+2 ;COMMAND=WRITE DATA
49 016456 012737 177400 045542  MOV      #SCTRWC,DTADPB+4 ;WORD COUNT
50 016464 010037 045544      MOV      RO,DTADPB+6    ;BUFFER ADDRESS
51 016470 110137 045546      MOV      R1,DTADPB+10   ;SECTOR
52 016474 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
53 016500 012737 016500 001124  MOV      #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
54 016506 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
54 016512 012737 000151 045540  MOV      #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK DATA
55 016520 013737 001450 045542  MOV      TRCKWC,DTADPB+4 ;WORD COUNT
56 016526 012737 052776 045544  MOV      #BUFFER,DTADPB+6 ;BUFFER ADDRESS
57 016534 105037 045546      CLR      DTADPB+10     ;SECTOR
58 016540 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
59 016544 062700 001000      ADD      #512.,RO      ;MOVE TO NEXT SECTOR
60 016550 005201      INC      R1
61 016552 023701 002456      CMP      PRMLMT+24,R1  ;DONE?
62 016556 103334      BHS     1$             ;NO--BRANCH
63 016560 000004      EXIT16: SCOPE         ;CALL SCOPE ROUTINE
64
66
67
77

```

```

*****
*TEST 17 TRACK ADDRESSING TEST
*THIS TEST WILL WRITE DATA IN THE FORM OF TRACK ADDRESSES
*IN CYLINDER 'FC' SECTOR 'FS' OF EVERY TRACK WITH EACH TRACK
*GETTING ITS OWN TRACK ADDRESS.
*A WRITE CHECK IS THEN PERFORMED ON EACH TRACK TO ENSURE
*THE DATA IS VALID. THEN TRACK 0 IS REWRITTEN AND TRACK 1
*THROUGH LAST TRACK IS WRITE CHECKED. THEN TRACK 1 IS
*REWRITTEN AND TRACK 2 THROUGH LAST TRACK IS WRITE CHECKED.
*THIS PROCEDURE IS CONTINUED UP THROUGH REWRITING NEXT TO LAST
*TRACK AND WRITE CHECKING LAST TRACK.
*****
TST17:

```

```

016562      NOP
016562 000240      BIT      BITS+<17*2>,TSTNMS ;DO THIS TEST?
016564 033737 001564 001330  BNE     +6             ;BR IF YES
016572 001002      JMP     TST20         ;NO--JUMP TO TEST20
016574 000137 017226

016600 012737 000017 001116  MOV      #17,$TSTNM     ;SET TEST #17 AND CLEAR ($ERFLG)
016606 004737 026672      JSR      PC,LODPRM     ;LOAD THE PARMETERS FOR THE TEST
016612 012737 016702 001124  MOV      #TEST17,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
016620 013737 002324 001220  MOV      RPT,$TIMES    ;GET THE ITERATION COUNT
016626 113737 001462 001131  MOV      ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
016634 012737 000017 001240  MOV      #17,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
78 016642 032777 010000 162304  BIT      #SW12,@SWR    ;INHIBIT TYPING TEST NUMBER ?

```

```

016650 001406          BEQ      .+16          ;BR IF YES
016652 104401 046253  TYPE      .MSGTST        ;TYPE 'TEST'
016656 013746 001240  MOV      $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
016662 104403          TYPOS     ;GO TYPE--OCTAL ASCII
016664      003        .BYTE      3          ;TYPE 3 DIGIT(S)
016665      000        .BYTE      0          ;SUPPRESS LEADING ZEROS

83 016666 012737 017224 001346  MOV      #EXIT17,BYPASS
016674 012737 016702 001122  MOV      #TEST17,$LPADR ;SETUP THE LOOP ADDRESS
016702          TEST17:
84 016702 012706 001100  MOV      #STACK,SP      ;SET THE STACK POINTER
85 016706 004737 032256  JSR      PC,FILBUF      ;FILL THE BUFFER WITH TRACK ADDRESS
86 016712 012737 000161 045540  MOV      #WRITE,DTADPB+2 ;COMMAND=WRITE DATA
87 016720 013737 002326 045550  MOV      FC,DTADPB+12   ;CYLINDER
88 016726 113737 002342 045546  MOVVB   FS,DTADPB+10    ;SECTOR
89 016734 012737 177400 045542  MOV      #SCTRWC,DTADPB+4 ;WORD COUNT
90 016742 012737 052776 045544  MOV      #BUFFER,DTADPB+6 ;BUFFER ADDRESS
91 016750 005000          CLR      R0            ;TRACK=0
92 016752 012737 016752 001124  MOV      #.,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
016760 012706 001100  MOV      #STACK,SP     ;LOAD THE STACK POINTER

93
94 016764 110037 045547          1$:  MOVVB   R0,DTADPB+11   ;TRACK ADDRESS
95 016770 004037 027670          JSR      R0,DRVCL      ;START A DATA TRANSFER
96 016774 062737 001000 045544  ADD      #256.*2.,DTADPB+6 ;UPDATE BUFFER ADDRESS
97 017002 005200          INC      R0            ;UPDATE TRACK NUMBER
98 017004 023700 001372          CMP      LSTRK,R0      ;OUT OF TRACKS?
99 017010 002365          BGE      1$           ;NO--BRANCH
100 017012 012737 052776 045544  MOV      #BUFFER,DTADPB+6 ;BUFFER ADDRESS
101 017020 005000          CLR      R0
102 017022 012737 017022 001124  MOV      #.,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
017030 012706 001100  MOV      #STACK,SP     ;LOAD THE STACK POINTER

103
104 017034 012737 000151 045540  MOV      #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK
105 017042 110037 045547          2$:  MOVVB   R0,DTADPB+11   ;TRACK ADDRESS
106 017046 004037 027670          JSR      R0,DRVCL      ;START A DATA TRANSFER
107 017052 062737 001000 045544  ADD      #256.*2.,DTADPB+6 ;UPDATE BUFFER ADDRESS
108 017060 005200          INC      R0            ;UPDATE TRACK NUMBER
109 017062 023700 001372          CMP      LSTRK,R0      ;OUT OF TRACKS?
110 017066 002365          BGE      2$           ;NO--BRANCH
111 017070 005000          CLR      R0            ;FIRST TRACK ADDRESS
112
113 017072 110037 045547          3$:  MOVVB   R0,DTADPB+11   ;TRACK
114 017076 010001          MOV      R0,R1         ;FORM BUFFER ADDRESS
115 017100 012737 052776 045544  MOV      #BUFFER,DTADPB+6 ;BUFFER ADDRESS
116 017106 005301          4$:  DEC      R1
117 017110 002411          BLT      5$
118 017112 062737 001000 045544  ADD      #256.*2.,DTADPB+6
119 017120 000772          BR      4$
120 017122 012737 017122 001124  MOV      #.,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
017130 012706 001100  MOV      #STACK,SP     ;LOAD THE STACK POINTER
121 017134 012737 000161 045540  5$:  MOV      #WRITE,DTADPB+2 ;COMMAND=WRITE DATA
122 017142 004037 027670          JSR      R0,DRVCL      ;START A DATA TRANSFER
123
124 017146 062737 001000 045544  6$:  ADD      #256.*2.,DTADPB+6 ;UPDATE BUFFER ADDRESS
125 017154 105237 045547          INCB   DTADPB+11      ;MOVE TO NEXT TRACK
126 017160 012737 017160 001124  MOV      #.,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
017166 012706 001100  MOV      #STACK,SP     ;LOAD THE STACK POINTER

```

127 017172 012737 000151 045540
 128 017200 004037 027670
 129 017204 123737 001372 045547
 130 017212 003355
 131 017214 005200
 132 017216 023700 001372
 133 017222 003323
 134 017224 000004
 135
 155
 156

```

MOV #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK DATA
JSR R0,DRVCAL ;START A DATA TRANSFER
CMPB LSTRK,DTADPB+11 ;OUT OF TRACKS?
BGT 6$ ;NO--BRANCH
INC R0 ;NEXT TRACK TO WRITE
CMP LSTRK,R0 ;OUT OF TRACKS?
BGT 3$ ;NO--BRANCH
EXIT17: SCOPE ;CALL SCOPE ROUTINE
    
```

```

*****
*TEST 20 DATA TEST
*THIS TEST PERFORMS DATA STORAGE AND RETRIEVAL ON CYLINDERS
*FC THROUGH LC BY THE INCREMENT IC USING THE DATA PATTERNS
*SPECIFIED. THE FOLLOWING SEQUENCE OCCURS FOR EACH CYLINDER:
* (1. SET NT TO FT THEN REPEAT 2-4 UNTIL NT > LT
* (2. WRITE THEN WRITE CHECK FS THROUGH LS OF TRACK NT
* (3. READ THEN SOFTWARE COMPARE FS THROUGH LS OF TRACK NT
* (4. INCREMENT NT BY IT
* (5. REPEAT STEPS 1-4 FOR EACH DATA PATTERN
* (6. REPEAT STEPS 1-5 FOR FC THROUGH LC ADVANCING BY IC
*
*IF A WRITE CHECK ERROR OCCURS THE ERROR IS REPORTED AND
*THE TRACK IN ERROR IS REWRITTEN AND CHECKED. THIS CHECK IS
*ACCOMPLISHED BY PERFORMING TWO(2) SUCCESSIVE ERROR FREE
*WRITE CHECKS. IF THE CHECK FAILS THE ERROR IS REPORTED AS
*FATAL AND NO READ OCCURS.
*FS DEFAULTS TO 1 AND LS DEFAULTS TO 0
*PAT DEFAULTS TO 17777 (ALL POSSIBLE PATTERNS)
*****
    
```

017226
 017226 000240
 017230 033737 001526 001332
 017236 001002
 017240 000137 020006

 017244 012737 000020 001116
 017252 004737 026672
 017256 012737 017454 001124
 017264 013737 002324 001220
 017272 113737 001462 001131
 017300 012737 000020 001240
 157
 017306 032777 010000 161640
 017314 001406
 017316 104401 046253
 017322 013746 001240
 017326 104403
 017330 003
 017331 000

 158
 159
 160
 161 017332 012737 017332 001122
 162 017340 005000
 163 017342 005004

```

TST20:
NOP
BIT BITS+<20*2-40>,TSTNMS+2 ;DO THIS TEST ?
BNE .+6 ;BR IF YES
JMP TST21 ;NO--JUMP TO TEST21
  

MOV #20,$STNM ;SET TEST #20 AND CLEAR ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TEST20,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV RPT,$TIMES ;GET THE ITERATION COUNT
MOVB ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #20,$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--LOCAL ASCII
.BYTE 3 ;TYPE 3 DIGIT(S)
.BYTE 0 ;;SUPPRESS LEADING ZEROS
  

;SET UP THE TRACK WORD COUNT FOR DATA TRANSFER
MOV #,$LPADR ;SETUP THE LOOP ADDRESS
CLR R0 ;CLEAR SWITCH
CLR R4 ;FORM WORD COUNT IN R4
    
```



```

164 017344 013701 002344      MOV    LS,R1      ;LOAD LAST SECTOR
165 017350 163701 002342      SUB    FS,R1      ;COMPARE WITH FIRST SECTOR
166 017354 002004              BGE    1$        ;SKIP NEXT IF FS < OR = LS
167
168 017356 063701 002456      ADD    PRMLMT+24,R1 ;ADD MAXIMUM SECTOR ADDRESS TO
169 017362 005201              INC    R1         ;MAKE THE DIFFERENCE POSITIVE
170 017364 005100              COM    R0         ;SET SWITCH FOR FS > LS
171
172 017366 062704 000400      1$:  ADD    #256.,R4   ;WORDS/SECTOR
173 017372 005301              DEC    R1         ;LS - FS SECTORS MINUS ONE
174 017374 002374              BGE    1$        ;INCR. WORD COUNT IF NO. SECTORS STILL +
175 017376 005404              NEG    R4         ;FORM NEGATIVE WORD COUNT
176 017400 010405              MOV    R4,R5     ;COPY NORMAL WORD COUNT INTO SMALL WC
177 017402 005700              TST    R0        ;FS > LS SWITCH SET?
178 017404 001412              BEQ    3$        ;NO, SKIP NEXT
179
180 017406 005005              CLR    R5        ;FORM WORD COUNT FOR FS > LS
181 017410 013701 002456      MOV    PRMLMT+24,R1 ;MAX SECTOR ADDRESS
182 017414 163701 002342      SUB    FS,R1     ;SUBTRACT THE FIRST SECTOR
183 017420 062705 000400      2$:  ADD    #256.,R5   ;WORDS/SECTOR
184 017424 005301              DEC    R1         ;NO SECTORS MINUS ONE
185 017426 002374              BGE    2$        ;INCR. WORD COUNT IF NO. SECTORS STILL +
186 017430 005405              NEG    R5        ;FORM NEGATIVE WORD COUNT FOR THIS CASE
187
188                          ;SET UP FOR DATA TRANSFERS AND PATTERN VARIATIONS
189
190 017432 113737 002342 045546 3$:  MOV    #5,DTADPB+10 ;SECTOR
191 017440 012737 052776 045544      MOV    #BUFFER,DTADPB+6 ;DATA BUFFER
219 017446 012737 020004 001346      MOV    #EXIT20,BYPASS
      017454              TEST20:
      017454 012706 001100      MOV    #STACK,SP  ;LOAD THE STACK POINTER
      017460 005037 001432      CLR    WCEFLG    ;CLEAR THE WRITE CHECK ERROR FLAG
      017464 013701 002326      MOV    FC,R1     ;PICKUP FIRST CYLINDER
      017470 000407              BR     2$        ;SKIP PATTERN SET-UP FIRST TIME THRU

      017472 005720              1$:  TST    (R0)+     ;MOVE TO NEXT DATA PATTERN
      017474 022700 000040      CMP    #16.*2.,R0 ;OUT OF PATTERNS?
      017500 003004              BGT    3$        ;NO, STAY ON THIS CYLINDER UNTIL DONE
      017502 004037 032202      JSR    R0,INCCYL ;MOVE TO NEXT CYLINDER
      017506 000536              BR     EXIT20   ;OUT OF CYLINDERS

      ;DO NEXT CYLINDER

      017510 005000              2$:  CLR    R0        ;START WITH PATTERN 0
      017512 036657 001526 002346 3$:  BIT    BITS(R0),PAT ;THIS PATTERN SELECTED?
      017520 001764              BEQ    1$        ;NO, GO BACK AND GET ONE THAT WAS
      017522 013702 002334              MOV    FT,R2     ;FIRST TRACK
      017526 010137 045550              MOV    R1,DTADPB+12 ;LOAD THE CYLINDER
      017532 110237 045547              4$:  MOV    R2,DTADPB+11 ;LOAD THE TRACK

      017536 020127 001466              CMP    R1,#822.  ;CHECK FOR LAST DISK CYLINDER
      017542 001003              BNE    10$       ;SKIP LAST TRACK CHECK IF NOT
      017544 020237 001372              CMP    R2,LSTRK  ;LAST DISK TRACK ?
      017550 001515              BEQ    EXIT20   ;DON'T TEST LAST TRACK AS IT HAS BAD
220 017552              10$:          ;BLOCK INFORMATION STORED ON IT
221
222 017552 010437 045542              MOV    R4,DTADPB+4 ;LOAD THE WORD COUNT

```

```
223 017556 023701 002330      CMP      LC,R1          ;LAST DISK CYLINDER TO TEST ?
224 017562 003005              BGT      5$            ;NO, SKIP TRACK CHECK
225 017564 023702 001372      CMP      LSTRK,R2       ;LAST DISK TRACK TO TEST ?
226 017570 002002              BGE      5$            ;NO, SKIP NEXT
227 017572 010537 045542      MOV      R5,DTADPB+4    ;SHORT WORD COUNT
228 017576 017703 161352      5$: MOV      @SWR,R3      ;INHIBIT WRITE AND
229 017602 005103              COM      R3            ;WRITE CHECK?
230 017604 032703 000030      BIT      #SW04'SW03,R3
231 017610 001436              BEQ      7$            ;YES--BRANCH
232 017612 004737 032732      JSR      PC,SETBJF      ;MOVE DATA PATTERN INTO THE BUFFER
233 017616 032777 000020 161330  BIT      #SW04,@SWR     ;INHIBIT WRITE?
234 017624 001012              BNE      6$            ;YES, DO NEXT PORTION OF TESTING
235 017626 012737 017626 001124  MOV      #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
017634 012706 001100      MOV      #STACK,SP     ;LOAD THE STACK POINTER
236 017640 012737 000161 045540  MOV      #WRITE,DTADPB+2 ;COMMAND=WRITE DATA
237
238
239
240 017646 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
241 017652 032777 000010 161274  6$: BIT      #SW03,@SWR   ;INHIBIT WRITE CHECK?
242 017660 001012              BNE      7$            ;YES--BRANCH
243 017662 012737 017662 001124  MOV      #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
017670 012706 001100      MOV      #STACK,SP     ;LOAD THE STACK POINTER
244 017674 012737 000151 045540  MOV      #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK DATA
245 017702 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
246 017706 005737 001432      7$: TST      WCEFLG      ;WRITE CHECK ERROR FLAG SET?
247 017712 001404              BEQ      8$            ;NO--BRANCH
248 017714 032777 000001 161232  BIT      #SW00,@SWR     ;PERFORM READ AFTER FATAL 'WCE'?
249 017722 001424              BEQ      9$            ;NO--BRANCH
250 017724 032777 000004 161222  8$: BIT      #SW02,@SWR   ;INHIBIT READ DATA AND SOFTWARE COMPARE?
251 017732 001020              BNE      9$            ;YES--BRANCH
252 017734 012737 017734 001124  MOV      #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
017742 012706 001100      MOV      #STACK,SP     ;LOAD THE STACK POINTER
253 017746 012737 000171 045540  MOV      #READ,DTADPB+2 ;COMMAND=READ
254 017754 004037 027670      JSR      RO,DRVCAL      ;START A DATA TRANSFER
255 017760 032777 000002 161166  BIT      #SW01,@SWR     ;COMPARE THE DATA?
256 017766 001002              BNE      9$            ;NO--BRANCH
257 017770 004737 033022      JSR      PC,DATCMP      ;YES--DO IT
258 017774 004037 032152      9$: JSR      RO,INCTRK    ;MOVE TO NEXT TRACK
259 020000 000634              BR       1$            ;OUT OF TRACKS GO TO NEXT PATTERN
260 020002 000653              BR       4$            ;LOOP
261 020004 000004      EXIT20: SCOPE          ;CALL SCOPE ROUTINE
262
287
288
```

```
::*****
:*TEST 21      RANDOM ADDRESS AND RANDOM PATTERN TEST
:*STARTING AT 'FC' AND GOING SEQUENTIALLY TO 'LC' THE DISK
:*PACK IS WRITTEN WITH A RANDOM PATTERN. THE FIRST TWO WORDS
:*OF EACH SECTOR WILL BE THE BASE OF THE RANDOM GENERATOR
:*FOR THAT SECTOR.

:*THE TEST THEN PERFORMS THE FOLLOWING SEQUENCE 'R' TIMES
:*'R' DEFAULTS TO 1000.
:*
:*+1)  GENERATE A RANDOM SECTOR ADDRESS
:*+2)  WRITE A RANDOM PATTERN AT THE ADDRESS
:*     GENERATED IN 1.
```

```

:*3) GENERATE A NEW RANDOM SECTOR ADDRESS
:*4) READ THE SECTOR AT THE ADDRESS
:*   GENERATED IN 3.
:*5) DO A SOFTWARE CHECK OF THE DATA READ IN 4 AGAINST
:*   THE ORIGINAL RANDOM PACK DATA.
:*6) DO A WRITE CHECK OF THE DATA WRITTEN IN 2
:*7) GENERATE A NEW RANDOM SECTOR ADDRESS
:*8) READ THE SECTOR AT THE ADDRESS
:*   GENERATED IN 7.
:*9) DO A SOFTWARE CHECK OF THE DATA READ IN 8
:*10) DO A WRITE CHECK OF THE DATA WRITTEN IN 2
:*****

```

TST21:

```

020006
020006 000240
020010 033737 001530 001332
020016 001002
020020 000137 020624

020024 012737 000021 001116
020032 004737 026672
020036 012737 020302 001124
020044 013737 002324 001220
020052 113737 001462 001131
020060 012737 000021 001240
289
020066 032777 010000 161060
020074 001406
020076 104401 046253
020102 013746 001240
020106 104403
020110 003
020111 000

290 020112 012737 020112 001122
293 020120 012737 020622 001346
294 020126 012737 176543 025646
295 020134 012737 123456 025650
296 020142 013737 002326 045550
297 020150 013737 001450 045542
298 020156 012737 052776 045544
299 020164 012737 000161 045540
300 020172 032737 100000 001314
301 020200 001027
302 020202 004037 033340
303 020206 005037 045546 1$:
304 020212 012737 020212 001124
020220 012706 001100
305 020224 2$:
020224 004037 027670
306 020230 105237 045547
307 020234 123737 001372 045547
308 020242 002370
309 020244 005237 045550
310 020250 023737 045550
311 020256 002553
312
313

```

```

NOP
BIT BITS+<21*2-40>,TSTNMS+2 ;DO THIS TEST ?
BNE +6 ;BR IF YES
JMP TST22 ;NO--JUMP TO TEST22

MOV #21,$TSTNM ;SET TEST #21 AND CLEAR ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
MOV #TST21,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV RPT,$TIMES ;GET THE ITERATION COUNT
MOVB ERR.CT,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #21,$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

MOV #,$LPADR ;SETUP THE LOOP ADDRESS
MOV #EXIT21,BYPASS
MOV #176543,$HINUM ;PRIME THE RANDOM NUMBER GENERATOR
MOV #123456,$LONUM
MOV FC,DTADPB+12 ;CYLINDER
MOV TRCKWC,DTADPB+4 ;WORD COUNT FOR 32/30 SECTORS (FULL TRACK)
MOV #BUFFER,DTADPB+6 ;BUFFER ADDRESS
MOV #WRITE,DTADPB+2 ;COMMAND
BIT #SW15,C.SWR ;WRITE THE DISK PACK BEFORE TESTING?
BNE 3$ ;NO--BEGIN TESTING
JSR RO,FILRAN ;FILL DATA BUFFER WITH RANDOM DATA
CLR DTADPB+10 ;SECTOR AND TRACK
MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #STACK,SP ;LOAD THE STACK POINTER

JSR RO,DRVCAL ;START A DATA TRANSFER
INCB DTADPB+11 ;NEXT TRACK
CMPB LSTRK,DTADPB+11 ;TIME FOR NEXT CYLINDER ?
BGE 2$ ;NO--DO NEXT TRACK ON THIS CYL.
INC DTADPB+12 ;INCR CYLINDER ADDRESS
CMP LC,DTADPB+12 ;OUT OF CYLINDERS?
BGE 1$ ;NO--CONTINUE SEQUENTIAL RANDOM WRITE

```

```

314 020260 012737 177400 045542 38:  MOV #SCTRWC,DTADPB+4 ;WORD COUNT
319 020266 012737 020302 001122  MOV #TEST21,$LPADR
    020274 012737 020302 001124  MOV #TEST21,$LPERR
    020302                                TEST21:
320 020302 012706 001100  MOV #STACK,SP ;SET STACK POINTER
321 020306 004037 033614  JSR RO,RANADR ;GENERATE A RANDOM ADDRESS
322 020312 013737 045546 001436  MOV DTADPB+10,$VADR ;SAVE THE TRACK/SECTOR
323 020320 013737 045550 001440  MOV DTADPB+12,$VADR+2 ;SAVE THE CYLINDER
324 020326 012737 000161 045540  MOV #WRITE,DTADPB+2 ;COMMAND=WRITE DATA
325 020334 012701 052776  MOV #BUFFER,R1 ;WRITE BUFFER ADDRESS FOR 'RANPAT'
326 020340 010137 045544  MOV R1,DTADPB+6 ;INTO THE DATA PARAMETER BLOCK
327 020344 004037 033560  JSR RO,RANPAT ;GENERATE RANDOM 256 WORD PATTERN
328                                ;AND PUT INTO THE WRITE BUFFER
329 020350 012737 020350 001124  MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
    020356 012706 001100  MOV #STACK,SP ;LOAD THE STACK POINTER
330 020362 004037 027670  JSR RO,DRVCAL ;START A DATA TRANSFER
331
332 020366 004037 033614  JSR RO,RANADR ;GENERATE A NEW RANDOM ADDRESS
333 020372 012737 000171 045540  MOV #READ,DTADPB+2 ;COMMAND=READ DATA
334 020400 012737 053776 045544  MOV #BUFFER+512.,DTADPB+6 ;READ BUFFER ADDRESS
335 020406 012737 020406 001124  MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
    020414 012706 001100  MOV #STACK,SP ;LOAD THE STACK POINTER
336 020420 004037 027670  JSR RO,DRVCAL ;START A DATA TRANSFER
337 020424 005737 001464  TST BASFLG ;IF BAD SECTOR ENCOUNTERED,SKIP NEXT CALL
338 020430 100402  BMI .+6 ;DON'T COMPARE DATA
339 020432 004037 033362  JSR RO,RANCK ;SOFTWARE CHECK THE DATA
340
341 020436 013737 001436 045546  MOV $VADR,DTADPB+10 ;GET ADDRESS OF WHERE THE LAST
342 020444 013737 001440 045550  MOV $VADR+2,DTADPB+12 ;WRITE WAS PERFORMED
343 020452 012737 000151 045540  MOV #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK DATA
344 020460 012737 052776 045544  MOV #BUFFER,DTADPB+6 ;DATA BUFFER ADDRESS FOR HARDWARE
345                                ;CHECK OF THE DATA
346 020466 012737 020466 001124  MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
    020474 012706 001100  MOV #STACK,SP ;LOAD THE STACK POINTER
347 020500 004037 027670  JSR RO,DRVCAL ;START A DATA TRANSFER
348
349 020504 004037 033614  JSR RO,RANADR ;GENERATE A NEW RANDOM ADDRESS
350 020510 012737 000171 045540  MOV #READ,DTADPB+2 ;COMMAND=READ
351 020516 012737 053776 045544  MOV #BUFFER+512.,DTADPB+6 ;DATA BUFFER ADDRESS
352 020524 012737 020524 001124  MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
    020532 012706 001100  MOV #STACK,SP ;LOAD THE STACK POINTER
353 020536 004037 027670  JSR RO,DRVCAL ;START A DATA TRANSFER
354 020542 005737 001464  TST BASFLG ;ENCOUNTER BAD SECTOR ?
355 020546 100402  BMI .+6 ;DON'T COMPARE DATA IF SO
356 020550 004037 033362  JSR RO,RANCK ;SOFTWARE CHECK THE DATA
357
358 020554 013737 001436 045546  MOV $VADR,DTADPB+10 ;GET DISK ADDRESS OF THE
359 020562 013737 001440 045550  MOV $VADR+2,DTADPB+12 ;LAST WRITE
360 020570 012737 000151 045540  MOV #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK DATA
361 020576 012737 052776 045544  MOV #BUFFER,DTADPB+6 ;DATA BUFFER ADDRESS
362 020604 012737 020604 001124  MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
    020612 012706 001100  MOV #STACK,SP ;LOAD THE STACK POINTER
363 020616 004037 027670  JSR RO,DRVCAL ;START A DATA TRANSFER
364 020622 000004  EXIT21: SCOPE ;CALL SCOPE ROUTINE
365
372
373

```

.....

```

; *TEST 22 ACCESS TIME ADJUSTMENT TEST
; *THIS TEST PERFORMS SEEKS BETWEEN CYLINDERS 0 & 255 TO ALLOW THE
; *OPERATOR TO ADJUST THE ACCESS TIME ON AN RM05/3/2 USING THE
; *DDU. THE PROGRAM STALLS APPROXIMATELY 5 SECONDS BETWEEN SEEKS
; *SO THAT THE ACCESS TIME INDICATORS ON THE DDU MAY BE OBSERVED.
; *****

```

```

TST22:
020624      020624 000240      NOP
020626      033737 001532 001332      BIT      BITS+<22*2-40>,TSTNMS+2 ;DO THIS TEST ?
020634      001002      BNE      .+6 ;BR IF YES
020636      000137 021020      JMP      $EOP ;NO--GO TO THE END OF THE PROGRAM

020642      012737 000022 001116      MOV      #22,$TSTNM ;SET TEST #22 AND CLEAR ($ERFLG)
020650      004737 026672      JSR      PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
020654      012737 020736 001124      MOV      #TEST22,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
020662      013737 002324 001220      MOV      RPT,$TIMES ;GET THE ITERATION COUNT
020670      112737 000144 001131      MOV      #100,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
020676      012737 000022 001240      MOV      #22,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

374 020704      032777 010000 160242      BIT      #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
020712      001406      BEQ      .+16 ;BR IF YES
020714      104401 046253      TYPE     ,MSGTST ;TYPE 'TEST'
020720      013746 001240      MOV      $TESTN,-(SP) ;SAVE $TESTN FOR TYPEOUT
020724      104403      TYPOS    ;GO TYPE--OCTAL ASCII
020726      003 ;TYPE 3 DIGIT(S)
020727      000 ;SUPPRESS LEADING ZEROS

378 020730      012737 020736 001122      MOV      #TEST22,$LPADR ;SETUP THE LOOP ADDRESS
020736      TEST22:
379 020736      012706 001100      MOV      #STACK.SP ;SETUP THE STACK POINTER
380 020742      013737 002330 045470      MOV      LC,DPB.A+12 ;ENDING CYLINDER
381 020750      112737 000105 045460      MOV      #SEEK,DPB.A+2 ;SEEK=COMMAND
382 020756      004037 027132      JSR      RO,CALL.A ;GO EXECUTE THE COMMAND
383 020762      004037 030664      JSR      RO,STALL ;STALL
384 020766      001456      .WORD    STALL3 ;ADDRESS OF STALL VALUE
385 020770      013737 002326 045470      MOV      FC,DPB.A+12 ;STARTING CYLINDER
386 020776      112737 000105 045460      MOV      #SEEK,DPB.A+2 ;SEEK=COMMAND
387 021004      004037 027132      JSR      RO,CALL.A ;GO EXECUTE THE COMMAND
388 021010      004037 030664      JSR      RO,STALL ;STALL
389 021014      001456      .WORD    STALL3 ;ADDRESS OF STALL VALUE
390 021016      000004      EXIT22: SCOPE ;CALL SCOPE ROUTINE
391

```

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 RESTART
  
```

```

021020                                $EOP:
021020 104401 021026                    TYPE      ,65$      ;;TYPE ASCIZ STRING
021024 000407                            BR        64$      ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <CRLF><LF>/END OF PASS/
64$:
021044                                TST      DRVSEL    ;ANY DRIVES SELECTED?
021044 005737 001326                    BEQ      1$      ;NO--BRANCH
021050 001434                            TYPE      ,67$    ;;TYPE ASCIZ STRING
021052 104401 021060                    BR        66$    ;;GET OVER THE ASCIZ
021056 000405                            ;;67$: .ASCIZ / ON DRIVE/
021072                                66$:
021072 013746 001350                    MOV      CHKDRV,-(SP) ;;SAVE $CHKV FOR TYPEOUT
021076 104403                            TYP0S    ;GO TYPE--OCTAL ASCII
021100      002                            .BYTE   2        ;;TYPE 2 DIGIT(S)
021101      000                            .BYTE   ;        ;;SUPPRESS LEADING ZEROS
021102 104401 021110                    TYPE     ,69$    ;;TYPE ASCIZ STRING
021106 000412                            BR        68$    ;;GET OVER THE ASCIZ
;;69$: .ASCIZ / ERRORS DETECTED=/
68$:
021134                                MOV      $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
021134 013746 001126                    TYPOC    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
021140 104402                            1$: CLR      $ERTTL  ;;ZERO ERROR TOTAL
021142 005037 001126                    CLR      $TSTNM   ;;ZERO THE TEST NUMBER
021146 005037 001116                    CLR      $TIMES   ;;ZERO THE NUMBER OF ITERATIONS
021152 005037 001220                    INC      $PASS    ;;INCREMENT THE PASS NUMBER
021156 005237 001242                    BIC     #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
021162 042737 100000 001242            DEC      (PC)+    ;;LOOP?
021170 005327                            $EOPCT: .WORD  8.
021172 000010                            BGT     $DOAGN   ;;YES
021174 003027                            MOV     (PC)+,@(PC)+ ;;RESTORE COUNTER
021176 012737                            $ENDCT: .WORD  8.
021200 000010                            TYPE     ,65$    ;;TYPE ASCIZ STRING
021202 021172                            BR      64$     ;;GET OVER THE ASCIZ
021204 104401 021212                    ;;65$: .ASCIZ <CRLF>/END OF TEST/
021210 000407                            64$:
021230                                $GET42: TYPE     , $ENULL  ;;TYPE NULL CHARACTER
021230 104401 021260                    MOV     @#42,R0  ;;GET MONITOR ADDRESS
021234 013700 000042                    BEQ     $DOAGN   ;;BRANCH IF NO MONITOR
021240 001405                            RESET    ;;CLEAR THE WORLD
021242 000005                            $ENDAD: JSR    PC,(R0) ;;GO TO MONITOR
021244 004710                            NOP     ;;SAVE ROOM
021246 000240                            NOP     ;;FOR
021250 000240                            NOP     ;;ACT11
021252 000240                            $DOAGN:
021254                                JMP     @(PC)+   ;;RETURN
021254 000137                            $RTNAD: .WORD  RESTART
021256 006340                            $ENULL: .BYTE  -1,-1,0
021260      377      377      000        .EVEN  ;;NULL CHARACTER STRING
  
```

2
3

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
021264 112737 000001 021530 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERROR
021272 112737 000001 021526 $ATY3:  MOV  #1,$MFLG     ;;TO TYPE A MESSAGE
021300 000403                BR      $ATYC
021302 112737 000001 021530 $ATY4:  MOV  #1,$FFLG     ;;TO ONLY REPORT FATAL ERROR
021310 $ATYC:
021310 010046                MOV  R0,-(SP)      ;;PUSH R0 ON STACK
021312 010146                MOV  R1,-(SP)      ;;PUSH R1 ON STACK
021314 105737 021526                TSTB $MFLG        ;;SHOULD TYPE A MESSAGE?
021320 001450                BEQ   5$          ;;IF NOT: BR
021322 122737 000001 001254        CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
021330 001031                BNE   3$          ;;IF NOT: BR
021332 132737 000100 001255        BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
021340 001425                BEQ   3$          ;;IF NOT: BR
021342 017600 000004                MOV  @4(SP),R0     ;;GET MESSAGE ADDR.
021346 062766 000002 000004        ADD  #2,4(SP)     ;;BUMP RETURN ADDR.
021354 005737 001234                1$:  TST  $MSGTYPE    ;;SEE IF DONE W/ LAST XMISSION?
021360 001375                BNE   1$          ;;IF NOT: WAIT
021362 010037 001250                MOV  R0,$MSGAD    ;;PUT ADDR IN MAILBOX
021366 105720                2$:  TSTB (R0)+      ;;FIND END OF MESSAGE
021370 001376                BNE   2$
021372 163700 001250                SUB  $MSGAD,R0    ;;SUB START OF MESSAGE
021376 006200                ASR  R0           ;;GET MESSAGE LNTH IN WORDS
021400 010037 001252                MOV  R0,$MSGGLT   ;;PUT LENGTH IN MAILBOX
021404 012737 000004 001234        MOV  #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
021412 000413                BR   5$
021414 017637 000004 021440        3$:  MOV  @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
021422 062766 000002 000004        ADD  #2,4(SP)     ;;BUMP RETURN ADDRESS
021430 013746 177776                MOV  177776,-(SP) ;;PUSH 177776 ON STACK
021434 004737 022366                JSR  PC,$TYPE     ;;CALL TYPE MACRO
021440 000000                4$:  .WORD 0
021442                5$:
021442 105737 021530                10$: TSTB $FFLG      ;;SHOULD REPORT FATAL ERROR?
021446 001416                BEQ   12$         ;;IF NOT: BR
021450 005737 001254                TST  $ENV         ;;RUNNING UNDER APT?
021454 001413                BEQ   12$         ;;IF NOT: BR
021456 005737 001234                11$: TST  $MSGTYPE    ;;FINISHED LAST MESSAGE?
021462 001375                BNE   11$         ;;IF NOT: WAIT
021464 017637 000004 001236        MOV  @4(SP),$FATAL ;;GET ERROR #
021472 062766 000002 000004        ADD  #2,4(SP)     ;;BUMP RETURN ADDR.
021500 005237 001234                INC  $MSGTYPE     ;;TELL APT TO TAKE ERROR
021504 105037 021530                12$: CLRB $FFLG    ;;CLEAR FATAL FLAG
021510 105037 021527                CLRB $LFLG       ;;CLEAR LOG FLAG
021514 105037 021526                CLRB $MFLG       ;;CLEAR MESSAGE FLAG
021520 012601                MOV  (SP)+,R1    ;;POP STACK INTO R1
021522 012600                MOV  (SP)+,R0    ;;POP STACK INTO R0
021524 000207                RTS  PC          ;;RETURN
021526 000                $MFLG: .BYTE 0  ;;MESSG. FLAG
021527 000                $LFLG: .BYTE 0  ;;LOG FLAG
021530 000                $FFLG: .BYTE 0  ;;FATAL FLAG
                .EVEN
                APTSIZE = 200
                APTENV = 001
                APTSPOOL = 100
  
```

4
5

000040

APTC SUP = 040

.SBTTL ERROR HANDLER ROUTINE

```

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

```

021532          $ERROR:
021532 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
021534 032777 000400 157412          BIT      #SW08,@SWR          ;;SEND ERROR MESSAGE TO TTY?
021542 001411          BEQ      7$          ;;YES--BRANCH
021544 005737 001324          TST      LPTAVL          ;;IS THERE A LINE PRINTER AVAILABLE?
021550 001406          BEQ      7$          ;;NO--BRANCH
021552 013737 001522 001164          MOV      LPS,$TPS          ;;YES--SETUP STATUS
021560 013737 001524 001166          MOV      LPB,$TPB          ;;AND BUFFER REG.'S FOR LINE PRINTER
021566 105237 001117          7$:      INCB     $ERFLG          ;;SET THE ERROR FLAG
021572 001775          BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
021574 013777 001116 157354          MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
021602 032777 002000 157344          BIT      #BIT10,@SWR          ;;BELL ON ERROR?
021610 001402          BEQ      1$          ;;NO - SKIP
021612 104401 001224          TYPE     $BELL          ;;RING BELL
021616 005237 001126          1$:      INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
021622 011637 001132          MOV      (SP),$ERRPC          ;;GET ADDRESS OF ERROR INSTRUCTION
021626 162737 000002 001132          SUB     #2,$ERRPC
021634 117737 157272 001130          MOV     @ $ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
021642 032777 020000 157304          BIT      #BIT13,@SWR          ;;SKIP TYPEOUT IF SET
021650 001004          BNE     20$          ;;SKIP TYPEOUTS
021652 004737 022000          JSR     PC,TYPERR          ;;GO TO USER ERROR ROUTINE
021656 104401 001231          TYPE     $CRLF
021662          20$:
021662 122737 000001 001254          CMP     #APTENV,$ENV          ;;RUNNING IN APT MODE
021670 001007          BNE     2$          ;;NO,SKIP APT ERROR REPORT
021672 113737 001130 021704          MOV     $ITEMB,21$          ;;SET ITEM NUMBER AS ERROR NUMBER
021700 004737 021302          JSR     PC,$ATY4          ;;REPORT FATAL ERROR TO APT
021704          21$:      .BYTE     0
021705          .BYTE     C
021706 000777          22$:      BR      22$          ;;APT ERROR LOOP
021710 005777 157240          2$:      TST     @SWR          ;;HALT ON ERROR
021714 100002          BPL     3$          ;;SKIP IF CONTINUE
021716 000000          HALT          ;;HALT ON ERROR!
021720 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
021722 032777 001000 157224          3$:      BIT      #BIT09,@SWR          ;;LOOP ON ERROR SWITCH SET?
021730 001402          BEQ     4$          ;;BR IF NO
021732 013716 001124          MOV     $LPERR,(SP)          ;;FUDGE RETURN FOR LOOPING
021736 005737 001222          4$:      TST     $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
021742 001402          BEQ     5$          ;;BR IF NONE
021744 013716 001222          MOV     $ESCAPE,(SP)          ;;FUDGE RETURN ADDRESS FOR ESCAPE
021750          5$:
    
```



```

021750 022737 021244 000042      CMP      #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
021756 001001                      BNE      6$              ;;BRANCH IF NO
021760 000000                      HALT                      ;;YES
021762                                6$:
021762 013737 001516 001164      MOV      TPS,$TPS        ;SET STATUS AND BUFFER REG.'S
021770 013737 001520 001166      MOV      TPB,$TPB        ;FOR TTY
021776 000002                      RTI                       ;RETURN FROM ERROR CALL

6
7
8
9
10
11
12
13
14
15
16
17 022000 113737 001116 001212  TYPERR: MOVB     $TSTNM,$TMP0      ;SAVE THE TEST NUMBER
18 022006 104412                      SAVREG                    ;SAVE R0 - R5
19 022010 162700 000004          SUB      #4,R0            ;FORM TEST PC
20 022014 010037 001176          MOV      R0,$REG0        ;COPY R0-R5 IN $REG0-$REG5
21 022020 010137 001200          MOV      R1,$REG1
22 022024 010237 001202          MOV      R2,$REG2
23 022030 010337 001204          MOV      R3,$REG3
24 022034 010437 001206          MOV      R4,$REG4
25 022040 010537 001210          MOV      R5,$REG5
26 022044 113700 001130          MOVB     $ITEMB,R0       ;PICKUP ERROR ITEM NUMBER
27 022050 010001          MOV      R0,R1           ;AND COPY IT INTO R1
28 022052 005300          DEC      R0              ;FORM INDEX FOR ERROR TABLE
29 022054 106300          ASLB    R0
30 022056 106300          ASLB    R0
31 022060 106300          ASLB    R0
32 022062 103002          BCC     1$              ;IS ERROR > 37?
33 022064 062700 000240          ADD     #ITEM41-$ERRTB,R0 ;YES--FORM OFFSET
34 022070 062700 002002          1$: ADD     #ERRTB,R0      ;FORM ADDRESS
35 022074 012037 022110          MOV     (R0)+,2$        ;GET ERROR MESSAGE (EM) POINTER
36 022100 001447          BEQ     7$              ;BRANCH IF THERE ISN'T ONE
37 022102 104401 001231          TYPE   ,SCRLF          ;'CARRIAGE RETURN - LINE FEED
38 022106 104401          TYPE
39 022110 000000          2$: .WORD   0            ;'EM' POINTER GOES HERE
40 022112 162701 000041          SUB     #41,R1          ;SPECIAL ERROR ITEM NUMBER?
41 022116 100440          BMI     7$              ;NO--BRANCH
42 022120 013701 001354          MOV     SVSTAT,R1      ;GET STATUS/ERROR INDICATOR
43 022124 106301          ASLB   R1              ;STRIP 'DGNE' BIT (BIT07)
44 022126 006301          ASL    R1              ;STRIP 'ERROR' BIT (BIT15)
45 022130 012702 001750          MOV     #STATBL,R2     ;1ST ADDRESS ON STATUS MESSAGE POINTERS
46 022134 005003          CLR    R3              ;CARRIAGE RETURN-LINE FEED SWITCH
47 022136 104401 022144          TYPE   ,65$           ;;TYPE ASCII STRING
022142 000402          BR     64$            ;;GET OVER THE ASCII
;;65$: .ASCII / (/
64$:
48 022150 012237 022172          3$: MOV     (R2)+,5$     ;MESSAGE POINTER
49 022154 006301          ASL    R1              ;TYPE THIS MESSAGE?
50 022156 103013          BCC    6$              ;NO--BRANCH
51 022160 005103          COM    R3              ;YES--TYPE A 'CR' & 'LF'?
52 022162 001002          BNE    4$              ;NO--BRANCH

```

```

53 022164 104401 001231          TYPE      ,%CRLF          ;YES
54 022170 104401          4$:  TYPE
55 022172 000000          5$:  .WORD      0          ;MESSAGE POINTER GOES HERE
56 022174 005701          TST      R1          ;MORE TO TYPE?
57 022176 001403          BEQ      6$          ;NO--BRANCH
58 022200 104401 046657          TYPE      ,BLNKS2          ;TYPE 2 SPACES
59 022204 000761          BR      3$          ;LOOP
60 022206 001360          6$:  BNE      3$          ;BRANCH IF NOT FINISHED
61 022210 104401 022216          TYPE      ,67$          ;:TYPE ASCII STRING
    022214 000401          BR      66$          ;:GET OVER THE ASCII
    .:67$: .ASCIIZ  /)/
    66$:
62 022220 012037 022234          7$:  MOV      (R0)+,%8$          ;PICK UP DATA HEADER (DH) POINTER
63 022224 001404          BEQ      9$          ;BRANCH IF NONE
64 022226 104401 001231          TYPE      ,%CRLF          ;CARRIAGE RETURN-LINE FEED
65 022232 104401          TYPE
66 022234 000000          8$:  .WORD      0          ;'DH' POINTER GOES HERE
67 022236 012001          9$:  MOV      (R0)+,%R1          ;PICKUP DATA TABLE (DT) POINTER
68 022240 001450          BEQ      20$         ;BRANCH IF NONE
69 022242 005005          CLR      R5          ;SET INDENT SWITCH
70 022244 012000          MOV      (R0)+,%R0          ;DATA FORMAT (DF) POINTER
71 022246 012002          MOV      (R0)+,%R2          ;NUMBER OF DH'S TO TYPE
72 022250 001441          BEQ      17$         ;BRANCH IF DH NUMBER IS 0
73 022252 005105          COM      R5          ;NO INDENT
74 022254 104401 001231          TYPE      ,%CRLF          ;CARRIAGE RETURN-LINE FEED
75 022260 112003          10$: MOVB     (R0)+,%R3          ;NUMBER OF DATA WORDS TO TYPE
76 022262 112004          MOVB     (R0)+,%R4          ;AND HOW TO TYPE THEM
77 022264 006004          11$: ROR      R4          ;OCTAL OR DECIMAL?
78 022266 103403          BCS      12$         ;DECIMAL--BRANCH
79 022270 013146          MOV      @(%R1)+,%-(SP) ;:SAVE @(%R1)+ FOR TYPEOUT
    022272 104402          TYPOC
80 022274 000402          BR      13$
81 022276          12$: MOV      @(%R1)+,%-(SP) ;:SAVE @(%R1)+ FOR TYPEOUT
    022276 013146          TYPDS
    022300 104405          13$: DEC      R3          ;:GO TYPE--DECIMAL ASCII WITH SIGN
82 022302 005303          BEQ      14$         ;MORE NUMBERS TO TYPE?
83 022304 001403          BEQ      14$         ;NO--BRANCH
84 022306 104401 046657          TYPE      ,BLNKS2          ;TYPE 2 SPACES
85 022312 000764          BR      11$         ;LOOP
86 022314 005302          14$: DEC      R2          ;MORE DH'S?
87 022316 003421          BLE      20$         ;NO--BRANCH
88 022320 104401 001231          TYPE      ,%CRLF          ;YES--START A NEW LINE
89 022324 005105          COM      R5          ;INDENT?
90 022326 001002          BNE      15$         ;NO--BRANCH
91 022330 104401 046657          TYPE      ,BLNKS2          ;TYPE 2 SPACES
92 022334 012037 022342          15$: MOV      (R0)+,%16$          ;GET NEXT DH
93 022340 104401          TYPE      ,%CRLF          ;AND TYPE IT
94 022342 000000          16$: .WORD      0          ;DH POINTER GOES HERE
95 022344 104401 001231          TYPE      ,%CRLF          ;CARRIAGE RETURN-LINE FEED
96 022350 005705          TST      R5          ;INDENT?
97 022352 001342          BNE      10$         ;NO--BRANCH
98 022354 104401 046657          17$: TYPE      ,BLNKS2          ;TYPE 2 SPACES
99 022360 000737          BR      10$         ;LOOP
100 022362 104413          20$: RESREG          ;RESTORE R0 - R5
101 022364 000207          RTS      PC          ;RETURN
102
103          .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

```

```

022366 105737 001173 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
022372 100002      BPL 1$          ;; BR IF YES
022374 000000      HALT          ;; HALT HERE IF NO TERMINAL
022376 000430      BR 3$          ;; LEAVE
022400 010046      1$: MOV RO,-(SP)  ;; SAVE RO
022402 017600 000002 MOV @2(SP),RO  ;; GET ADDRESS OF ASCIZ STRING
022406 122737 000001 001254 CMPB #APTENV,$ENV  ;; RUNNING IN APT MODE
022414 001011      BNE 62$        ;; NO,GO CHECK FOR APT CONSOLE
022416 132737 000100 001255 BITB #APTSPOOL,$ENVM  ;; SPOOL MESSAGE TO APT
022424 001405      BEQ 62$        ;; NO,GO CHECK FOR CONSOLE
022426 010037 022436 MOV RO,61$     ;; SETUP MESSAGE ADDRESS FOR APT
022432 004737 021272 JSR PC,$ATY3   ;; SPOOL MESSAGE TO APT
022436 000000      61$: .WORD 0    ;; MESSAGE ADDRESS
022440 132737 000040 001255 62$: BITB #APTCSUP,$ENVM  ;; APT CONSOLE SUPPRESSED
022446 001003      BNE 60$        ;; YES,SKIP TYPE OUT
022450 112046      2$: MOVB (RO)+,-(SP)  ;; PUSH CHARACTER TO BE TYPED ONTO STACK
022452 001005      BNE 4$          ;; BR IF IT ISN'T THE TERMINATOR
022454 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
022456 012600      60$: MOV (SP)+,RO  ;; RESTORE RO
022460 062716 000002 3$: ADD #2,(SP)  ;; ADJUST RETURN PC
022464 000002      RTI          ;; RETURN
022466 122716 000011 4$: CMPB #HT,(SP)  ;; BRANCH IF <HT>
022472 001430      BEQ 8$          ;;
022474 122716 000200 CMPB #CRLF,(SP)  ;; BRANCH IF NOT <CRLF>
022500 001006      BNE 5$          ;;
022502 005726      TST (SP)+      ;; POP <CR><LF> EQUIV
022504 104401      TYPE          ;; TYPE A CR AND LF
022506 001231      $CRLF
022510 105037 022716 CLRB $CHARCNT  ;; CLEAR CHARACTER COUNT
022514 000755      BR 2$          ;; GET NEXT CHARACTER
022516 004737 022600 5$: JSR PC,$TYPEC  ;; GO TYPE THIS CHARACTER
022522 123726 001172 6$: CMPB $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
022526 001350      BNE 2$          ;; IF NO GO GET NEXT CHAR.
022530 013746 001170 MOV $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
;; AND THE NULL CHAR.
022534 105366 000001 7$: DECB 1(SP)  ;; DOES A NULL NEED TO BE TYPED?
022540 002770      BLT 6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
022542 004737 022600 JSR PC,$TYPEC  ;; GO TYPE A NULL
022546 105337 022716 DECB $CHARCNT  ;; DO NOT COUNT AS A COUNT
022552 000770      BR 7$          ;; LOOP

```

;HORIZONTAL TAB PROCESSOR

```

022554 112716 000040      8$:   MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
022560 004737 022600      9$:   JSR    PC,$TYPEC    ;;TYPE A SPACE
022564 132737 000007 022716  BITB   #7,$CHARCNT    ;;BRANCH IF NOT AT
022572 001372          BNE    9$             ;;TAB STOP
022574 005726          TST    (SP)+          ;;POP SPACE OFF STACK
022576 000724          BR     2$             ;;GET NEXT CHARACTER
022600                                     $TYPEC:
022600 105777 156354      TSTB   @STKS          ;;CHAR IN KYBD BUFFER?
022604 100022          BPL    10$            ;;BR IF NOT
022606 017746 156350      MOV    @STKB,-(SP)     ;;GET CHAR
022612 042716 177600      BIC    #177600,(SP)   ;;STRIP EXTRANEIOUS BITS
022616 122716 000023      CMPB   #$XOFF,(SP)   ;;WAS CHAR XOFF
022622 001012          BNE    102$          ;;BR IF NOT
022624                                     101$:
022624 105777 156330      TSTB   @STKS          ;;WAIT FOR CHAR
022630 100375          BPL    101$            ;;BR IF NOT
022632 117716 156324      MOVB   @STKB,(SP)     ;;GET CHAR
022636 042716 177600      BIC    #177600,(SP)   ;;STRIP IT
022642 122716 000021      CMPB   #$XON,(SP)    ;;WAS IT XON?
022646 001366          BNE    101$            ;;BR IF NOT
022650                                     102$:
022650 005726          TST    (SP)+          ;;FIX STACK
022652                                     10$:
022652 105777 156306      TSTB   @STPS          ;;WAIT UNTIL PRINTER IS READY
022656 100375          BPL    10$            ;;BR IF NOT
022660 116677 000002 156300  MOVB   2(SP),@STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
022666 122766 000015 000002  CMPB   #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
022674 001003          BNE    1$             ;;BRANCH IF NO
022676 105037 022716      CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
022702 000406          BR     $TYPEX          ;;EXIT
022704 122766 000012 000002 1$:   CMPB   #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
022712 001402          BEQ    $TYPEX          ;;BRANCH IF YES
022714 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
022716 000000      $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
022720 000207      $TYPEX: RTS    PC

```

104
105

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

        ;*          TYPON          ;;CALL FOR TYPEOUT
        ;*
        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
        ;*CALI
        ;*          MOV          NUM,-(SP)      ;;NUMBER TO BE TYPED
        ;*          TYPOC          ;;CALL FOR TYPEOUT

022722 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
022726 116637 000001 023145  MOVB     1(SP),%OFILL      ;;LOAD ZERO FILL SWITCH
022734 112637 023147          MOVB     (SP)+,%SOMODE+1  ;;NUMBER OF DIGITS TO TYPE
022740 062716 000002          ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
022744 000406          BR      $TYPON
022746 112737 000001 023145  $TYPOC: MOVB     #1,%OFILL      ;;SET THE ZERO FILL SWITCH
022754 112737 000006 023147  MOVB     #6,%SOMODE+1  ;;SET FOR SIX(6) DIGITS
022762 112737 000005 023144  $TYPON: MOVB     #5,%OCNT      ;;SET THE ITERATION COUNT
022770 010346          MOV      R3,-(SP)      ;;SAVE R3
022772 010446          MOV      R4,-(SP)      ;;SAVE R4
022774 010546          MOV      R5,-(SP)      ;;SAVE R5
022776 113704 023147          MOVB     %SOMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
023002 005404          NEG      R4
023004 062704 000006          ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
023010 110437 023146          MOVB     R4,%SOMODE  ;;SAVE IT FOR USE
023014 113704 023145          MOVB     %OFILL,R4    ;;GET THE ZERO FILL SWITCH
023020 016605 000012          MOV      12(SP),R5   ;;PICKUP THE INPUT NUMBER
023024 005003          CLR      R3      ;;CLEAR THE OUTPUT WORD
023026 006105          1$:    ROL      R5      ;;ROTATE MSB INTO 'C'
023030 000404          BR      3$
023032 006105          2$:    ROL      R5      ;;FORM THIS DIGIT
023034 006105          ROL      R5
023036 006105          ROL      R5
023040 010503          MOV      R5,R3
023042 006103          3$:    ROL      R3      ;;GET LSB OF THIS DIGIT
023044 105337 023146          DECB     %SOMODE     ;;TYPE THIS DIGIT?
023050 100016          BPL      7$          ;;BF IF NO
023052 042703 177770          BIC      #177770,R3  ;;GET RID OF JUNK
023056 001002          BNE      4$          ;;TEST FOR 0
023060 005704          TST     R4          ;;SUPPRESS THIS 0?
023062 001403          SEQ     5$          ;;BR IF YES
023064 005204          4$:    INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
023066 052703 000060          BIS      #'0,R3     ;;MAKE THIS DIGIT ASCII
023072 052703 000040          5$:    BIS      #' ,R3  ;;MAKE ASCII IF NOT ALREADY
023076 110337 023142          MOVB     R3,8$      ;;SAVE FOR TYPING
023102 104401 023142          TYPE     ,8$      ;;GO TYPE THIS DIGIT
023106 105337 023144          7$:    DECB     %OCNT  ;;COUNT BY 1
023112 003347          BGT     2$          ;;BR IF MORE TO DO
023114 002402          BLT     6$          ;;BR IF DONE
023116 005204          INC     R4          ;;INSURE LAST DIGIT ISN'T A BLANK
023120 000744          BR      2$          ;;GO DO THE LAST DIGIT
023122 012605          6$:    MOV      (SP)+,R5  ;;RESTORE R5
023124 012604          MOV      (SP)+,R4  ;;RESTORE R4
023126 012603          MOV      (SP)+,R3  ;;RESTORE R3
023130 016666 000002 000004  MOV      2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
023136 012616          MOV      (SP)+,(SP)
023140 000002          RTI
023142          8$:    .BYTE 0      ;;RETURN
023143          .BYTE 0      ;;STORAGE FOR ASCII DIGIT
023144          .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
          .BYTE 0      ;;OCTAL DIGIT COUNTER
    
```

023145 000
 023146 000000
 106
 107

\$OFILL: .BYTE 0 ;:ZERO FILL SWITCH
 \$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

023150			\$TYPDS:	MOV R0,-(SP)	::PUSH R0 ON STACK
023150	010046			MOV R1,-(SP)	::PUSH R1 ON STACK
023152	010146			MOV R2,-(SP)	::PUSH R2 ON STACK
023154	010246			MOV R3,-(SP)	::PUSH R3 ON STACK
023156	010346			MOV R5,-(SP)	::PUSH R5 ON STACK
023160	010546			MOV #20200,-(SP)	::SET BLANK SWITCH AND SIGN
023162	012746	020200		MOV 20(SP),R5	::GET THE INPUT NUMBER
023166	016605	000020		BPL 1\$::BR IF INPUT IS POS.
023172	100004			NEG R5	::MAKE THE BINARY NUMBER POS.
023174	005405			MOVB #'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
023176	112766	000055 000001	1\$:	CLR R0	::ZERO THE CONSTANTS INDEX
023204	005000			MOV #SDBLK,R3	::SETUP THE OUTPUT POINTER
023206	012703	023364		MOVB #' ,(R3)+	::SET THE FIRST CHARACTER TO A BLANK
023212	112723	000040	2\$:	CLR R2	::CLEAR THE BCD NUMBER
023216	005002			MOV \$DTBL(R0),R1	::GET THE CONSTANT
023220	016001	023354	3\$:	SUB R1,R5	::FORM THIS BCD DIGIT
023224	160105			BLT 4\$::BR IF DONE
023226	002402			INC R2	::INCREASE THE BCD DIGIT BY 1
023230	005202			BR 3\$	
023232	000774			ADD R1,R5	::ADD BACK THE CONSTANT
023234	060105		4\$:	TST R2	::CHECK IF BCD DIGIT=0
023236	005702			BNE 5\$::FALL THROUGH IF 0
023240	001002			TSTB (SP)	::STILL DOING LEADING 0'S?
023242	105716			BMI 7\$::BR IF YES
023244	100407			ASLB (SP)	::MSD?
023246	106316		5\$:	BCC 6\$::BR IF NO
023250	103003			MOVB 1(SP),-1(R3)	::YES--SET THE SIGN
023252	116663	000001 177777	6\$:	BIS #'0,R2	::MAKE THE BCD DIGIT ASCII
023260	052702	000060	7\$:	BIS #' ,R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
023264	052702	000040		MOVB R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
023270	110223			TST (R0)+	::JUST INCREMENTING
023272	005720			CMR R0,#10	::CHECK THE TABLE INDEX
023274	020027	000010		BLT 2\$::GO DO THE NEXT DIGIT
023300	002746			BGT 8\$::GO TO EXIT
023302	003002			MOV R5,R2	::GET THE LSD
023304	010502			BR 6\$::GO CHANGE TO ASCII
023306	000764			TSTB (SP)+	::WAS THE LSD THE FIRST NON-ZERO?
023310	105726		8\$:	BPL 9\$::BR IF NO
023312	100003			MOVB -1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
023314	116063	177777 177776	9\$:	CLRB (R3)	::SET THE TERMINATOR
023322	105013			MOV (SP)+,R5	::POP STACK INTO R5
023324	012605				

```

023326 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
023330 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
023332 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
023334 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
023336 104401 023364  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
023342 016666 000002 000004  MOV      2(SP),4(SP)    ;;ADJUST THE STACK
023350 012616      MOV      (SP)+,(SP)
023352 000002      RTI                      ;;RETURN TO USER
023354 023420      $DTBL: 10000.
023356 001750      1000.
023360 000144      100.
023362 000012      10.
023364      $DBLK: .BLKW 4
    
```

108
109

.SBTTL TTY INPUT ROUTINE

```

*****
023374 000000      .ENABL  LSB
023376 000000      $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
023400 000000      $TKQIN: .WORD 0      ;;INPUT POINTER
023402 000000      $TKQOUT: .WORD 0     ;;OUTPUT POINTER
023404 023404      $TKQSRV: .BLKB 2    ;;TTY KEYBOARD QUEUE
$TKQEND=.
    
```

;*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
    
```

```

023404 005037 023374      $TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
023410 012737 023402 023376  MOV      # $TKQSRV,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
023416 013737 023376 023400  MOV      $TKQIN,$TKQOUT    ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
023424 012737 023454 000060  MOV      # $TKSRV,@ $TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
023432 012737 000200 000062  MOV      #200,@ $TKVEC+2  ;;'BR' LEVEL 4
023440 005777 155516      TST      @ $TKB           ;;CLEAR DONE FLAG
023444 012777 000100 155506  MOV      #100,@ $TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
023452 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 (START2)

```

023454 117746 155502      $TKSRV: MOVB     @ $TKB, -(SP)  ;;PICKUP THE CHARACTER
023460 042716 177600      BIC      #^C177, (SP)      ;;STRIP THE JUNK
023464 021627 000003      CMP      (SP),#3          ;;IS IT A CONTROL C?
023470 001007      BNE      1$              ;;BRANCH IF NO
023472 104401 024604      TYPE     ,SCNTLC         ;;TYPE A CONTROL-C (^C)
023476 004737 023404      JSR      PC,$TKINT       ;;INIT THE KEYBOARD
023502 005726      TST      (SP)+           ;;CLEAN UP STACK
023504 000137 004612      JMP      START2          ;;CONTROL C RESTART
023510 021627 000007      1$:     CMP      (SP),#7    ;;IS IT A CONTROL G?
    
```

```

023514 001004          BNE 2$          ;;BRANCH IF NO
023516 022737 000176 001154  CMP #SWREG,SWR  ;;IS SOFT-SWR SELECTED?
023524 001500          BEQ 6$          ;;GO TO SWR CHANGE

023526          2$:
023526 022737 000002 023374  CMP #2,$TKCNT  ;;IS THE QUEUE FULL?
023534 001004          BNE 3$          ;;BRANCH IF NO
023536 104401 001224          TYPE ,SBELL    ;;RING THE TTY BELL
023542 005726          TST (SP)+      ;;CLEAN CHARACTER OFF OF STACK
023544 000451          BR 5$          ;;EXIT
023546 021627 000023          3$:  CMP (SP),#23    ;;IS IT A CONTROL-S?
023552 001021          BNE 32$         ;;BRANCH IF NO
023554 005077 155400          CLR @STKS     ;;DISABLE TTY KEYBOARD INTERRUPTS
023560 005726          TST (SP)+      ;;CLEAN CHAR OFF STACK
023562 105777 155372          31$: TSTB @STKS    ;;WAIT FOR A CHAR
023566 100375          BPL 31$         ;;LOOP UNTIL ITS THERE
023570 117746 155366          MOVB @STKB,-(SP) ;;GET THE CHARACTER
023574 042716 177600          BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
023600 022627 000021          CMP (SP)+,#21  ;;IS IT A CONTROL-Q?
023604 001366          BNE 31$         ;;BRANCH IF NO
023606 012777 000100 155344  MOV #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
023614 000002          RTI          ;;RETURN
023616 005237 023374          32$: INC $TKCNT     ;;COUNT THIS CHARACTER
023622 021627 000140          CMP (SP),#140  ;;IS IT UPPER CASE?
023626 002405          BLT 4$          ;;BRANCH IF YES
023630 021627 000175          CMP (SP),#175  ;;IS IT A SPECIAL CHAR?
023634 003002          BGT 4$          ;;BRANCH IF YES
023636 042716 000040          BIC #40,(SP)  ;;MAKE IT UPPER CASE
023642 112677 177530          4$:  MOVB (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
023646 005237 023376          INC $TKQIN    ;;UPDATE THE POINTER
023652 023727 023376 023404  CMP $TKQIN,$STKQEND ;;GO OFF THE END?
023660 001003          BNE 5$         ;;BRANCH IF NO
023662 012737 023402 023376  MOV #STKQSRRT,$TKQIN ;;RESET THE POINTER
023670 000002          5$:  RTI          ;;RETURN
  
```

```

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

```

023672 022737 000176 001154  $CKSWR: CMP #SWREG,SWR  ;;IS THE SOFT-SWR SELECTED
023700 001124          BNE 15$         ;;EXIT IF NOT
023702 105777 155252          TSTB @STKS    ;;IS A CHAR WAITING?
023706 100121          BPL 15$         ;;IF NOT, EXIT
023710 117746 155246          MOVB @STKB,-(SP) ;;YES
023714 042716 177600          BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
023720 021627 000007          CMP (SP),#7    ;;IS IT A CONTROL-G?
023724 001300          BNE 2$         ;;IF NOT, PUT IT IN THE TTY QUEUE
                          ;;AND EXIT
  
```

```

*****
;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
  
```

```

023726 123727 001150 000001  6$:  CMPB $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
023734 001674          BEQ 2$          ;;BRANCH IF YES
023736 005726          TST (SP)+      ;;CLEAR CONTROL-G OFF STACK
  
```


023740	004737	023404		JSR	PC,\$TKINT	::FLUSH THE TTY INPUT QUEUE
023744	005077	155210		CLR	@\$TKS	::DISABLE TTY KEYBOARD INTERRUPTS
023750	112737	000001	001151	MOVB	#1,\$INTAG	::SET INTERRUPT MODE INDICATOR
023756	104401	024616		TYPE	,\$CNTLG	::ECHO THE CONTROL-G (^G)
023762	104401	024623		\$GTSWR: TYPE	,\$MSWR	::TYPE CURRENT CONTENTS
023766	013746	000176		MOV	SWREG,-(SP)	::SAVE SWREG FOR TYPEOUT
023772	104402			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
023774	104401	024634		TYPE	,\$MNEW	::PROMPT FOR NEW SWR
024000	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
024002	005046			CLR	-(SP)	::THE NEW SWR
024004	105777	155150	7\$:	TSTB	@\$TKS	::CHAR THERE?
024010	100375			BPL	7\$::IF NOT TRY AGAIN
024012	117746	155144		MOVB	@\$TKB,-(SP)	::PICK UP CHAR
024016	042716	177600		BIC	#^C177,(SP)	::MAKE IT 7-BIT ASCII
024022	021627	000003		CMP	(SP),#3	::IS IT A CONTROL-C?
024026	001015			BNE	9\$::BRANCH IF NOT
024030	104401	024604		TYPE	,\$CNTLC	::YES, ECHO CONTROL-C (^C)
024034	062706	000006		ADD	#6,SP	::CLEAN UP STACK
024040	123727	001151	000001	CMPB	\$INTAG,#1	::REENABLE TTY KEYBOARD INTERRUPTS?
024046	001003			BNE	8\$::BRANCH IF NO
024050	012777	000100	155102	MOV	#100,@\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
024056	000137	004612	8\$:	JMP	START2	::CONTROL-C RESTART
024062	021627	000025	9\$:	CMP	(SP),#25	::IS IT A CONTROL-U?
024066	001005			BNE	10\$::BRANCH IF NOT
024070	104401	024611		TYPE	,\$CNTLU	::YES, ECHO CONTROL-U (^U)
024074	062706	000006	20\$:	ADD	#6,SP	::IGNORE PREVIOUS INPUT
024100	000737			BR	19\$::LET'S TRY IT AGAIN
024102	021627	000015	10\$:	CMP	(SP),#15	::IS IT A <CR>?
024106	001022			BNE	16\$::BRANCH IF NO
024110	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
024114	001403			BEQ	11\$::BRANCH IF YES
024116	016677	000002	155030	MOV	2(SP),@SWR	::SAVE NEW SWR
024124	062706	000006	11\$:	ADD	#6,SP	::CLEAR UP STACK
024130	104401	001231	14\$:	TYPE	,\$CRLF	::ECHO <CR> AND <LF>
024134	123727	001151	000001	CMPB	\$INTAG,#1	::RE-ENABLE TTY KBD INTERRUPTS?
024142	001003			BNE	15\$::BRANCH IF NOT
024144	012777	000100	155006	MOV	#100,@\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
024152	000002		15\$:	RTI		::RETURN
024154	004737	022600	16\$:	JSR	PC,\$TYPEC	::ECHO CHAR
024160	021627	000060		CMP	(SP),#60	::CHAR < 0?
024164	002420			BLT	18\$::BRANCH IF YES
024166	021627	000067		CMP	(SP),#67	::CHAR > 7?
024172	003015			BGT	18\$::BRANCH IF YES
024174	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
024200	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
024204	001403			BEQ	17\$::BRANCH IF YES
024206	006316			ASL	(SP)	::NO, SHIFT PRESENT
024210	006316			ASL	(SP)	::CHAR OVER TO MAKE
024212	006316			ASL	(SP)	::ROOM FOR NEW ONE.
024214	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR

TTY INPUT ROUTINE

```

024220 056616 177776          BIS      -2(SP),(SP)    ;;SET IN NEW CHAR
024224 000667                BR       7$           ;;GET THE NEXT ONE
024226 104401 001230      18$:  TYPE    $QUES      ;;TYPE ?<CR><LF>
024232 000720                BR       20$           ;;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

```

```

024234 011646                $RDCHR: MOV     (SP),-(SP)    ;;PUSH DOWN THE PC AND
024236 016666 000004 000002  MOV     4(SP),2(SP)    ;;THE PS
024244 005066 000004        CLR     4(SP)        ;;GET READY FOR A CHARACTER
024250 005046                CLR     -(SP)        ;;PUT NEW PS ON STACK
024252 012746 024260        MOV     #64$,-(SP)    ;;PUT NEW PC ON STACK
024256 000002                RTI                    ;;POP NEW PC AND PS
024260
024260 005737 023374      64$:  TST     $TKCNT        ;;WAIT ON A CHARACTER
024264 001775                BEQ     1$
024266 005337 023374      1$:   DEC     $TKCNT        ;;DECREMENT THE COUNTER
024272 117766 177102 000004  MOV     @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
024300 005237 023400        INC     $TKQOUT        ;;UPDATE THE POINTER
024304 023727 023400 023404  CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
024312 001003                BNE     2$           ;;BRANCH 'F NO
024314 012737 023402 023400  MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
024322 000002                RTI                    ;;RETURN

```

```

*****
*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

```

```

024324 010346                $RDLIN: MOV     R3,-(SP)    ;;SAVE R3
024326 005046                CLR     -(SP)        ;;CLEAR THE RUBOUT KEY
024330 012703 024560      1$:   MOV     #$TTYIN,R3    ;;GET ADDRESS
024334 022703 024604      2$:   CMP     #$TTYIN+20.,R3  ;;BUFFER FULL?
024340 101456                BLOS   4$           ;;BR IF YES
024342 104410                RDCHR                ;;GO READ ONE CHARACTER FROM THE TTY
024344 112613                MOV     (SP)+,(R3)    ;;GET CHARACTER
024346 122713 000177      10$:  CMP     #177,(R3)    ;;IS IT A RUBOUT
024352 001022                BNE     5$           ;;BR IF NO
024354 005716                TST     (SP)        ;;IS THIS THE FIRST RUBOUT?
024356 001007                BNE     6$           ;;BR IF NO
024360 112737 000134 024556  MOV     #'\",9$      ;;TYPE A BACK SLASH
024366 104401 024556        TYPE    ,9$
024372 012716 177777        MOV     #-1,(SP)    ;;SET THE RUBOUT KEY
024376 005303      6$:   DEC     R3           ;;BACKUP BY ONE
024400 020327 024560        CMP     R3,$TTYIN  ;;STACK EMPTY?
024404 103434                BLO   4$           ;;BR IF YES
024406 111337 024556        MOV     (R3),9$    ;;SETUP TO TYPEOUT THE DELETED CHAR.
024412 104401 024556        TYPE    ,9$      ;;GO TYPE

```

```

024416 000746          BR      2$          ;;GO READ ANOTHER CHAR.
024420 005716          TST      (SP)          ;;RUBOUT KEY SET?
024422 001406          BEQ      7$          ;;BR IF NO
024424 112737 000134 024556  MOVB    #' \,9$        ;;TYPE A BACK SLASH
024432 104401 024556          TYPE    ,9$
024436 005016          CLR      (SP)          ;;CLEAR THE RUBOUT KEY
024440 122713 000025          CMPB   #25,(R3)        ;;IS CHARACTER A CTRL U?
024444 001003          BNE     8$          ;;BR IF NO
024446 104401 024611          TYPE   ,SCNTLU        ;;TYPE A CONTROL 'U'
024452 000726          BR      1$          ;;GO START OVER
024454 122723 000022          CMPB   #22,(R3)        ;;IS CHARACTER A '^R'?
024462 105013          BNE     3$          ;;BRANCH IF NO
024464 104401 001231          CLRB   (R3)          ;;CLEAR THE CHARACTER
024470 104401 024560          TYPE   ,SCRLF        ;;TYPE A 'CR' & 'LF'
024474 000717          TYPE   ,STTYIN       ;;TYPE THE INPUT STRING
024476 104401 001230          BR      2$          ;;GO PICKUP ANOTHER CHACTER
024502 000712          TYPE   ,SQUES        ;;TYPE A '?'
024504 111337 024556          BR      1$          ;;CLEAR THE BUFFER AND LOOP
024510 104401 024556          MOVB   (R3),9$        ;;ECHO THE CHARACTER
024514 122723 000015          TYPE    ,9$
024520 001305          CMPB   #15,(R3)+      ;;CHECK FOR RETURN
024522 105063 177777          BNE     2$          ;;LOOP IF NOT RETURN
024526 104401 001232          CLRB   -1(R3)        ;;CLEAR RETURN (THE 15)
024532 005726          TYPE   ,SLF          ;;TYPE A LINE FEED
024534 012603          TST    (SP)+         ;;CLEAN RUBOUT KEY FROM THE STACK
024536 011646          MOV     (SP)+,R3      ;;RESTORE R3
024540 016666 000004 000002  MOV     (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
024546 012766 024560 000004  MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
024554 000002          MOV     #STTYIN,4(SP)
024556 000          RTI          ;;RETURN
024557 000          9$: .BYTE 0          ;;STORAGE FOR ASCII CHAR. TO TYPE
024560          .BYTE 0          ;;TERMINATOR
024604 136 103 015 $TTYIN: .BLKB 20.          ;;RESERVE 20. BYTES FOR TTY INPUT
024611 136 125 015 $CNTLC: .ASCIZ /^C/<15><12>          ;;CONTROL 'C'
024616 136 107 015 $CNTLU: .ASCIZ /^U/<15><12>          ;;CONTROL 'U'
024623 015 012 123 $CNTLG: .ASCIZ /^G/<15><12>          ;;CONTROL 'G'
024634 040 040 116 $MSWR: .ASCIZ <15><12>/SWR = /
          $MNEW: .ASCIZ / NEW = /
          .EVEN

```

110
111

.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=10T

```

```

024646          $SCOPE:
024646 104407          CKSWR
024650 032777 040000 154276 1$: BIT    #BIT14,@SWR          ;;TEST FOR CHANGE IN SOFT-SWR
024656 001101          BNE   $OVER          ;;LOOP ON PRESENT TEST?
                                ;;YES IF SW14-1

```

```
024660 000416          :#####START OF CODE FOR THE XOR TESTER#####
          $XTSTR: 3R      6$          ;; IF RUNNING ON THE 'XOR' TESTER CHANGE
024662 013746 000004          MOV      @#ERRVEC,-(SP)          ;; THIS INSTRUCTION TO A 'NOP' (NOP=240)
024666 012737 024706 000004          MOV      #5$,@#ERRVEC          ;; SAVE THE CONTENTS OF THE ERROR VECTOR
024674 005737 177060          TST      @#177060          ;; SET FOR TIMEOUT
024700 012637 000004          MOV      (SP)+,@#ERRVEC          ;; TIME OUT ON XOR?
024704 000450          BR      $SVLAD          ;; RESTORE THE ERROR VECTOR
024706 022626          5$:    CMP      (SP)+,(SP)+          ;; GO TO THE NEXT TEST
024710 012637 000004          MOV      (SP)+,@#ERRVEC          ;; CLEAR THE STACK AFTER A TIME OUT
024714 000413          BR      7$          ;; RESTORE THE ERROR VECTOR
024716          6$:;#####END OF CODE FOR THE XOR TESTER#####          ;; LOOP ON THE PRESENT TEST
024716 105737 001117          2$:    TSTB     $ERFLG          ;; HAS AN ERROR OCCURRED?
024722 001421          BEQ      3$          ;; BR IF NO
024724 123737 001131 001117          CMPB     $ERMAX,$ERFLG          ;; MAX. ERRORS FOR THIS TEST OCCURRED?
024732 101015          BHI      3$          ;; BR IF NO
024734 032777 001000 154212          BIT     #BIT09,@SWR          ;; LOOP ON ERROR?
024742 001404          BEQ      4$          ;; BR IF NO
024744 013737 001124 001122          7$:    MOV      $LPERR,$LPADR          ;; SET LOOP ADDRESS TO LAST SCOPE
024752 000443          BR      $OVER          ;;
024754 105037 001117          4$:    CLRB     $ERFLG          ;; ZERO THE ERROR FLAG
024760 005037 001220          CLR     $TIMES          ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
024764 000412          BR      1$          ;; ESCAPE TO THE NEXT TEST
024766 032777 004000 154160          3$:    BIT     #BIT11,@SWR          ;; INHIBIT ITERATIONS?
024774 001006          BNE     1$          ;; BR IF YES
024776 005237 001120          INC     $ICNT          ;; INCREMENT ITERATION COUNT
025002 023737 001220 001120          CMP     $TIMES,$ICNT          ;; CHECK THE NUMBER OF ITERATIONS MADE
025010 002024          BGE     $OVER          ;; BR IF MORE ITERATION REQUIRED
025012 012737 000001 001120          1$:    MOV      #1,$ICNT          ;; REINITIALIZE THE ITERATION COUNTER
025020 013737 025076 001220          MOV     $MXCNT,$TIMES          ;; SET NUMBER OF ITERATIONS TO DO
025026 105237 001116          $SVLAD: INCB     $TSTNM          ;; COUNT TEST NUMBERS
025032 113737 001116 001240          MOVB    $TSTNM,$TESTN          ;; SET TEST NUMBER IN APT MAILBOX
025040 011637 001122          MOV     (SP),$LPADR          ;; SAVE SCOPE LOOP ADDRESS
025044 011637 001124          MOV     (SP),$LPERR          ;; SAVE ERROR LOOP ADDRESS
025050 005037 001222          CLR     $ESCAPE          ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
025054 112737 000001 001131          MOVB    #1,$ERMAX          ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
025062 013777 001116 154066          $OVER:  MOV     $TSTNM,@DISPLAY          ;; DISPLAY TEST NUMBER
025070 013716 001122          MOV     $LPADR,(SP)          ;; FUDGE RETURN ADDRESS
025074 000002          RTI          ;; FIXES PS
025076 000001          $MXCNT: 1          ;; MAX. NUMBER OF ITERATIONS
```

112
113

.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
```

```

025100          $SAVREG:
025100 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
025102 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
025104 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
025106 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
025110 010446      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
025112 010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
025114 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
025120 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
025124 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PS OF CALL
025130 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PC OF CALL
025134 000002      RTI
    
```

```

;*RESTORE R0-R5
;*CALL:
;*
;* RESREG
025136          $RESREG:
025136 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PC OF CALL
025142 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PS OF CALL
025146 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
025152 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
025156 012605      MOV     (SP)+,R5      ;;POP STACK INTO R5
025160 012604      MOV     (SP)+,R4      ;;POP STACK INTO R4
025162 012603      MOV     (SP)+,R3      ;;POP STACK INTO R3
025164 012602      MOV     (SP)+,R2      ;;POP STACK INTO R2
025166 012601      MOV     (SP)+,R1      ;;POP STACK INTO R1
025170 012600      MOV     (SP)+,R0      ;;POP STACK INTO R0
025172 000002      RTI
    
```

114
115

.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.
    
```

```

025174 010046      $TRAP: MOV     R0,-(SP)      ;;SAVE R0
025176 016600 000002  MOV     2(SP),R0      ;;GET TRAP ADDRESS
025202 005740      TST     -(R0)      ;;BACKUP BY 2
025204 111000      MOVVB  (R0),R0      ;;GET RIGHT BYTE OF TRAP
025206 006300      ASL     R0          ;;POSITION FOR INDEXING
025210 016000 025230  MOV     $TRPAD(R0),R0 ;;INDEX TO TABLE
025214 000200      RTS     R0          ;;GO TO ROUTINE
    
```

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

```

025216 011646      $TRAP2: MOV     (SP),-(SP)  ;;MOVE THE PC DOWN
025220 016666 000004 000002  MOV     4(SP),2(SP)  ;;MOVE THE PSW DOWN
025226 000002      RTI          ;;RESTORE THE PSW
    
```

.SBTTL TRAP TABLE

```

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

```

:          ROUTINE
:          -----
025230 025216 $TRPAD: .WORD $TRAP2
025232 022366 $TYPE  ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
025234 022746 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
025236 022722 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
025240 022762 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
025242 023150 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

025244 023762 $GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

025246 023672 $CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
025250 024234 $RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
025252 024324 $RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
025254 025100 $SAVREG ;;CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
025256 025136 $RESREG ;;CALL=RESREG TRAP+13(104413) RESTORE R0-R5 ROUTINE

```

116
117

.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

```

```

025260 016637 000002 025310 $SB2D: MOV    2(SP),1$    ;;SAVE BINARY NUMBER
025266 012746 025310        MOV    #1$,-(SP)    ;;SET POINTER
025272 004737 025314        JSR    PC,@#$DB2D   ;;CALL DOUBLE LENGTH CONVERT
025276 062716 000005        ADD    #5,(SP)     ;;ONLY ALLOW FIVE CHARACTERS
025302 012666 000002        MOV    (SP)+,2(SP)  ;;PICKUP POINTER
025306 000207                RTS    PC           ;;RETURN
025310 000000 000000        1$:  .WORD  0,0

```

118
119

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

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

```

```

025314 104412 $DB2D: SAVREG    ;;SAVE REGISTERS
025316 016602 000002        MOV    2(SP),R2    ;;PICKUP THE DATA POINTER
025322 012700 025474        MOV    #$DECVL,R0  ;;GET ADDRESS OF "$DECVL" STRING
025326 010066 000002        MOV    R0,2(SP)   ;;PUT ADDRESS OF ASCII STRING ON STACK
025332 012201                MOV    (R2)+,R1    ;;PICKUP THE BINARY NUMBER
025334 012202                MOV    (R2)+,R2
025336 012737 000012 025412        MOV    #10.,4$    ;;SET UP TO DO 10 CONVERSIONS

```

```

025344 012704 025424      MOV    #STNPWR,R4      ;;ADDRESS OF TEN POWER
025350 012705 025426      MOV    #STNPWR+2,R5
025354 005003      1$:   CLR    R3          ;;CLEAR PARTIAL
025356 161401      2$:   SUB    (R4),R1     ;;SUBTRACT TEN POWER
025360 005602      SBC    R2
025362 161502      SUB    (R5),R2
025364 002402      BLT    3$             ;;BR IF TEN POWER TO LARCE
025366 005203      INC    R3             ;;ADD 1 TO PARTIAL
025370 000772      BR     2$             ;;LOOP
025372 062401      3$:   ADD    (R4)+,R1    ;;RESTORE SUBTRACTED VALUE
025374 005502      ADC    R2
025376 062402      ADD    (R4)+,R2
025400 022525      CMP    (R5)+,(R5)+   ;;MOVE TO NEXT TEN POWER
025402 052703 000060      BIS    #'0,R3        ;;CHANGE PARTIAL TO ASCII
025406 110320      MOVB   R3,(R0)+      ;;SAVE IT
025410 005327      DEC    (PC)+         ;;DONE?
025412 000000      4$:   .WORD  0
025414 001357      BNE    1$            ;;BR IF NO
025416 105020      CLRB  (R0)+          ;;TERMINATOR
025420 104413      RESREG
025422 000207      RTS    PC            ;;RESTORE REGISTERS
025424 145000      $STNPWR: 145000      ;;RETURN
025426 035632      35632                ;;1.0E09
025430 160400      160400                ;;1.0E08
025432 002765      2765                  ;;1.0E07
025434 113200      113200                ;;1.0E06
025436 000230      230                   ;;1.0E05
025440 041100      041100                ;;1.0E04
025442 000017      17                    ;;1.0E03
025444 103240      103240                ;;1.0E02
025446 000001      1                     ;;1.0E01
025450 023420      23420                ;;1.0E00
025452 000000      0
025454 001750      1750
025456 000000      0
025460 000144      144
025462 000000      0
025464 000012      12
025466 000000      0
025470 000001      1
025472 000000      0
025474      $DECVL: .BLKB 12.      ;;RESERVE STORAGE FOR ASCII STRING

120
121
.SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS

;*****
;*THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE
;*LEADING NUMBERS.
;*CALL
;*   MOV    #NUMADR,-(SP)  ;;FIRST ADDRESS OF ASCII STRING
;*   JSR    PC,@#SSUPRS

025510 010046      $SUPRS: MOV    R0,-(SP)  ;;SAVE R0
025512 016000 000004      MOV    4(SP),R0      ;;PICKUP THE POINTER
025516 105710      1$:   TSTB  (R0)        ;;TERMINATOR?
025520 001403      BEQ    2$            ;;BR IF YES
    
```

122
123

025522 122720 000060
 025526 001773
 025530 005300
 025532 010037 025540
 025536 104401
 025540 000000
 025542 012600
 025544 012616
 025546 000207

```

    CMPB    #'0,(R0)+    ;;IS THIS AN ASCII '0' ?
    BEQ     1$           ;;BR IF YES
    2$:    DEC     R0      ;;BACKUP BY '1'
          MOV     R0,3$   ;;SAVE FOR TYPING
          TYPE    ;;GO TYPE
    3$:    .WORD   0      ;;ASCIZ POINTER GOES HERE
          MOV     (SP)+,R0 ;;RESTORE R0
          MOV     (SP)+,(SP) ;;RESTORE THE STACK
          RTS     PC      ;;RETURN
    
```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

```

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

124
125

025550
 025550 010046
 025552 010146
 025554 010246
 025556 013700 025650
 025562 013701 025646
 025566 012702 177771
 025572 006300
 025574 006101
 025576 005202
 025600 001374
 025602 063700 025650
 025606 005501
 025610 063701 025646
 025614 062700 001057
 025620 005501
 025622 062701 047401
 025626 010037 025650
 025632 010137 025646
 025636 012602
 025640 012601
 025642 012600
 025644 000207
 025646 176543
 025650 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     $HINUM,R1   ;;ADD NUMBER TO MAKE X 129
          ADD     #1057,R0    ;;ADD LOW CONSTANT
          ADC     R1          ;;PROPOGATE CARRY
          ADD     #47401,R1   ;;ADD HIGH CONSTANT
          MOV     R0,$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

```

    ;*****
    ;*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
: *      'V'=0   IMPLIES NO ERROR
: *      'V'=1   IMPLIES ERROR OCCURRED
: *      'C'=0   DIVIDE OVERFLOW OCCURRED
: *      'C'=1   ATTEMPTED TO DIVIDE BY ZERO
: *
: *      STACK      NO ERROR      OVERFLOW      DIVIDE BY ZERO
: *      -----
: *      TOP        REMAINDER     ALL ZEROS      ALL ONES
: *      +2         QUOTIENT      ALL ZEROS      ALL ONES
: *
025652          $DIV:
025652 104400          TRAP
025654 042716 000017  BIC      #17,(SP)      ;;PUSH OLD PSW AND PC ON STACK
025660 010046          MOV      R0,-(SP)      ;;STRIP AWAY CONDITION CODES
025662 010146          MOV      R1,-(SP)      ;;PUSH R0 ON STACK
025664 010246          MOV      R2,-(SP)      ;;PUSH R1 ON STACK
025666 010346          MOV      R3,-(SP)      ;;PUSH R2 ON STACK
025670 005046          CLR      -(SP)         ;;PUSH R3 ON STACK
025672 012746 000021  MOV      #17,-(SP)     ;;SAVE A PLACE FOR SIGNS
025676 016601 000024  MOV      24(SP),R1     ;;SETUP THE ITERATION COUNTER
025702 016600 000022  MOV      22(SP),R0     ;;PICKUP THE DIVIDEND
025706 100005          BPL      1$           ;;CHECK THE SIGN
025710 105366 000003  DECB    3(SP)         ;;KEEP TRACK OF THE SIGN
025714 005400          NEG      R0           ;;AND NEGATE THE ORIGINAL
025716 005401          NEG      R1           ;;NUMBER
025720 005600          SBC      R0
025722 016602 000020  1$:     MOV      20(SP),R2     ;;PICKUP THE DIVISOR
025726 002407          BLT      2$           ;;CHECK THE SIGN
025730 003011          BGT      3$           ;;DIVISOR OF 0 IS A NO-NO
025732 052766 000003 000014  BIS      #3,14(SP)     ;;SET 'V' & 'C'
025740 012700 177777          MOV      #-1,R0       ;;SET REMAINDER TO ALL ONES
025744 000424          BR      7$           ;;EXIT
025746 005266 000002  2$:     INC      2(SP)         ;;KEEP TRACK OF DIVISORS SIGN
025752 000401          BR      4$
025754 005402          3$:     NEG      R2           ;;NEGATE THE ORIGINAL NUMBER
025756 000241          4$:     CLC
025760 000405          BR      6$           ;;START FORMING QUOTIENT
025762 006100          5$:     ROL      R0           ;;POSITION MSB'S
025764 010003          MOV      R0,R3        ;;COPY
025766 060203          ADD      R2,R3        ;;COMPARE DIVIDEND & DIVISOR
025770 103001          BCC      6$           ;;BR IF DIVIDEND > DIVISOR
025772 010300          MOV      R3,R0        ;;REMAINDER AFTER THIS LOOP
025774 006101          6$:     ROL      R1           ;;QUOTIENT BIT ENTERS HERE
025776 005316          DEC      (SP)         ;;DONE?
026000 001370          BNE      5$           ;;BR IF NO
026002 005701          TST      R1           ;;OVERFLOW?
026004 100005          BPL      8$           ;;BR IF NO
026006 052766 000002 000014  BIS      #2,14(SP)     ;;SET 'V' IN RETURN STATUS WORD
026014 005000          CLR      R0           ;;SET REMAINDER TO ALL ZEROS
026016 010001          7$:     MOV      R0,R1        ;;COPY REMAINDER INTO QUOTIENT
026020 005726          8$:     TST      (SP)+     ;;CLEAR COUNTER FROM STACK
026022 005716          TST      (SP)        ;;REMAINDER SIGN CORRECTION NEEDED?

```

```

026024 002004          BGE      9$          ;;BR IF NO
026026 005400          NEG      R0          ;;NEGATE REMAINDER
026030 105066 000001   CLR      1(SP)       ;;CLEAR SIGN
026034 005316          DEC      (SP)       ;;BUT DON'T FORGET QUOTIENT
026036 005726          9$:   TST      (SP)+     ;;QUOTIENT SIGN CORRECTION NEEDED?
026040 001401          BEQ      10$         ;;BR IF NO
026042 005401          NEG      R1          ;;NEGATE QUOTIENT
026044 010166 000020   10$:  MOV      R1,20(SP)    ;;RETURN QUOTIENT AND
026050 010066 000016   MOV      R0,16(SP)    ;;REMAINDER TO USER
026054 012603          MOV      (SP)+,R3    ;;POP STACK INTO R3
026056 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
026060 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
J26062 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
026064 012666 000002   MOV      (SP)+,2(SP)  ;;SETUP TO RETURN CONDITION CODES
026070 000002          RTI          ;;RETURN

126
127          ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTER,
128          ;AND DRIVERS, THEN SELECTS THE DRIVE.
129          ;CALL:
130          ;      JSR      PC,CNTCLR      ;CALL TO ROUTINE
131          ;
132
133 026072 013704 037066   CNTCLR: MOV      RMADR,R4      ;GET RMCS1 BASE ADDRESS
134 026076 012764 000040 000010   MOV      #CLR,RMCS2(R4)    ;ISSUE MASSBUS CLEAR AND
135 026104 013764 001350 000010   MOV      CHKDRV,RMCS2(R4)  ;SELECT THE DRIVE
136 026112 000207          RTS      PC          ;RETURN
137
138          ;SET 'LPTAVL' TO THE PROPER STATE.
139          ; LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
140          ; LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
141          ;CALL
142          ;      JSR      PC,LP.AVL
143          ;      RETURN
144
145 026114 005037 001324   LP.AVL: CLR      LPTAVL      ;START WITH NO PRINTER AVAILBLE
146 026120 012737 026144 000004   MOV      #1$,ERRVEC      ;SETUP THE TIMEOUT VECTOR
147 026126 005037 000006   CLR      ERRVEC+2
148 026132 005777 153364   TST      @LPS            ;IS THERE A LINE PRINTER?
149 026136 005237 001324   INC      LPTAVL          ;YES--SET AVAILABLE SWITCH
150 026142 000401          BR      2$
151 026144 022626          1$:   CMP      (SP)+,(SP)+    ;NO--POP STACK
152 026146 012737 000006 000004   2$:   MOV      #ERRVEC+2,ERRVEC ;RESTORE TIMEOUT VECTOR
153 026154 000207          RTS      PC          ;RETURN
154
155          ;THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
156          ;AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK
157          ;"CLKSTA" WILL INDICATE THE CLOCK TYPE
158          ; 0= NO CLOCK
159          ; +1= KW11-P
160          ; -1= KW11-L
161          ;THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
162          ;PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
163          ;(TIME PER CLOCK TICK IN MICROSECONDS) AS
164          ;PER SW00.
165          ;SW00=0 -- 60HZ
166          ;SW00=1 -- 50HZ
167          ;CALL
    
```

```

168
169
170
171 026156 010146
172 026160 012701 000006
173 026164 011146
174 026166 005011
175 026170 014146
176 026172 012711 026222
177 026176 005037 001340
178 026202 005777 153274
179 026206 012737 000001 001340
180 026214 004737 026324
181 026220 000414
182 026222 022626
183 026224 012711 026250
184 026230 005777 153260
185 026234 012737 177777 001340
186 026242 004737 026366
187 026246 000401
188 026250 022626
189 026252 012621
190 026254 012621
191 026256 012601
192 026260 032737 000100 001314
193 026266 001407
194 026270 012737 000020 001342
195 026276 012737 047040 001344
196 026304 000406
197 026306 012737 000016 001342
198 026314 012737 040432 001344
199 026322 000207
200
201 026324
202 026324 032737 000040 001314
203 026332 001014
204 026334 012777 026422 153134
205 026342 012777 000300 153130
206 026350 012777 000001 153126
207 026356 012777 000115 153116
208
209 026364 000207
210
211 026366
212 026366 032737 000040 001314
213 026374 001011
214 026376 012777 026422 153104
215 026404 012777 000300 153100
216 026412 012777 000100 153074
217 026420 000207
218
219 026422 013746 001342
220 026426 004737 043324
221 026432 000002
222
223
224

```

```

: JSR PC,ST.CLK
: RETURN
ST.CLK: MOV R1,-(SP) ;SAVE R1
MOV #ERRVEC+2,R1 ;SAVE AND SETUP TIMEOUT VECTOR
MOV (R1),-(SP)
CLR (R1) ;LEVEL 0
MOV -(R1),-(SP)
MOV #1$, (R1) ;GO TO 1$ ON TIMEOUT
CLR CLKSTA ;SET CLOCK STATUS TO NO CLOCK
TST @PKCS ;IS THERE A KW11-P?
MOV #1,CLKSTA ;YES--SET STATUS TO KW11-P
JSR PC,ST.PCLK ;START THE KW11-P
BR 3$ ;GO TO EXIT
1$: CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
MOV #2$, (R1) ;IF TIMEOUT GO TO 2$
TST @LKS ;IS THERE A KW11-L?
MOV #-1,CLKSTA ;YES-- SET STATUS TO KW11-L
JSR PC,ST.LCLK ;START THE KW11-L
BR 3$ ;EXIT
2$: CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
3$: MOV (SP)+,(R1)+ ;RESTORE THE TIMEOUT VECTOR
MOV (SP)+,(R1)+
MOV (SP)+,R1 ;RESTORE R1
BIT #SW06,C.SWR ;50HZ OR 60HZ?
BEQ 4$ ;BRANCH IF 60
MOV #20,TICKMS ;SETUP TIME PER
MOV #20000.,TICKUS ;TICK FOR 50HZ
BR 5$
4$: MOV #16,TICKMS ;SETUP TIME PER
MOV #16666.,TICKUS ;TICK FOR 60HZ
5$: RTS PC ;RETURN
ST.PCLK:
BIT #SW05,C.SWR ;ALLOW SOFTWARE TIMEOUTS?
BNE 1$ ;NO--BRANCH
MOV #SRVCLK,@PKV ;SETUP THE KW11-P VECTOR
MOV #300,@PKV+2
MOV #1,@PKB ;COUNT ONE TICK
MOV #115,@PKCS ;'INT.EN.',COUNT DOWN', 'MODE 1 (REPEAT)',
;'LINE FREQ', AND 'RUN'
1$: RTS PC ;RETURN
ST.LCLK:
BIT #SW05,C.SWR ;ALLOW SOFTWARE TIMEOUTS?
BNE 1$ ;NO--BRANCH
MOV #SRVCLK,@LKV ;SETUP THE KW11-L VECTOR
MOV #300,@LKV+2
MOV #100,@LKS ;START THE KW11-L
1$: RTS PC ;RETURN
SRVCLK: MOV TICKMS,-(SP) ;TIME PER TICK IN MILLISECONDS
JSR PC,RMTMR ;COUNT THE ELAPSED TIME
RTI ;RETURN AFTER INTERRUPT

```

:THIS ROUTINE SETS UP DEFAULT PARAMETER VALUES WHEN THE PROGRAM IS
:STARTED OR WHEN THE VALUE OF BIT00 IN 'C.SWR' IS CHANGED.

```

225 ;CALL
226 ;
227 ; JSR PC,LODFLT
228 ; RETURN
229 LODFLT:
    026434 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
    026436 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
    026440 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
    026442 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
230 026444 012737 176777 001330 MOV #176777,TSTNMS ;SELECT TESTS 0-10, 12-17
231 026452 012737 000003 001332 MOV #3,TSTNMS+2 ;SELECT TESTS 20 & 21
232 026460 012700 002514 MOV #DFLT,R0 ;DEFAULT PARAMETERS POINTER
233 026464 012701 003116 MOV #PRM0,R1 ;TABLE POINTER
234 026470 010102 MOV R1,R2 ;STOP ADDRESS
235 026472 012021 1$: MOV (R0)+,(R1)+ ;MOVE DEFAULT PARAMETERS INTO
236 026474 020002 CMP R0,R2 ;RUN TIME TABLES ** DONE?
237 026476 103775 BLO 1$ ;NO--BRANCH
238 026500 012700 004160 MOV #PAT8,R0 ;PATO DEFAULTS TO PATTERN 8
239 026504 012701 003560 MOV #PATO,R1
240 026510 012021 2$: MOV (R0)+,(R1)+
241 026512 020027 004220 CMP R0,#PAT9
242 026516 103774 BLO 2$
243 026520 032737 000001 001314 BIT #BIT00,C.SWR ;16 BIT MODE ?
244 026526 001012 BNE 3$ ;BR IF 18 BIT MODE
245 026530 012737 000037 002456 MOV #31.,PRMLMT+24 ;SET 'FS' LIMIT TO 31.
246 026536 012737 000037 002460 MOV #31.,PRMLMT+26 ;SET 'LS' LIMIT TO 31.
247 026544 012737 160000 001450 MOV #-<256.*32.>,TRCKWC ;WORD COUNT FOR A 16 BIT TRACK
248 026552 000411 BR 4$ ;CONTINUE
249 026554 012737 000035 002456 3$: MOV #29.,PRMLMT+24 ;SET 'FS' LIMIT TO 29.
250 026562 012737 000035 002460 MOV #29.,PRMLMT+26 ;SET 'LS' LIMIT TO 29.
251 026570 012737 161000 001450 MOV #-<256.*30.>,TRCKWC ;WORD COUNT FOR AN 18 BIT TRACK
252 026576 012701 002362 4$: MOV #PRMPT,R1 ;ADDRESS OF PARAMETER POINTER TABLE
253 026602 005711 5$: TST (R1) ;END OF PARAMETER POINTER TABLE ?
254 026604 001425 BEQ 8$ ;BR IF YES
255 026606 032731 004000 BIT #BIT11,@(R1)+ ;IS 'LS' SELECTED AS A VARIABLE IN THIS TEST ?
256 026612 001773 BEQ 5$ ;BR IF NO
257 026614 016102 177776 MOV -(R1),R2 ;GET FIRST POSITION IN PARAMETER TABLE
258 026620 011246 MOV (R2),-(SP) ;AND SAVE VARIABLES BITS THAT ARE USED.
259 026622 012703 000014 MOV #12.,R3 ;POSITION OF 'LS' IN TEST PARAMATER TABLE
260 026626 006216 6$: ASR (SP) ;IS THIS PARAMETER A VARIABLE ?
261 026630 103002 BCC 7$ ;BR IF NO
262 026632 062702 000002 ADD #2,R2 ;YES, POINT TO NEXT PARAMETER IN TABLE
263 026636 005303 7$: DEC R3 ;AT 'LS' PARAMETER YET ?
264 026640 001372 BNE 6$ ;BR IF NO
265 026642 005726 TST (SP)+ ;ADJUST THE STACK
266 026644 021237 002456 CMP (R2),PRMLMT+24 ;IS 'LS' TOO LARGE FOR THE MODE SELECTED ?
267 026650 101754 BLOS 5$ ;BR IF NO
268 026652 013712 002456 MOV PRMLMT+24,(R2) ;RESET VALUE FOR MODE USED
269 026656 000751 BR 5$ ;CONTINUE
270 026660 8$: MOV (SP)+,R3 ;:POP STACK INTO R3
    026660 012603 MOV (SP)+,R2 ;:POP STACK INTO R2
    026662 012602 MOV (SP)+,R1 ;:POP STACK INTO R1
    026664 012601 MOV (SP)+,R0 ;:POP STACK INTO R0
    026666 012600 ;:POP STACK INTO R0
271 026670 000207 RTS PC ;RETURN

```

;THIS POUTINE FILLS THE PARAMETER TABLE THE CURRENT TEST.

```

274      :CALL
275      :      MOV      #TESTNUM,$STSTM ;LOAD THE TEST NUMBER
276      :      JSR      PC,LODPRM
277      :      RETURN
278
279      LODPRM:
          026672      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
          026674      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
          026676      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
          026700      010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
280      026702      005004      CLR      R4            ;CLEAR R4
281      026704      113704      001116      MOVB     $STSTM,R4     ;GET THE TEST NUMBER
282      026710      006304      ASL      R4            ;SETUP TO ADDRESS WORDS
283      026712      016401      002362      MOV      PRMPT(R4),R1  ;GET THE TEST'S PARAMETER TABLE ADDRESS
284      026716      012702      002322      MOV      #PRM,R2      ;PARAMETER EXECUTION TABLE
285      026722      005003      CLR      R3            ;R3 IS USED AS A COUNTER
286      026724      013704      001350      MOV      CHKDRV,R4     ;GET DRIVE ADDRESS
287      026730      012122      MOV      (R1)+,(R2)+   ;LOAD PARAMETER SPECIFIER
288      026732      006237      002322      1$:     ASR      PRM          ;IS THIS PARAMETER USED IN THE TEST ?
289      026736      103002      BCC      2$           ;BR IF NOT
290      026740      012122      MOV      (R1)+,(R2)+   ;LOAD THE VALUE
291      026742      000401      BR       3$           ;CONTINUE
292      026744      005022      2$:     CLR      (R2)+        ;CLEAR THE UNUSED PARAMETER LOCATION
293      026746      005203      3$:     INC      R3          ;COUNT THE POSITION IN THE OUTPUT TABLE
294      026750      022702      002350      CMP      #PAT+2,R2     ;FINISHED ?
295      026754      001437      BEQ      7$           ;BR IF YES
296      026756      022703      000007      CMP      #7,R3        ;DOING TRACK PARAMETERS ?
297      026762      001363      BNE      1$           ;BR IF NO
298      026764      122764      000007      036750  CMPB     #7,DRVTYP(R4) ;IS DEVICE AN RM05 ?
299      026772      001422      BEQ      6$           ;IF SO, OVERLAY FT, LT & IT WITH FT', LT' & IT'
300      026774      013737      002444      001372  MOV      PRMLT+12,LSTRK ;GET LAST TRACK FOR AN RM03:2
301      027002      062703      000003      ADD      #3,R3        ;ADJUST COUNTER
302      027006      006237      002322      ASR      PRM          ;COUNT THE PARAMETER
303      027012      103001      BCC      4$           ;BR IF FT' IS NOT USED
304      027014      005721      TST      (R1)+        ;MOVE THE INPUT POINTER
305      027016      006237      002322      4$:     ASR      PRM          ;COUNT THE PARAMETER
306      027022      103001      BCC      5$           ;BR IF LT' NOT USED
307      027024      005721      TST      (R1)+        ;MOVE THE INPUT POINTER
308      027026      006237      002322      5$:     ASR      PRM          ;COUNT THE PARAMETER
309      027032      103337      BCC      1$           ;BR IF IT' NOT USED
310      027034      005721      TST      (R1)+        ;MOVE THE INPUT POINTER
311      027036      000735      BR       1$           ;KEEP GOING
312      027040      013737      002452      001372  6$:     MOV      PRMLT+20,LSTRK ;GET LAST TRACK FOR AN RM05
313      027046      162702      000006      SUB      #6,R2        ;BACKUP THE OUTPUT POINTER
314      027052      000727      BR       1$           ;KEEP GOING
315      027054      7$:
          027054      012604      MOV      (SP)+,R4     ;;POP STACK INTO R4
          027056      012603      MOV      (SP)+,R3     ;;POP STACK INTO R3
          027060      012602      MOV      (SP)+,R2     ;;POP STACK INTO R2
          027062      012601      MOV      (SP)+,R1     ;;POP STACK INTO R1
316      027064      000207      RTS      PC           ;RETURN
317
318      ;THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND.
319      ;INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF "CONTROL SWITCH".
320      ;BIT07.
321      ;CALL
322      :      JSR      PC,LDCMD
    
```

```

323 ; RETURN
324
325 027066 032737 000200 001314 LDCMD: BIT #SW07,C.SWR ;DO EXPLICIT SEEKS?
326 027074 001007 BNE 1$ ;YES--BRANCH
327 027076 012737 000173 045500 MOV #READHD,DPB.B+2 ;NO--SET UP FOR READ HEADER AND
328 027104 012737 000173 045520 MOV #READHD,DPB.C+2 ;DATA COMMAND
329 027112 000406 BR 2$
330 027114 012737 000105 045500 1$: MOV #SEEK,DPB.B+2 ;SETUP FOR SEEK COMMAND
331 027122 012737 000105 045520 MOV #SEEK,DPB.C+2
332 027130 000207 2$: RTS PC
333
334 ;THIS ROUTINE WILL CALL THE RM05 DRIVER AND THEN WAIT ON THE FUNCTION
335 ;TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
336 ;CALL
337 ; FILL 'DPB' WITH COMMAND INFORMATION
338 ; JSR RO,CALL.A
339 ; RETURN
340
341 027132 005037 001222 CALL.A: CLR $ESCAPE ;NO ESCAPE ADDRESS
342 027136 004037 037654 JSR RO,RM05 ;CALL RM05 DRIVER
343 027142 045456 DPB.A
344 027144 000772 BR CALL.A
345 027146 005737 045474 1$: TST DPB.A+16 ;DONE?
346 027152 001775 BEQ 1$ ;NO--LOOP
347 027154 100042 BPL 4$ ;BRANCH IF NO ERROR
348 027156 012737 027252 001222 MOV #3$, $ESCAPE ;:ESCAPE TO 3$ ON ERROR
349 027164 013737 045470 001364 MOV DPB.A+12,CYL.DS ;CYLINDER
    027172 113737 045467 001370 MOV DPB.A+11,TRK.DS ;TRACK
    027200 113737 045466 001366 MOV DPB.A+10,SEC.DS ;SECTOR
    027206 012746 045474 MOV #DPB.A+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
    027212 004737 030526 JSR PC,ERINDX ;FORM DISPATCH INDEX
    027216 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
    027220 104041 EMT 41 ;NON-EXIST DRIVE
    027222 104042 EMT 42 ;PARITY ERROR
    027224 104043 EMT 43 ;UNSAFE ERROR
    027226 104044 EMT 44 ;NON-I/O ERROR
350 027230 000240 NOP ;TO SYNC THE CALLING SEQ OF ERINDX
351 027232 005737 045572 TST RM.REG+RMER1 ;ANY DRIVE ERROR
352 027236 001004 BNE 2$ ;BRANCH IF SO
353 027240 032737 100000 045620 BIT #BIT15,RM.REG+RMER2 ;BAD SPOT ERROR
354 027246 001005 BNE 4$ ;BRANCH IF SO
355 027250 2$: EMT 45
356 027252 013746 04 474 3$: MOV DPB.A+16,-(SP) ;STATUS WORD
357 027256 004737 J466 JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
358 027262 000200 4$: RTS RO ;RETURN
359
360 ;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND IF
361 ;THE COMMAND IS A READ HEADER AND DATA THE HEADER (CYLINDER, TRACK,
362 ;AND SECTOR) READ IS CHECKED FOR VALIDITY.
363 ;CALL
364 ; FILL DPB
365 ; JSR RO,CALL.B
366 ; RETURN
367
368 027264 005037 001222 CALL.B: CLR $ESCAPE ;NO ESCAPE ADDRESS
369 027270 004037 037654 JSR RO,RM05 ;CALL RM05 DRIVER
    
```

```

370 027274 045476          DPB.B
371 027276 000772          BR          CALL.B
372 027300 005737 045514  1$:  TST          DPB.B+16          ;DONE?
373 027304 001775          BEQ          1$          ;NO--BRANCH
374 027306 100047          BPL          4$          ;BRANCH IF NO ERROR
375 027310 012737 027414 001222  MOV          #3$, $ESCAPE ;ESCAPE TO 3$ ON ERROR
376 027316 013737 045510 001364  MOV          DPB.B+12,CYL.DS ;CYLINDER
    027324 113737 045507 001370  MOVB         DPB.B+11,TRK.DS ;TRACK
    027332 113737 045506 001366  MOVB         DPB.B+10,SEC.DS ;SECTOR
    027340 012746 045514          MOV          #DPB.B+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
    027344 004737 030526          JSR          PC,ERINDX      ;FORM DISPATCH INDEX
    027350 062607          ADD          (SP)+,PC      ;REPORT PROPER ERROR
    027352 104041          EMT          41          ;NON-EXIST DRIVE
    027354 104042          EMT          42          ;PARITY ERROR
    027356 104043          EMT          43          ;UNSAFE ERROR
    027360 104044          EMT          44          ;NON-I/O ERROR
377 027362 000240          NOP          ;TO SYNC THE CALLING SEQ OF ERINDX: RT.
378 027364 005737 045572          TST          RM.REG+RMER1 ;DRIVE ERROR ?
379 027370 001404          BEQ          2$          ;BR IF NOT
380 027372 032737 177677 045572  BIT          #^C100,RM.REG+RMER1 ;SEE IF ONLY 'HCE' SET
381 027400 001412          BEQ          4$          ;BR IF IT IS
382 027402          ;
383 027402 032737 100000 045620  2$:  BIT          #BIT15,RM.REG+RMER2 ;BSE ERROR
384 027410 001025          BNE          6$          ;BRANCH IF SO
385 027412 104045          EMT          45
386 027414 013746 045514  3$:  MOV          DPB.B+16,-(SP) ;STATUS WORD
387 027420 004737 030466          JSR          PC,LOP.CK     ;SEE IF LOOP, ABORT, OR CONTINUE
388 027424 000410          BR          5$          ;CHECK FOR STALL
389 027426 123727 045500 000173  4$:  CMPB         DPB.B+2,#READHD ;DOING IMPLIED SEEKS?
390 027434 001004          BNE          5$          ;NO--BRANCH
391 027436 004037 030746          JSR          RO,VERIFY     ;YES--GO CHECK THE DATA
392 027442 045506          DPB.B+10
393 027444 000407          BR          6$          ;ERROR DURING VERIFY
394 027446 032737 040000 001314  5$:  BIT          #SW14,C.SWR   ;STALL?
395 027454 001403          BEQ          6$          ;NO--BRANCH
396 027456 004037 030664          JSR          RO,STALL     ;YES--CALL STALL ROUTINE
397 027462 001452          .WORD       STALL1       ;STALL TIME POINTER
398 027464 000200          6$:  RTS          RO          ;RETURN
399
400          ;THIS ROUTINE IS THE SAME AS "CALL.B" EXCEPT FOR THE DPB USED.
401          ;CALL
402          ;
403          ;   FILL DPB
404          ;   JSR          RO,CALL.C
405          ;   RETURN
406 027466 005037 001222  CALL.C: CLR          $ESCAPE ;NO ESCAPE ADDRESS
407 027472 004037 037654          JSR          RO,RM05      ;CALL RM05 DRIVER
408 027476 045516          DPB.C
409 027500 000772          BR          CALL.C
410 027502 005737 045534  1$:  TST          DPB.C+16          ;DONE?
411 027506 001775          BEQ          1$          ;NO--LOOP
412 027510 100047          BPL          4$          ;YES--BRANCH IF NO ERROR
413 027512 012737 027616 001222  MOV          #3$, $ESCAPE ;ESCAPE TO 3$ ON ERROR
414 027520 013737 045530 001364  MOV          DPB.C+12,CYL.DS ;CYLINDER
    027526 113737 045527 001370  MOVB         DPB.C+11,TRK.DS ;TRACK
    027534 113737 045526 001366  MOVB         DPB.C+10,SEC.DS ;SECTOR
    027542 012746 045534          MOV          #DPB.C+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
    
```

```

027546 004737 030526 JSR PC,ERINDX ;FORM DISPATCH INDEX
027552 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
027554 104041 EMT 41 ;NON-EXIST DRIVE
027556 104042 EMT 42 ;PARITY ERROR
027560 104043 EMT 43 ;UNSAFE ERRGR
027562 104044 EMT 44 ;NON-I/O ERROR
415 027564 000240 NOP ;TO SYNC THE CALLING SEQ OF ERINDX: RT.
416 027566 005737 045572 TST RM.REG+RMER1 ;DRIVE ERROR ?
417 027572 001404 BEQ 2$ ;BR IF NOT
418 027574 032737 177677 045572 BIT #^C100,RM.REG+RMER1 ;SEE IF ONLY 'MCE' SET
419 027602 001412 BEQ 4$ ;BR IF IT IS
420 027604 2$:
421 027604 032737 100000 045620 BIT #B1115,RM.REG+RMER2 ;BSE ERROR ONLY ?
422 027612 001025 BNE 6$ ;BRANCH IF SO
423 027614 104045 EMT 45
424 027616 013746 045534 3$: MOV DPB.C+16,-(SP) ;STATUS WORD
425 027622 004737 030466 JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
426 027626 000410 BR 5$
427 027630 123727 045520 000173 4$: CMPB DPB.C+2,#READHD ;DOING IMPLIED SEEK?
428 027636 001004 BNE 5$ ;NO--EXIT
429 027640 004037 030746 JSR RO,VERIFY ;YES--CHECK THE DATA
430 027644 045526 DPB.C+10
431 027646 000407 BR 6$ ;ERROR DURING VERIFY
432 027650 032737 040000 001314 5$: BIT #SW14,C.SWR ;STALL?
433 027656 001403 BEQ 6$ ;NO--BRANCH
434 027660 004037 030664 JSR RO,STALL ;YES--CALL STALL ROUTINE
435 027664 001452 .WORD STALL1 ;STALL TIME POINTER
436 027666 000200 6$: RTS RO

```

;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND
 ;ON AN ERROR LOCATION "ERR.CT" IS EXAMINED. IF ERR.CT IS EQUAL TO
 ;SERFLG EXIT IS TO THE NEXT TEST.

```

;CALL
;
; FILL DPB
; JSR RO,DRVCAL
;
; RETURN

```

```

447 027670 005037 001222 DRVCAL: CLR $ESCAPE ;NO ESCAPE ADDRESS
448 027674 005037 001432 CLR WCEFLG ;CLEAR WRITE CHECK ERROR FLAG
449 027700 004037 037654 JSR RO,RM05 ;CALL RM05 DRIVER
450 027704 045536 DTADPB
451 027706 000770 BR DRVCAL
452
453 027710 005737 045554 DRVCL1: TST DTADPB+16 ;DONE
454 027714 001775 BEQ DRVCL1 ;NO--LOOP
455 027716 100402 BMI 1$ ;BR IF ERRORS
456 027720 000137 030446 JMP 14$ ;NO ERRORS
457 027724
458 027724 012737 030032 001222 1$: MOV #3$, $ESCAPE ;;ESCAPE TO 3$ ON ERROR
027732 013737 045550 001364 MOV DTADPB+12,CYL.DS ;CYLINDER
027740 113737 045547 001370 MOV DTADPB+11,TRK.DS ;TRACK
027746 113737 045546 001366 MOV DTADPB+10,SEC.DS ;SECTOR
027754 012746 045554 MOV #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
027760 004737 030526 JSR PC,ERINDX ;FORM DISPATCH INDEX
027764 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
027766 104041 EMT 41 ;NON-EXIST DRIVE

```



```

INTEGER DIVIDE ROUTINE
027770 104042
027772 104043
027774 104044
459 027776 000240
460 030000 005737 045572
461 030004 001011
462 030006 032737 100000 045620
463 030014 001405
464 030016 012737 177777 001464
465 030024 000137 030446
466 030030
030030 104045
467 030032 122737 000020 001116 3$:
468 030040 001170
469 030042 013746 045554
470 030046 004737 030466
471 030052 122737 000151 045540
472 030060 001164
473 030062 032737 040000 045566
474 030070 001560
475 030072 032777 000020 151054
476 030100 001154
477 030102 112737 000161 045540
478 030110 005037 001222
479 030114 004037 037654
480 030120 045536
481 030122 000240
482 030124 005737 045554 4$:
483 030130 001775
484 030132 100043
485 030134 012737 030422 001222
486 030142 013737 045550 001364
030150 113737 045547 001370
030156 113737 045546 001366
030164 012746 045554
030170 004737 030526
030174 062607
030176 104041
030200 104042
030202 104043
030204 104044
487 030206 000240
488 030210 005737 045572
489 030214 001011
490 030216 032737 100000 045620
491 030224 001405
492 030226 012737 177777 001464
493 030234 000137 030446
494 030240
030240 104045
495 030242 112737 000151 045540 5$:
496 030250 004037 037654 6$:
497 030254 045536
498 030256 000240
499 030260 005737 045554 7$:
500 030264 001775
501 030266 100410
EMT 42 ;PARITY ERROR
EMT 43 ;UNSAFE ERROR
EMT 44 ;NON-I/O ERROR
NOP ;TO SYN THE CALLING SEQ OF THE ERINDX:
TST RM.REG+RMER1 ;ANY DRIVE ERROR ?
BNE 2$ ;REPORT THE I/O ERROR IF SO
BIT #BIT15,RM.REG+RMER2 ;BAD SPOT ERROR ?
BEQ 2$ ;BRANCH IF NOT
MOV #-1,BASFLG ;SET BAD SECTOR ENCOUNTER FLAG
JMP 14$ ;OTHERWISE ,DON'T REPORT THE BSE

EMT 45
CMPB #20,$TSTNM ;TEST 20?
BNE 11$ ;NO--BRANCH
MOV DTADPB+16,-(SP) ;STATUS WORD
JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
CMPB #WRCKD,DTADPB+2 ;DOING A WRITE CHECK?
BNE 12$ ;NO--BRANCH
BIT #BIT14,RM.REG+10 ;IS 'WCE'=1?
BEQ 12$ ;NO--BRANCH
BIT #SW04,@SWR ;INHIBIT WRITES?
BNE 12$ ;YES--BRANCH
MOVB #WRITE,DTADPB+2 ;SETUP FOR A WRITE
CLR $ESCAPE ;NO ESCAPE ADDRESS
JSR R0,RM05 ;DO THE WRITE

DTADPB
NOP
TST DTADPB+16 ;DONE?
BEQ 4$ ;NO--LOOP
BPL 6$ ;YES--BRANCH IF NO ERROR
MOV #11,$ESCAPE ;ESCAPE TO 11$ ON ERROR
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
EMT 42 ;PARITY ERROR
EMT 43 ;UNSAFE ERROR
EMT 44 ;NON-I/O ERROR
NOP ;TO SYN THE CALLING SEQ OF ERINDX
TST RM.REG+RMER1 ;ANY DRIVE ERROR ?
BNE 5$ ;BRANCH IF SO
BIT #BIT15,RM.REG+RMER2 ;BAD SOPT ERROR
BEQ 5$ ;BRANCH IF NOT
MOV #-1,BASFLG ;SET BAD SECTOR ENCOUNTER FLAG
JMP 14$ ;EXIT

EMT 45
MOVB #WRCKD,DTADPB+2 ;COMMAND=WRITE CHECK DATA
JSR R0,RM05 ;DO THE WRITE CHECK

DTADPB
NOP
TST DTADPB+16 ;DONE?
BEQ 7$ ;NO--LOOP
BMI 9$ ;YES--BRANCH IF ERROR

```

```
502 030270 004037 037654 JSR RO,RM05 ;DO A 2ND WRITE CHECK
503 030274 045536 DTADPB
504 030276 000240 NOP
505 030300 005737 045554 8$: TST DTADPB+16 ;DONE?
506 030304 001775 BEQ 8$ ;NO--LOOP
507 030306 100057 BPL 14$ ;YES--BRANCH IF NO ERROR
508 030310 012737 000001 001432 9$: MOV #1,WCEFLG ;SET THE WRITE CHECK ERROR FLAG
509 030316 012737 030422 001222 MOV #11$, $ESCAPE ;ESCAPE TO 11$ ON ERROR
510 030324 013737 045550 001364 MOV DTADPB+12,CYL.DS ;CYLINDER
030332 113737 045547 001370 MOV DTADPB+11,TRK.DS ;TRACK
030340 113737 045546 001366 MOV DTADPB+10,SEC.DS ;SECTOR
030346 012746 045554 MOV #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
030352 004737 030526 JSR PC,ERINDX ;FORM DISPATCH INDEX
030356 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
030360 104041 EMT 41 ;NON-EXIST DRIVE
030362 104042 EMT 42 ;PARITY ERROR
030364 104043 EMT 43 ;UNSAFE ERROR
030366 104044 EMT 44 ;NON-I/O ERROR
511 030370 000240 NOP ;SO SYN THE CALLING SEQ OF ERINDX:
512 030372 005737 045572 TST RM.REG+RMER1 ;ANY DRIVE ERROR
513 030376 001010 BNE 10$ ;BRANCH IF SO
514 030400 032737 100000 045620 BIT #BIT15,RM.REG+RMER2 ;BAD SOPT ERROR ?
515 030406 001404 BEQ 10$ ;BRANCH IF NOT
516 030410 012737 177777 001464 MOV #-1,BASFLG ;SET BAD SECTOR ENCOUNTER FLAG
517 030416 0C0413 BR 14$ ;OTHERWISE EXIT
518 030420 10$: EMT 46
030420 104046 EMT 46
519 030422 013746 045554 11$: MOV DTADPB+16,-(SP) ;STATUS WORD
520 030426 004737 030466 JSR PC,LOP.CK ;SEE IF LOOP, ABORT, OR CONTINUE
521 030432 123737 001462 001117 12$: CMPB ERR.CT,$ERFLG ;GO TO NEXT TEST?
522 030440 101002 BHI 14$ ;NO--BRANCH
523 030442 013700 001346 13$: MOV BYPASS,RO ;YES--GET EXIT ADDRESS
524 030446 032737 040000 001314 14$: BIT #SW14,C.SWR ;STALL?
525 030454 001403 BEQ 15$ ;NO--BRANCH
526 030456 004037 030664 JSR RO,STALL ;YES--CALL STALL ROUTINE
527 030462 001454 .WORD STALL2 ;STALL TIME POINTER
528 030464 000200 15$: RTS RO
529
530 ;THIS SUBROUTINE CHECK FOR LOOP, ABORT, OR CONTINUE SWITCHES AFTER
531 ;ERRORS 41, 42, 43, 44, 45, AND 46.
532 ;CALL
533 ; MOV DTA+16,-(SP) ;STATUS WORD FROM DPB IN USE
534 ; JSR PC,LOP.CK
535 ; RETURN
536
537 030466 032777 001000 150460 LOP.CK: BIT #SW9,@SWR ;LOOP ON ERROR
538 030474 001402 BEQ 1$ ;BR IF NOT
539 030476 000177 150422 JMP @ $LPERR ;START AT THE LOOP ADDRESS
540 030502 005037 001222 1$: CLR $ESCAPE ;CLEAR ERROR ESCAPE FLAG
541 030506 032766 072006 000002 BIT #BIT14!BIT13!BIT12!BIT10!BIT02!BIT01,2(SP) ;CHECK ERROR TYPE
542 030514 001402 BEQ 2$ ;BR IF DRIVE NOT OFFLINE, UNLOADED, OR
543 ;PERSISTENT UNSAFE OR FATAL MASSBUS PARITY
544 030516 000137 021020 JMP $EOP ;TERMINATE DRIVE
545 030522 012616 2$: MOV (SP)+,(SP) ;ADJUST RETURN ADDRESS
546 030524 000207 RTS PC
547
548 ;THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
```

```

549      ;TO THE PROPER ERROR CALL. THE INDEX IS FORMED BY EXAMINING
550      ;THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
551      ;INDEX          STATUS/ERROR
552      ;-----
553      ; 0 BIT14!BIT13!BIT08!BIT01
554      ; 2 BIT11!BIT10!BIT02
555      ; 4 BIT12!BIT04
556      ; 6 BIT05!BIT03!<BIT09 & COMMAND=NON-I/O>
557      ;10 BIT06!<BIT09 & COMMAND=I/O>
558      ;CALL
559      ;   JSR      #DPB+16,-(SP)      ;ADDRESS OF STATUS/ERROR INDICATOR
560      ;   JSR      PC,ERINDX         ;FORM INDEX
561      ;   RETURN                      ;INDEX IS ON THE STACK
562
563      030526 010046
564      030530 010146
565      030532 016600 000006
566      030536 011037 001354
567      030542 005001
568      030544 032710
569      030546 020402
570      030550 001037
571      030552 032710
572      030554 006004
573      030556 001033
574      030560 032710
575      030562 050020
576      030564 001027
577      030566 032710
578      030570 000050
579      030572 001023
580      030574 032710
581      030576 000100
582      030600 001017
583      030602 032710
584      030604 001000
585      030606 001420
586      030610 122760 000150 177762
587      030616 003011
588      030620 005737 045572
589      030624 001005
590      030626 032737 100000 045620
591      030634 001401
592      030636 005201
593      030640 005201
594      030642 005201
595      030644 005201
596      030646 005201
597      030650 006301
598      030652 010166 000006
599      030656 012601
600      030660 012600
601      030662 000207
602
603
604
605

```

```

;-----
0 BIT14!BIT13!BIT08!BIT01
2 BIT11!BIT10!BIT02
4 BIT12!BIT04
6 BIT05!BIT03!<BIT09 & COMMAND=NON-I/O>
10 BIT06!<BIT09 & COMMAND=I/O>
CALL
;   JSR      #DPB+16,-(SP)      ;ADDRESS OF STATUS/ERROR INDICATOR
;   JSR      PC,ERINDX         ;FORM INDEX
;   RETURN                      ;INDEX IS ON THE STACK
ERINDX: MOV      R0,-(SP)        ;SAVE R0
        MOV      R1,-(SP)        ;SAVE R1
        MOV      6(SP),R0        ;GET STATUS/ERROR INDICATOR POINTER
        MOV      (R0),SVSTAT     ;SAVE THE STATUS/ERROR INDICATOR
        CLR      R1              ;START INDEX AT ZERO
        BIT      (PC)+,(R0)      ;FORM INDEX OF 0?
        .WORD   BIT13!BIT08!BIT01
        BNE     5$              ;YES--BRANCH
        BIT      (PC)+,(R0)      ;FORM PARITY ERROR OR PORT REQUEST INDEX (2)?
        .WORD   BIT11!BIT10!BIT02
        BNE     4$              ;YES--BRANCH
        BIT      (PC)+,(R0)      ;FORM UNSAFE INDEX (4)?
        .WORD   BIT14!BIT12!BIT04
        BNE     3$              ;YES--BRANCH
        BIT      (PC)+,(R0)      ;FORM NON-I/O ERROR INDEX (6)?
        .WORD   BIT05!BIT03
        BNE     2$              ;YES--BRANCH
        BIT      (PC)+,(R0)      ;FORM I/O ERROR INDEX (10)?
        .WORD   BIT06
        BNE     1$              ;YES--BRANCH
        BIT      (PC)+,(R0)      ;SOFTWARE TIMEOUT?
        .WORD   BIT09
        BEQ     5$              ;NO--FORM INDEX OF 0
        CMPB   #150,-16(R0)     ;YES--I/O?
        BGT     2$              ;NO--BRANCH
        TST    RM.REG+RMER1     ;ANY DRIVE ERROR ?
        BNE     1$              ;BRANCH IF SO
        BIT    #BIT15,RM.REG+RMER2 ;BSE ERROR
        BEQ     1$              ;BRANCH IF NOT
        INC    R1              ;SKIP , NOT REPORT BSE ERROR
        1$: INC    R1           ;INDEX=10---ERROR=45 OR 46
        2$: INC    R1           ;INDEX=6---ERROR=44
        3$: INC    R1           ;INDEX=4---ERROR=43
        4$: INC    R1           ;INDEX=2---ERROR=42
        5$: ASL    R1           ;INDEX=0---ERROR=41
        MOV    R1,6(SP)        ;RETURN INDEX TO USER
        MOV    (SP)+,R1        ;RESTORE R1
        MOV    (SP)+,R0        ;RESTORE R0
        RTS    PC             ;RETURN FROM CALL

```

;THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
 ;AMOUNT OF TIME IF BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
 ;IF BIT 13 OF C.SWR = 1.

```

606 ;STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0 - 7, AND STALL2
607 ;CONTAINS THE TIME FOR TESTS 16-21.
608 ;CALL
609 ;
610 ; JSR RO,STALL
611 ; TIME POINTER ;WHERE TO FIND THE STALL TIME
612 030664 013046 STALL: MOV @ (RO)+, -(SP) ;PICKUP STALL TIME
613 030666 032737 020000 001314 BIT #SW13,C.SWR ;USE A RANDOM TIME?
614 030674 001406 BEQ 1$ ;NO--BRANCH
615 030676 004737 025550 JSR PC,$RAND ;YES--FORM RANDOM NUMBER
616 030702 013716 025650 MOV $LONUM,(SP) ;AND USE IT FOR THE STALL TIME
617 030706 042716 177700 BIC #^C77,(SP) ;BUT NEVER > 64 MILLISECONDS
618 030712 005046 1$: CLR -(SP) ;CLEAR TEMP. LOCATION
619 030714 162766 000001 000002 2$: SUB #1,2(SP) ;MORE STALL REQUIRED?
620 030722 103407 BLO 4$ ;NO--BRANCH
621 030724 012716 000144 MOV #100.,(SP) ;STALL FOR ABOUT 1 MILLISECOND
622 030730 005700 3$: TST RO ;NOP TO KILL TIME
623 030732 005366 000000 DEC 0(SP) ;COUNT
624 030736 001374 BNE 3$ ;LOOP IF MORE COUNTS NEEDED
625 030740 000765 BR 2$
626 030742 022626 4$: CMP (SP)+,(SP)+ ;CLEAN OFF THE STACK
627 030744 000200 RTS RO ;EXIT
628
629 ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEKS
630 ;CALL
631 ;
632 ; JSR RO,VERIFY
633 ; ADR POINTER ;ADDRESS OF DPB+10 (SECTOR NUMBER)
634 ; RETURN
635 030746 010146 VERIFY: MOV R1,-(SP) ;SAVE R1
636 030750 012001 MOV (RO)+,R1 ;GET ADDRESS OF DPB+10
637 030752 042737 150000 052776 BIC #150000,BUFFER ;STRIP FORMAT AND BAD SECTOR BITS FROM CYLINDER NUMBER
638 030760 023761 052776 000002 CMP BUFFER,2(R1) ;CYLINDER NUMBER OK?
639 030766 001003 BNE 1$ ;NO--BRANCH
640 030770 023711 053000 CMP BUFFER+2,(R1) ;YES--HOW ABOUT TRACK/SECTOR?
641 030774 001441 BEQ 3$ ;BRANCH IF GOOD
642 030776 013737 052776 001356 1$: MOV BUFFER,CYL.RD ;SAVE THE EXPECTED AND THE
643 031004 113737 053001 001360 MOVB BUFFER+3,TRK.RD ;RECIEVED CYLINDER, TRACK,
644 031012 113737 053000 001362 MOVB BUFFER+2,SEC.RD ;AND SECTOR
645 031020 112137 001366 MOVB (R1)+,SEC.DS
646 031024 112137 001370 MOVB (R1)+,TRK.DS
647 031030 011137 001364 MOV (R1),CYL.DS
648 031034 012737 031046 001222 MOV #2$,$ESCAPE ;;ESCAPE TO 2$ ON ERROR
649 031042 005740 TST -(RO) ;MAKE IT TEST PC+4
650 031044 104012 EMT 12
651 031046 012737 000107 045460 2$: MOV #RECAL,DPB.A+2 ;LOAD RECALIBRATE ORDER CODE
652 031054 004037 027132 JSR RO,CALL.A ;GO EXECUTE THE COMMAND
653 031060 005037 001222 CLR $ESCAPE ;CLEAR ERROR ESCAPE FLAG
654 031064 032777 001000 150062 BIT #SW9,@SWR ;LOOP ON ERROR ?
655 031072 001404 BEQ 4$ ;BR IF NOT
656 031074 000177 150024 JMP @LPERR ;RETURN TO ERROR LOOP ADDRESS
657 031100 062700 000002 3$: ADD #2,RO ;INCREMENT RETURN ADDRESS
658 031104 012601 4$: MOV (SP)+,R1 ;RESTORE R1
659 031106 000200 RTS RO ;EXIT
660
661 ;THIS ROUTINE WILL PERFORM A 'MASSBUS' INIT. FOLLOWED BY
662 ;A 'RECALIBRATE' ON THE DRIVE UNDER TEST.
    
```

```

663 ;NOTE: THIS ROUTINE DESTROYS R1 AND R4
664 ;CALL
665 ; JSR RO,SRCH00 ;DO A MASSBUS INIT. AND RECAL
666 ; RETURN1 ;RETURN HERE IF NO ERROR
667 ; RETURN2 ;RETURN HERE ON ERROR
668
669 031110 005001 SRCH00: CLR R1 ;INCASE OF ERROR (TYPTIM)
670 031112 005037 177776 CLR PS
671 031116 012777 041740 005744 MOV #ISR,@RMVEC ;SETUP INTERRUPT VECTOR
672 031124 013704 037066 MOV RMADR,R4 ;PICKUP ADDRESS OF RMCS1
673 031130 012764 000040 000010 MOV #CLR,RMCS2(R4) ;MASSBUS INIT.
674 031136 005037 045546 CLR DTADPB+10 ;TRACK=0; SECTOR=0
675 031142 005037 045550 CLR DTADPB+12 ;CYLINDER =0
676 031146 012737 000107 045540 MOV #RECAL,DTADPB+2 ;COMMAND = RECALIBRATE
677 031154 005037 001222 CLR $ESCAPE ;NO ESCAPE ADDRESS
678 031160 004037 037654 JSR RO,RM05 ;CALL THE DRIVER
679 031164 045536 DTADPB ;DPB POINTER
680 031166 000440 BR 4$ ;QUEUE IS FULL
681 031170 005737 045554 1$: TST DTADPB+16 ;WAIT ON DONE
682 031174 001775 BEQ 1$
683 031176 100030 BPL 3$ ;TAKE NORMAL EXIT IF NO ERROR
684 031200 012737 031254 001222 MOV #2,$ESCAPE ;;ESCAPE TO 2$ ON ERROR
685 031206 013737 045550 001364 MOV DTADPB+12,CYL.DS ;CYLINDER
031214 113737 045547 001370 MOV DTADPB+11,TRK.DS ;TRACK
031222 113737 045546 001366 MOV DTADPB+10,SEC.DS ;SECTOR
031230 012746 045554 MOV #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
031234 004737 030526 JSR PC,ERINDX ;FORM DISPATCH INDEX
031240 062607 ADD (SP)+,PC ;REPORT PROPER ERROR
031242 104041 EMT 41 ;NON-EXIST DRIVE
031244 104042 EMT 42 ;PARITY ERROR
031246 104043 EMT 43 ;UNSAFE ERROR
031250 104044 EMT 44 ;NON-I/O ERROR
031252 104045 EMT 45 ;I/O ERROR
686 031254 005720 2$: TST (R0)+ ;ADJUST FOR ERROR EXIT
687 031256 000404 BR 4$ ;GO TO THE EXIT
688 031260 005064 000006 3$: CLR RMDA(R4) ;TRACK AND SECTOR = 0
689 031264 005064 000034 CLR RMDC(R4) ;CYLINDER = 0
690 031270 000200 4$: RTS R0 ;RETURN
691
692 ;THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS & THE SERVO SETTLE DOWN TEST
693
694 031272 000002 DORT1: RTI ;RETURN FROM INTERRUPT
695
696 ;THIS ROUTINE WILL INITIALIZE THE TIMERS USED BY THE 'TIMING ROUTINES
697 ;CALL
698 ; JSR PC,STRTMR
699 ; RETURN
700
701 031274 104412 STRTMR: SAVREG ;SAVE R0-R5
702 031276 012700 001374 MOV #TIM.UP,R0 ;START AT TIM.UP
703 031302 005020 1$: CLR (R0)+ ;CLEAR THE TIME TABLES
704 031304 020027 001430 CMP R0,#TIM.PT ;DONE?
705 031310 103774 BLO 1$ ;NO--BRANCH
706 031312 012710 052776 MOV #BUFFER,(R0) ;SETUP POINTER
707 031316 012737 077777 001374 MOV #^CBIT15,TIM.UP ;SET MINIMUM TIME TO MAXIMUM
708 031324 012737 077777 001412 MOV #^CBIT15,TIM.DN ;POSITIVE NUMBER
709 031332 104413 RESREG ;RESTORE R0-R5
    
```

```

710 031334 000207          RTS      PC          ;RETURN
711
712          ;THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
713          ;MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
714          ;NOTE: THIS ROUTINE DESTROYS R2
715          ;CALL
716          ;:
717          ;:      MOV      #TP,R3          ;PARAMETER POINTER
718          ;:      MOV      FLAG,R5        ;FLAG=0=COUNT UP
719          ;:      JSR      PC,COUNT      ;FLAG=-1=COUNT DOWN
720          ;:
721          ;:      RETURN
722 031336 012702 001374    COUNT:  MOV      #TIM.UP,R2      ;PICKUP THE 'UP' POINTER
723 031342 005705          TST      R5          ;USE IT?
724 031344 001402          BEQ      1$          ;YES--BRANCH
725 031346 012702 001412    MOV      #TIM.DN,R2      ;NO--PICKUP 'DOWN' POINTER
726 031352 027722 150130    1$:    CMP      @PKC,(R2)+    ;LESS THAN PREVIOUS LOW?
727 031356 002003          BGE      2$          ;NO--BRANCH
728 031360 017762 150122 177776    MOV      @PKC,-2(R2)    ;YES--SAVE IT
729 031366 027763 150114 000004    2$:    CMP      @PKC,4(R3)    ;LESS THAN THE LOW LIMIT?
730 031374 002001          BGE      3$          ;NO--BRANCH
731 031376 005212          INC      (R2)         ;YES--COUNT IT
732 031400 005722          3$:    TST      (R2)+         ;ADVANCE THE POINTER
733 031402 027722 150100    CMP      @PKC,(R2)+    ;GREATER THAN PREVIOUS HIGH?
734 031406 003403          BLE      4$          ;NO--BRANCH
735 031410 017762 150072 177776    MOV      @PKC,-2(R2)    ;YES--SAVE IT
736 031416 027763 150064 000006    4$:    CMP      @PKC,6(R3)    ;GREATER THAN THE HIGH LIMIT?
737 031424 003401          BLE      5$          ;NO--BRANCH
738 031426 005212          INC      (R2)         ;YES--COUNT IT
739 031430 005722          5$:    TST      (R2)+         ;ADVANCE THE POINTER
740 031432 067722 150050    ADD      @PKC,(R2)+    ;ADD THIS COUNT TO THE TOTAL
741 031436 005522          ADC      (R2)+
742 031440 005212          INC      (R2)         ;COUNT THIS READING
743 031442 022737 061326 001430    CMP      #BUFFER+<4*822.>,TIM.PT ;SAVE THIS COUNT?
744 031450 101406          BLOS    6$          ;NO--BRANCH
745 031452 017777 150030 147750    MOV      @PKC,@TIM.PT  ;YES--WELL SAVE IT THEN
746 031460 062737 000002 001430    ADD      #2,TIM.PT     ;ADVANCE THE POINTER
747 031466 000207          6$:    RTS      PC          ;RETURN
748
749          ;THIS ROUTINE PRINTS THE SPEC OF ALL TIMING TESTS
750          ;CALL
751          ;:
752          ;:      JSR      RO,SPTYP      ;
753          ;:      TABLE ADDRESS
754          ;:
755          ;:      TABLE: .WORD   ASCIZ MESSAGE POINTER
756          ;:      .WORD   MIN VALUE
757          ;:      .WORD   MAX VALUE
758 031470 012002          SPTYP:  MOV      (R0)+,R2      ;THE TABLE ADDRESS
759 031472 032777 000100 147454    BIT      #SW06,@SWR    ;ALLOW PRINT
760 031500 001035          BNE     3$          ;EXIT IF NOT
761 031502 104401 001231    TYPE    ,%CRLF
762 031506 104401 001231    TYPE    ,%CRLF
763 031512 012237 031520    MOV      (R2)+,1$
764 031516 104401          TYPE
765 031520 000000          1$:    .WORD   0
766 031522 012246          MOV      (R2)+,-(SP)  ;LOAD MIN VALUE
    
```

```

767 031524 001410          BEQ      2$          ;SKIP IF MIN VALUE IS 0
768 031526 104401 046500    TYPE     ,MSGMIN
769 031532 004737 025260    JSR      PC,$SB2D    ;CONVERT TO DECIMAL
770 031536 004737 025510    JSR      PC,$SUPRS   ;TYPE IT
771 031542 104401 046522    TYPE     ,MSGOUS     ;O US
772 031546 104401 046506    2$:     TYPE     ,MSGMAX
773 031552 011246          MOV      (R2),-(SP)  ;MAXIMUM VALUE
774 031554 004737 025260    JSR      PC,$SB2D
775 031560 004737 025510    JSR      PC,$SUPRS
776 031564 104401 046522    TYPE     ,MSGOUS
777 031570 104401 001231    TYPE     ,$CRLF     ;CR-LF
778 031574 000200          3$:     RTS      R0
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797 031576 012002          TYPTIM: MOV      (R0)+,R2    ;PICKUP THE TABLE POINTER
798 031600 032777 000100 147346 BIT      #SW06,$SWR   ;INHIBIT TIME REPORTS?
799 031606 001160          BNE     9$          ;YES--BRANCH
800 031610 012237 031630    MOV      (R2)+,2$    ;ADDRESS OF MESSAGE NUMBER 1
801 031614 012205          MOV      (R2)+,R5    ;ADDRESS OF MESSAGE NUMBER 2
802 031616 012203          MOV      (R2)+,R3    ;PICKUP THE LOW LIMIT
803 031620 011202          MOV      (R2),R2     ;AND THE HIGH LIMIT
804 031622 012704 001374    MOV      #TIM.UP,R4  ;PARAMETER POINTER
805 031626 104401          1$:     TYPE     ;TYPE THE MESSAGE
806 031630 000000          2$:     .WORD    0     ;ASCIZ MESSAGE POINTER GOES HERE
807 031632 005764 000014    TST     14(R4)      ;DID ANY COUNTS OCCUR?
808 031636 001542          BEQ     8$          ;NO--BRANCH
809 031640 104401 046500    TYPE     ,MSGMIN    ;'MIN='
810 031644 012446          MOV      (R4)+,-(SP) ;PUT (R4)+ ON THE STACK
      031646 004737 025260    JSR      PC,$SB2D   ;CHANGE TO DECIMAL ASCIZ
      031652 004737 025510    JSR      PC,$SUPRS  ;TYPE WITHOUT LEADING ZEROS
811 031656 104401 046522    TYPE     ,MSGOUS    ;'O US'
812 031662 005724          TST     (R4)+      ;ANY SEEKS BELOW THE LOW LIMIT
813 031664 001421          BEQ     3$          ;NO--BRANCH
814 031666 104401 046657    TYPE     ,BLNKS2    ;TYPE 2 SPACES
815 031672 016446 177776    MOV      -2(R4),-(SP);PUT -2(R4) ON THE STACK
      031676 004737 025260    JSR      PC,$SB2D   ;CHANGE TO DECIMAL ASCIZ
      031702 004737 025510    JSR      PC,$SUPRS  ;TYPE WITHOUT LEADING ZEROS
816 031706 104401 046527    TYPE     ,M BELOW  ;'BELOW THE MINIMUM OF'
817 031712 010546          MOV      R3,-(SP)  ;PUT R3 ON THE STACK
      031714 004737 025260    JSR      PC,$SB2D   ;CHANGE TO DECIMAL ASCIZ
      031720 004737 025510    JSR      PC,$SUPRS  ;TYPE WITHOUT LEADING ZEROS
    
```

```

;:THIS ROUTINE IS USED TO TYPE THE MINIMUM,
;:MAXIMUM, AND AVERAGE TIMES FOR THE TIMING TESTS
;:IT WILL ALSO CHECK THE TIMES TO ENSURE
;:THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.
;:NOTE: THIS ROUTINE DESTROYS R2-R5
;:CALL
    
```

```

;:      JSR      R0,TYPTIM    ;GO REPORT THE TIMES
;:      TABLE   ;POINT TO THE PROPER TABLE
;:      RETURN
    
```

```

;:TABLE: MSGADR1    ;ADDRESS OF ASCIZ MESSAGE NUMBER 1
;:        MSGADR2    ;ADDRESS OF ASCIZ MESSAGE NUMBER 2
;:        MIN.ALLOWED ;MINIMUM TIME ALLOWED
;:        MAX.ALLOWED ;MAXIMUM TIME ALLOWED
    
```

```

;:TYPTIM: MOV      (R0)+,R2    ;PICKUP THE TABLE POINTER
;:          BIT      #SW06,$SWR ;INHIBIT TIME REPORTS?
;:          BNE     9$          ;YES--BRANCH
;:          MOV      (R2)+,2$    ;ADDRESS OF MESSAGE NUMBER 1
;:          MOV      (R2)+,R5    ;ADDRESS OF MESSAGE NUMBER 2
;:          MOV      (R2)+,R3    ;PICKUP THE LOW LIMIT
;:          MOV      (R2),R2     ;AND THE HIGH LIMIT
;:          MOV      #TIM.UP,R4  ;PARAMETER POINTER
;:          TYPE     ;TYPE THE MESSAGE
;:          .WORD    0     ;ASCIZ MESSAGE POINTER GOES HERE
;:          TST     14(R4)      ;DID ANY COUNTS OCCUR?
;:          BEQ     8$          ;NO--BRANCH
;:          TYPE     ,MSGMIN    ;'MIN='
;:          MOV      (R4)+,-(SP) ;PUT (R4)+ ON THE STACK
;:          JSR      PC,$SB2D   ;CHANGE TO DECIMAL ASCIZ
;:          JSR      PC,$SUPRS  ;TYPE WITHOUT LEADING ZEROS
;:          TYPE     ,MSGOUS    ;'O US'
;:          TST     (R4)+      ;ANY SEEKS BELOW THE LOW LIMIT
;:          BEQ     3$          ;NO--BRANCH
;:          TYPE     ,BLNKS2    ;TYPE 2 SPACES
;:          MOV      -2(R4),-(SP);PUT -2(R4) ON THE STACK
;:          JSR      PC,$SB2D   ;CHANGE TO DECIMAL ASCIZ
;:          JSR      PC,$SUPRS  ;TYPE WITHOUT LEADING ZEROS
;:          TYPE     ,M BELOW  ;'BELOW THE MINIMUM OF'
;:          MOV      R3,-(SP)  ;PUT R3 ON THE STACK
;:          JSR      PC,$SB2D   ;CHANGE TO DECIMAL ASCIZ
;:          JSR      PC,$SUPRS  ;TYPE WITHOUT LEADING ZEROS
    
```

```

818 031724 104401 046522
819 031730 104401 046506
820 031734 012446
      031736 004737 025260
      031742 004737 025510
821 031746 104401 046522
822 031752 005724
823 031754 001421
824 031756 104401 046657
825 031762 016446 177776
      031766 004737 025260
      031772 004737 025510
826 031776 104401 046556
827 032002 010246
      032004 004737 025260
      032010 004737 025510
828 032014 104401 046522
829 032020 104401 046514
830 032024 012446
831 032026 012446
832 032030 012446
833 032032 004737 025652
834 032036 006126
835 032040 100001
836 032042 005216
837 032044
      032044 004737 025260
      032050 004737 025510
838 032054 104401 046522
839 032060 104401 046657
840 032064 016446 177776
      032070 004737 025260
      032074 004737 025510
841 032100 022737 000012 001116
842 032106 001407
843 032110 022737 000014 001116
844 032116 001403
845 032120 104401 046625
846 032124 000402
847 032126 104401 046605
848 032132
849 032132 010537 031630
850 032136 001404
851 032140 005005
852 032142 000631
853 032144 104401 046642
854 032150 000200
855
856
857
858
859
860
861
862
863 032152 020237 002336
864 032156 001410

```

```

3$:
TYPE ,MSGOUS
TYPE ,MSGMAX ;'MAX='
MOV (R4)+,-(SP) ;PUT (R4)+ ON THE STACK
JSR PC,$SB2D ;CHANGE TO DECIMAL ASCIZ
JSR PC,$SUPRS ;TYPE WITHOUT LEADING ZEROS
TYPE ,MSGOUS
TST (R4)+ ;ANY SEEKS ABOVE THE HIGH LIMIT
BEQ 4$ ;NO--BRANCH
TYPE ,BLNKS2 ;TYPE 2 SPACES
MOV -2(R4),-(SP) ;PUT -2(R4) ON THE STACK
JSR PC,$SB2D ;CHANGE TO DECIMAL ASCIZ
JSR PC,$SUPRS ;TYPE WITHOUT LEADING ZEROS
TYPE ,MABOVE ;'ABOVE THE MAXIMUM OF'
MOV R2,-(SP) ;PUT R2 ON THE STACK
JSR PC,$SB2D ;CHANGE TO DECIMAL ASCIZ
JSR PC,$SUPRS ;TYPE WITHOUT LEADING ZEROS
4$:
TYPE ,MSGAVG ;'AVG='
MOV (R4)+,-(SP) ;FORM THE AVERAGE
MOV (R4)+,-(SP)
MOV (R4)+,-(SP)
JSR PC,$DIV ;IS THE REMAINDER OVER HALF?
ROL (SP)+ ;NO--BRANCH
BPL 5$ ;YES--ROUND UP
5$:
JSR PC,$SB2D ;CHANGE TO DECIMAL ASCIZ
JSR PC,$SUPRS ;TYPE WITHOUT LEADING ZEROS
TYPE ,MSGOUS ;TYPE 2 SPACES
TYPE ,BLNKS2 ;PUT -2(R4) ON THE STACK
MOV -2(R4),-(SP) ;CHANGE TO DECIMAL ASCIZ
JSR PC,$SB2D ;TYPE WITHOUT LEADING ZEROS
JSR PC,$SUPRS ;TEST 12 ?
CMP #12,$STSTM ;BRANCH IF SO
BEQ 6$ ;TEST 14 ?
CMP #14,$STSTM ;BRANCH IF SO
BEQ 6$ ;TYPE 'SEEKS TIMED'
TYPE ,MSGNUM ;TYPE IT
BR 7$ ;TYPE 'SEARCH TIME'
6$:
TYPE ,MSGSEA
7$:
MOV R5,2$ ;NEXT MESSAGE POINTER
BEQ 9$ ;IF NONE EXIT
CLR R5 ;NO MORE THAN 2
BR 1$
8$:
TYPE ,MSGNON
9$:
RTS R0 ;EXIT

```

```

;THIS SUBROUTINE WILL INCREMENT THE TRACK
;NUMBER (R2) BY THE AMOUNT SPECIFIED BY 'IT'.
;CALL
;
; JSR R0,INCTRK
;
; RETURN1 ;TRACK NUMBER GREATER THAN LT15
; RETURN2 ;TRACK NUMBER INCREMENTED
INCTRK: CMP R2,LT ;LAST TRACK COMPLETED?
        BEQ 2$ ;YES--EXIT

```



```

865 032160 063702 002340          ADD    IT,R2          ;NO--UPDATE TRACK
866 032164 020237 002336          CMP    R2,LT          ;TRACK TO BIG?
867 032170 003402                BLE    1$             ;NO--EXIT
868 032172 013702 002336          MOV    LT,R2          ;YES--SET TRACK TO LAST TRACK
869 032176 005720                1$:   TST    (R0)+      ;ADJUST FOR RETURN 2
870 032200 000200                2$:   RTS     RO          ;RETURN
871
872
873          ;THIS SUBROUTINE WILL INCREMENT THE CYLINDER
874          ;NUMBER (R1) BY THE AMOUNT SPECIFIED BY 'IC'.
875          ;CALL
876          ;
877          ;       JSR     RO,INCCYL
878          ;       RETURN1          ;CYLINDER NUMBER GREATER THAN LC15
879          ;       RETURN2          ;CYLINDER NUMBER INCREMENTED
880 032202 020137 002330  INCCYL:  CMP    R1,LC          ;LAST CYLINDER COMPLETED?
881 032206 001410                BEQ    2$             ;YES--EXIT
882 032210 063701 002332          ADD    IC,R1          ;NO--UPDATE CYLINDER
883 032214 020137 002330          CMP    R1,1           ;CYLINDER TO BIG?
884 032220 003402                BLE    1$             ;NO--EXIT
885 032222 013701 002330          MOV    LC,(1)         ;YES--SET CYLINDER TO LAST CYLINDER
886 032226 005720                1$:   TST    (R0)+      ;ADJUST FOR RETURN 2
887 032230 000200                2$:   RTS     RO          ;RETURN
888
889          ;THIS ROUTINE DECREASES THE SECTOR ADDRESS.
890          ;CALL
891          ;
892          ;       CLR     -(SP)          ;CLEAR THE STACK
893          ;       JSR     PC,DECSEC      ;SUBROUTINE ENTRY
894          ;       RETURN
895 032232 113766 045564 000002  DECSEC:  MOVB   RM.REG+RMDA,2(SP) ;PUT THE SECTOR ADDRESS ON THE STACK
896 032240 005366 000002                DEC    2(SP)          ;DECREMENT THE ADDRESS
897 032244 100003                BPL    1$             ;BR IF NOT CORRECTION NEEDED
898 032246 013766 002456 000002  1$:   MOV    PRMLT+24,2(SP) ;OVERFLOW OCCURED, FORCE TO MAXIMUM ADDRESS
899 032254 000207                RTS     PC             ;RETURN
900
901          ;THIS SUBROUTINE IS USED TO FILL THE DATA BUFFER
902          ;WITH ADDRESSES FROM 0 TO 31 WITH EACH ADDRESS
903          ;BEING STORED IN 256 CONSECUTIVE LOCATIONS
904          ;CALL
905          ;
906          ;       JSR     PC,FILBUF
907          ;       RETURN
908 032256 104412                FILBUF:  SAVREG          ;SAVE R0 - R5
909 032260 005000                CLR    RO             ;FIRST DISK ADDRESS
910 032262 012701 052776          MOV    #BUFFER,R1     ;START FILLING HERE
911 032266 012702 000400                1$:   MOV    #256,R2     ;DO 256 WORDS
912 032272 010021                2$:   MOV    RO,(R1)+    ;STORE
913 032274 005302                DEC    R2             ;MORE?
914 032276 003375                BGT    2$             ;YES--BRANCH
915 032300 005200                INC    RO             ;NO--UPDATE DISK ADDRESS
916 032302 023700 002456          CMP    PRMLT+24,RO     ;DONE?
917 032306 103367                BHS    1$             ;NO--BRANCH
918 032310 104413                RESREG          ;RESTORE R0 - R5
919 032312 000207                RTS     PC             ;RETURN
920
921          ;THIS ROUTINE WILL CLEAR THE BUFFER BY

```

```

922 ;SETTING EACH WORD TO '177400'.
923 ;CALL
924 ; JSR RO,CLRBUF
925 ; RETURN
926
927 032314 104412 CLRBUF: SAVREG ;SAVE R0 - R5
928 032316 012701 177400 MOV #177400,R1 ;WORD TO FILL BUFFER WITH
929 032322 012702 052776 MOV #BUFFER,R2 ;FIRST ADDRESS OF BUFFER
930 032326 012703 112776 MOV #BUFFER+<512.*32>,R3 ;LAST ADDRESS+2 OF BUFFER
931 032332 010122 1$: MOV R1,(R2)+ ;FILL WORDS 1, 9,...249,...5625
932 032334 010122 MOV R1,(R2)+ ;FILL WORDS 2,10,...250,...5626
933 032336 010122 MOV R1,(R2)+ ;FILL WORDS 3,11,...251,...5627
934 032340 010122 MOV R1,(R2)+ ;FILL WORDS 4,12,...252,...5628
935 032342 010122 MOV R1,(R2)+ ;FILL WORDS 5,13,...253,...5629
936 032344 010122 MOV R1,(R2)+ ;FILL WORDS 6,14,...254,...5630
937 032346 010122 MOV R1,(R2)+ ;FILL WORDS 7,15,...255,...5631
938 032350 010122 MOV R1,(R2)+ ;FILL WORDS 8,16,...256,...5632
939 032352 020203 CMP R2,R3 ;DONE?
940 032354 103766 BLO 1$ ;NO--BRANCH
941 032356 104413 RESREG ;RESTORE R0 - R5
942 032360 000200 RTS RO ;RETURN FROM CALL
943
944 ;THIS ROUTINE IS USED TO CHECK THE DATA BUFFER
945 ;FOR ADDRESSES 0 THROUGH 31 WITH EACH ADDRESS
946 ;BEING STORED IN 256 CONSECUTIVE LOCATIONS
947 ;CALL
948 ; JSR RO,CKSCTR
949 ; RETURN
950
951 032362 104412 CKSCTR: SAVREG ;SAVE R0 - R5
952 032364 162706 000004 SUB #4,SP ;RESERVE TEMP. STORAGE AREA
953 032370 005001 CLR R1 ;FIRST SECTOR
954 032372 012716 052776 MOV #BUFFER,(SP) ;FIRST ADDRESS OF DATA BUFFER
955 032376 005066 000002 CLR 2(SP) ;NO ERRORS
956 032402 012702 000020 1$: MOV #16.,R2 ;LOOP COUNT (16*16=256)
957 032406 011603 MOV (SP),R3 ;GET 1ST ADDRESS OF THIS SECTORS DATA
958 032410 2$:
962 032410 020123 CMP R1,(R3)+ ;WORD 1
032412 001063 BNE 7$ ;BRANCH IF BAD
032414 020123 CMP R1,(R3)+ ;WORD 2
032416 001061 BNE 7$ ;BRANCH IF BAD
032420 020123 CMP R1,(R3)+ ;WORD 3
032422 001057 BNE 7$ ;BRANCH IF BAD
032424 020123 CMP R1,(R3)+ ;WORD 4
032426 001055 BNE 7$ ;BRANCH IF BAD
032430 020123 CMP R1,(R3)+ ;WORD 5
032432 001053 BNE 7$ ;BRANCH IF BAD
032434 020123 CMP R1,(R3)+ ;WORD 6
032436 001051 BNE 7$ ;BRANCH IF BAD
032440 020123 CMP R1,(R3)+ ;WORD 7
032442 001047 BNE 7$ ;BRANCH IF BAD
032444 020123 CMP R1,(R3)+ ;WORD 8
032446 001045 BNE 7$ ;BRANCH IF BAD
032450 020123 CMP R1,(R3)+ ;WORD 9
032452 001043 BNE 7$ ;BRANCH IF BAD
032454 020123 CMP R1,(R3)+ ;WORD 10
032456 001041 BNE 7$ ;BRANCH IF BAD

```



```
1007 ;THIS ROUTINE WILL MOVE THE 16 WORDS OF THE
1008 ;DESIRED PATTERN INTO THE DATA BUFFER.
1009 ;CALL
1010 :      MOV      #NX,R0      ;PATTERN NUMBER INDEX TO R0
1011 :      JSR      PC,SETBUF
1012
1013 SETBUF: SAVREG      ;SAVE R0 - R5
1014 MOV      #BUFFER,R1    ;FIRST ADDRESS
1015 MOV      DTADPB+4,R2   ;WORD COUNT
1016 1$:      MOV      PAT.PT(R0),R3 ;PICKUP PATTERN POINTER
1017 MOV      (R3)+,(R1)+   ;MOVE WORD 1 INTO DATA BUFFER
1018 MOV      (R3)+,(R1)+   ;MOVE WORD 2 INTO DATA BUFFER
1019 MOV      (R3)+,(R1)+   ;MOVE WORD 3 INTO DATA BUFFER
1020 MOV      (R3)+,(R1)+   ;MOVE WORD 4 INTO DATA BUFFER
1021 MOV      (R3)+,(R1)+   ;MOVE WORD 5 INTO DATA BUFFER
1022 MOV      (R3)+,(R1)+   ;MOVE WORD 6 INTO DATA BUFFER
1023 MOV      (R3)+,(R1)+   ;MOVE WORD 7 INTO DATA BUFFER
1024 MOV      (R3)+,(R1)+   ;MOVE WORD 8 INTO DATA BUFFER
1025 MOV      (R3)+,(R1)+   ;MOVE WORD 9 INTO DATA BUFFER
1026 MOV      (R3)+,(R1)+   ;MOVE WORD 10 INTO DATA BUFFER
1027 MOV      (R3)+,(R1)+   ;MOVE WORD 11 INTO DATA BUFFER
1028 MOV      (R3)+,(R1)+   ;MOVE WORD 12 INTO DATA BUFFER
1029 MOV      (R3)+,(R1)+   ;MOVE WORD 13 INTO DATA BUFFER
1030 MOV      (R3)+,(R1)+   ;MOVE WORD 14 INTO DATA BUFFER
1031 MOV      (R3)+,(R1)+   ;MOVE WORD 15 INTO DATA BUFFER
1032 MOV      (R3)+,(R1)+   ;MOVE WORD 16 INTO DATA BUFFER
1033 1020 033010 062702 000020 ADD      #16.,R2 ;DONE?
1034 1021 033014 001353 BNE      1$ ;NO--BRANCH
1035 1022 033016 104413 RESREG   ;RESTORE R0 - R5
1036 1023 033020 000207 RTS      PC ;RETURN
1037
1038 ;THIS ROUTINE COMPARES A 16 WORD DATA PATTERN
1039 ;AGAINST THE DATA BUFFER
1040 ;CALL
1041 :      MOV      #NX,R0      ;PATTERN NUMBER INDEX TO R0
1042 :      JSR      PC,DATCMP
1043 :      RETURN
1044
1045 DATCMP: SAVREG      ;SAVE R0 - R5
1046 MOV      #BUFFER,R1    ;FIRST ADDRESS OF BUFFER
1047 MOV      DTADPB+4,R2   ;WORD COUNT
1048 CLR      -(SP)         ;NO ERROR
1049 1$:      MOV      PAT.PT(R0),R3 ;PATTERN POINTER
1050 2$:      SUB      (R3)+,(R1)+   ;CHECK WORD 1
1051 BNE      4$           ;BRANCH IF DIFFERENT
1052 SUB      (R3)+,(R1)+   ;CHECK WORD 2
1053 BNE      4$           ;BRANCH IF DIFFERENT
1054 SUB      (R3)+,(R1)+   ;CHECK WORD 3
1055 BNE      4$           ;BRANCH IF DIFFERENT
1056 SUB      (R3)+,(R1)+   ;CHECK WORD 4
1057 BNE      4$           ;BRANCH IF DIFFERENT
1058 SUB      (R3)+,(R1)+   ;CHECK WORD 5
1059 BNE      4$           ;BRANCH IF DIFFERENT
1060 SUB      (R3)+,(R1)+   ;CHECK WORD 6
1061 BNE      4$           ;BRANCH IF DIFFERENT
1062 SUB      (R3)+,(R1)+   ;CHECK WORD 7
```

	033074	001030			BNE	4\$:BRANCH IF DIFFERENT
	033076	162321			SUB	(R3)+,(R1)+		:CHECK WORD 8
	033100	001026			BNE	4\$:BRANCH IF DIFFERENT
	033102	162321			SUB	(R3)+,(R1)+		:CHECK WORD 9
	033104	001024			BNE	4\$:BRANCH IF DIFFERENT
	033106	162321			SUB	(R3)+,(R1)+		:CHECK WORD 10
	033110	001022			BNE	4\$:BRANCH IF DIFFERENT
	033112	162321			SUB	(R3)+,(R1)+		:CHECK WORD 11
	033114	001020			BNE	4\$:BRANCH IF DIFFERENT
	033116	162321			SUB	(R3)+,(R1)+		:CHECK WORD 12
	033120	001016			BNE	4\$:BRANCH IF DIFFERENT
	033122	162321			SUB	(R3)+,(R1)+		:CHECK WORD 13
	033124	001014			BNE	4\$:BRANCH IF DIFFERENT
	033126	162321			SUB	(R3)+,(R1)+		:CHECK WORD 14
	033130	001012			BNE	4\$:BRANCH IF DIFFERENT
	033132	162321			SUB	(R3)+,(R1)+		:CHECK WORD 15
	033134	001010			BNE	4\$:BRANCH IF DIFFERENT
	033136	162321			SUB	(R3)+,(R1)+		:CHECK WORD 16
	033140	001006			BNE	4\$:BRANCH IF DIFFERENT
1041	033142	062702	000020		ADD	#16.,R2		:DONE ?
1042	033146	001333			BNE	1\$:NO--BRANCH
1043	033150	005726		3\$:	TST	(SP)+		:YES -- CLEAN UP STACK
1044	033152	104413			RESREG			:RESTORE R0 - R5
1045	033154	000207			RTS	PC		
1046	033156	010104		4\$:	MOV	R1,R4		:FORM THE WORD NUMBER
1047	033160	162704	052776		SUB	#BUFFER,R4		
1048	033164	006204			ASR	R4		:WORD NUMBER
1049	033166	010305			MOV	R3,R5		:FORM ADDRESS TO CONTINUE FROM
1050	033170	166005	003520		SUB	PAT.PT(R0),R5		
1051	033174	006305			ASL	R5		
1052	033176	062705	033042		ADD	#2\$,R5		:ADDRESS
1053	033202	064341			ADD	-(R3),-(R1)		:RECONSTRUCT THE BAD WORD
1054	033204	010137	001136		MOV	R1,\$BDADR		:SAVE THE ERROR INFORMATION
1055	033210	010337	001134		MOV	R3,\$GDADR		
1056	033214	012137	001142		MOV	(R1)+,\$BDDAT		
1057	033220	012337	001140		MOV	(R3)+,\$GDDAT		
1058	033224	005716			TST	(SP)		:1ST DATA COMPARE ERROR?
1059	033226	001023			BNE	6\$:NO--BRANCH
1060	033230	013737	045550	001364	MOV	DTADPB+12,CYL.DS		:CYLINDER
1061	033236	113737	045547	001370	MOVB	DTADPB+11,TRK.DS		:TRACK
1062	033244	113737	045546	001366	MOVB	DTADPB+10,SEC.DS		:SECTOR
1063	033252	016600	000016		MOV	16(SP),R0		:GET TEST PC+4
1064	033256	012737	033266	001222	MOV	#5\$,\$ESCAPE		:ESCAPE TO 5\$ ON ERROR
1065	033264	104013			EMT	13		
1066	033266	016600	000014		MOV	14(SP),R0		:PATTERN NUMBER INDEX
1067	033272	105116		5\$:	COMB	(SP)		:SET THE ERROR SWITCH
1068	033274	000404			BR	7\$		
1069	033276	012737	033306	001222	MOV	#7\$,\$ESCAPE		:ESCAPE TO 7\$ ON ERROR
1070	033304	104014			EMT	14		
1071	033306	032777	000002	145640	BIT	#SW01,@SWR		:STOP DATA COMPARE?
1072	033314	001315			BNE	3\$:YES--EXIT
1073	033316	123737	001462	001117	CMPB	ERR.CT,\$ERFLG		:MAX. ERRORS?
1074	033324	101004			BHI	8\$:NO--BRANCH
1075	033326	013766	001346	000016	MOV	BYPASS,16(SP)		:YES--ERROR EXIT
1076	033334	000705			BR	3\$		
1077	033336	000115		8\$:	JMP	(R5)		:NO--CONTINUE AT NEXT WORD

```

1078
1079
1080 ;THIS ROUTINE WILL FILL THE DATA BUFFER (256*22 WORDS) WITH
1081 ;A RANDOM PATTERN. THE FIRST TWO WORDS OF EVERY 256 WILL
1082 ;BE THE BASE OF THE RANDOM NUMBER GENERATOR FOR THE
1083 ;NEXT 254 WORDS.
1084 ;NOTE: THIS ROUTINE DESTROYS R1 AND R2
1085 ;CALL
1086 ;
1087 ; JSR R0,FILRAN
1088 ; RETURN
1089
1088 033340 012701 052776
1089 033344 013702 002456
1090 033350 004037 033560
1091 033354 005302
1092 033356 100374
1093 033360 000200
1094
1095 ;THIS ROUTINE USES THE FIRST TWO WORDS OF THE
1096 ;READ BUFFER TO GENERATED A RANDOM PATTERN. THEN
1097 ;THE READ BUFFER IS COMPARED TO THE PATTERN GENERATED.
1098 ;NOTE: THIS ROUTINE DESTROYS R1-R4
1099 ;CALL
1100 ; JSR R0,RANCK
1101 ; RETURN
1102
1103 033362 013746 025646
1104 033366 013746 025650
1105 033372 012702 053776
1106 033376 012701 054776
1107 033402 010103
1108 033404 011237 025650
1109 033410 016237 000002 025646
1110 033416 004037 033560
1111 033422 012637 025650
1112 033426 012637 025646
1113 033432 005046
1114 033434 162322
1115 033436 001441
1116 033440 012737 033512 001222
1117 033446 064342
1118 033450 010237 001136
1119 033454 010337 001134
1120 033460 012237 001142
1121 033464 012337 001140
1122 033470 010204
1123 033472 162704 053776
1124 033476 006204
1125 033500 005716
1126 033502 001002
1127 033504 105116
1128 033506 104015
1129 033510
1130 033510 104016
1130 033512 032777 001000 145434
1131 033520 001012
1132 033522 123737 001462 001117
1133 033530 101406

FILRAN: MOV #BUFFER,R1
MOV PRMLMT+24,R2 ;MAXIMUM NUMBER OF SECTORS
1$: JSR R0,RANPAT
DEC R2
BPL 1$
RTS R0

RANCK: MOV $HINUM,-(SP) ;SAVE THE PRESENT RANDOM NUMBER
MOV $LONUM,-(SP)
MOV #BUFFER+512.,R2 ;READ BUFFER ADDRESS
MOV #BUFFER+1024.,R1 ;RANDOM PATTERN ADDRESS
MOV R1,R3 ;COPY IT INTO R3 FOR LATER USE
MOV (R2),$LONUM ;PRIME THE RANDOM NUMBER GENERATOR
MOV 2(R2),$HINUM
JSR R0,RANPAT ;GENERATE A RANDOM PATTERN
MOV (SP)+,$LONUM ;RESTORE PRESENT RANDOM NUMBER
MOV (SP)+,$HINUM
CLR -(SP) ;NO ERRORS
1$: SUB (R3)+,(R2)+ ;ARE THESE TWO WORDS DIFFERENT?
BEQ 4$ ;NO--BRANCH
MOV #3$,$ESCAPE ;ESCAPE TO 3$ ON ERROR
ADD -(R3)-,(R2) ;RECREATE THE BAD WORD
MOV R2,$BADDR ;ADDRESS OF BAD DATA
MOV R3,$GDADR ;ADDRESS OF GOOD DATA
MOV (R2)+,$BDDAT ;BAD DATA
MOV (R3)+,$GDDAT ;GOOD DATA
MOV R2,R4 ;FORM WORD NUMBER (1 TO 256)
SUB #BUFFER+512.,R4
ASR R4
TST (SP) ;FIRST ERROR
BNE 2$ ;NO--BRANCH
COMB (SP) ;YES--SET ERROR SWITCH
EMT 15

2$: EMT 16
3$: BIT #SW09,@SWR ;LOOP ON ERROR?
BNE 5$ ;YES--BRANCH
CMPB ERR.CT,$ERFLG ;MAX. ERRORS OCCURRED?
BLOS 5$ ;YES--BRANCH
    
```

```

1134 033532 032777 000002 145414          BIT      #SW01,@SWR      ;STOP COMPARING?
1135 033540 001002          BNE      5$           ;YES--BRANCH
1136 033542 020103          4$:     CMP      R1,R3      ;ALL DATA BEEN COMPARED?
1137 033544 101333          BHI      1$           ;NO--BRANCH
1138 033546 005726          5$:     TST      (SP)+      ;ERROR OCCUR?
1139 033550 001402          BEQ      6$           ;NO--BRANCH
1140 033552 013700 001346          MOV      BYPASS,R0     ;TAKE ERROR EXIT
1141 033556 000200          6$:     RTS      R0       ;EXIT
1142
1143          ;THIS ROUTINE FILLS A 256 WORD BUFFER WITH A RANDOM
1144          ;PATTERN OF WHICH THE FIRST TWO WORDS ARE THE BASE
1145          ;OF THE PATTERN.
1146          ;CALL
1147          ;
1148          ;     MOV      #ADR,R1          ;ADDRESS OF THE BUFFER
1149          ;     JSR      R0,RANPAT
1150          ;     RETURN
1151 033560 010246          RANPAT: MOV      R2,-(SP)      ;SAVE R2
1152 033562 012702 000200          MOV      #256./2.,R2      ;GENERATE 256 WORDS
1153 033566 000402          BR      2$
1154 033570 004737 025550          1$:     JSR      PC,$RAND      ;GENERATE A RANDOM NUMBER
1155 033574 013721 025650          2$:     MOV      $LONUM,(R1)+  ;PUT LOW WORD IN BUFFER
1156 033600 013721 025646          MOV      $HINUM,(R1)+  ;PUT HIGH WORD IN BUFFER
1157 033604 005302          DEC      R2            ;DONE?
1158 033606 003370          BGT      1$           ;NO--BRANCH
1159 033610 012602          MOV      (SP)+,R2      ;RESTORE R2
1160 033612 000200          RTS      R0           ;EXIT
1161
1162          ;THIS ROUTINE GENERATES RANDOM CYLINDER, TRACK, AND SECTOR
1163          ;ADDRESSES AND SAVES THEM IN THE DPB (DTADPB+10, 11 & DTADPB+12).
1164          ;NOTE: THIS ROUTINE DESTROYS R1-R3
1165          ;CALL
1166          ;
1167          ;     JSR      R0,RANADR
1168          ;     RETURN
1169 033614 004737 025550          RANADR: JSR      PC,$RAND      ;GENERATE A RANDOM NUMBER
1170 033620 113701 025650          MOV      $LONUM,R1      ;FORM SECTOR IN R1
1171 033624 113702 025651          MOV      $LONUM+1,R2    ;FORM TRACK IN R2
1172 033630 013703 025646          MOV      $HINUM,R3     ;FORM CYLINDER IN R3
1173 033634 105701          TSTB    R1            ;ENSURE THE SECTOR IS BETWEEN 0 AND 31
1174 033636 002403          BLT     2$
1175 033640 123701 002456          1$:     CMPB    PRMLT+24,R1    ;CHECK MAXIMUM SECTOR ADDRESS
1176 033644 103003          BHIS    3$
1177 033646 000241          2$:     CLC
1178 033650 106001          RORB    R1
1179 033652 000772          BR      1$
1180 033654 105702          3$:     TSTB    R2            ;ENSURE THE TRACK IS BETWEEN 0 AND LAST TRACK
1181 033656 002403          BLT     5$
1182 033660 123702 001372          4$:     CMPB    LSTRK,R2
1183 033664 002003          BGE     6$
1184 033666 000241          5$:     CLC
1185 033670 106002          RORB    R2
1186 033672 000772          BR      4$
1187 033674 023703 002326          6$:     CMP      FC,R3      ;ENSURE THE CYLINDER IS BETWEEN FC AND LC
1188 033700 003413          BLE     7$
1189 033702 000241          CLC
1190 033704 006003          ROR     R3
    
```

```

1191 033706 005503          ADC      R3
1192 033710 001371          BNE     6$
1193 033712 010103          MOV     R1,R3
1194 033714 000303          SWAB   R3
1195 033716 060203          ADD     R2,R3
1196 033720 005203          INC     R3
1197 033722 003364          BGT     6$
1198 033724 005403          NEG     R3
1199 033726 000762          BR      6$
1200 033730 023703 002330          CMP     LC,R3
1201 033734 002003          BGE     8$
1202 033736 000241          CLC
1203 033740 006003          ROR     R3
1204 033742 000772          BR      7$
1205 033744 023703 002326          8$:    CMP     FC,R3
1206 033750 003403          BLE     9$
1207 033752 005203          INC     R3
1208 033754 000303          SWAB   R3
1209 033756 000764          BR      7$
1210 033760 110137 045546          9$:    MOVB   R1,DTADPB+10 ;SAVE SECTOR ADDRESS
1211 033764 110237 045547          MOVB   R2,DTADPB+11 ;SAVE TRACK ADDRESS
1212 033770 010337 045550          MOV     R3,DTADPB+12 ;SAVE CYLINDER ADDRESS
1213 033774 000200          RTS     R0 ;RETURN
1214
1215          ;THIS ROUTINE IS USED TO INPUT THE 'CONTROL SWITCHES'.
1216          ;IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
1217          ;SETTING IS READ AND STORED.
1218          ;NOTE: THIS ROUTINE DESTROYS R3 AND R4
1219          ;CALL
1220          ;
1221          ;      JSR     PC,GETSWR
1222          ;      RETURN          ;(C.SWR)=DESIRED CONTROL SWITCHES
1223 033776 032777 000200 145150 GETSWR: BIT     #SW07,@SWR ;READ CONTROL SWITCHES?
1224 034004 001430          BEQ     2$ ;NO--BRANCH
1225 034006 104401 034014          TYPE   ,65$ ;TYPE ASCIZ STRING
1226          034012 000410          BR      64$ ;GET OVER THE ASCIZ
1227          ;:65$: .ASCIZ <CRLF>/SET SWR<07>=0/
1228          64$:
1229          1$:    MOV     #MSG.CS,R3 ;'CONTROL SWITCHES='
1230          034034 012703 045702          MOV     C.SWR,R4 ;PRESENT CONTROL SWITCH SETTINGS
1231          034040 013704 001314          JSR     R0,GETNUM ;GET THE NEW SWITCH SETTINGS
1232          034044 004037 034070          BR      1$ ; COMMA
1233          034050 000771          NOP     ;PERIOD
1234          034052 000240          MOV     C.SWR,SAVCSW ;SAVE PREVIOUS VALUE
1235          034054 013737 001314 001316          MOV     R4,C.SWR ; DOUBLE PERIOD-SAVE NEW SWITCH SETTING
1236          034062 010437 001314          2$:    MOV     R4,C.SWR
1237          034066 000207          RTS     PC ;RETURN FROM CALL
1238
1239          ;THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
1240          ;INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
1241          ;IF REQUIRED.
1242          ;NOTE: THIS ROUTINE DESTROYS R1
1243          ;CALL
1244          ;
1245          ;      MOV     #ADR,R3 ;ADDRESS OF ASCIZ MESSAGE
1246          ;      MOV     #NUM,R4 ;OCTAL NUMBER
1247          ;      JSR     R0,GETNUM
1248          ;      RETURN1 ;INPUT TERMINATED WITH A COMMA
1249          ;      RETURN2 ;WITH A PERIOD

```



```

1245          :          RETURNS          ;WITH A DOUBLE PERIOD
1246          :          :                ;R4=INPUT NUMBER AND
1247          :          :                ;R2=R4*32 FOR ALL
1248          :          :                ;THREE RETURNS
1249          :          :
1250 034070 010337 034076 GETNUM: MOV    R3,2$          ;SAVE MESSAGE POINTER
1251 034074 104401 1$:    TYPE          ;TYPE THE MESSAGE
1252 034076 000000 2$:    .WORD    0          ;MESSAGE POINTER GOES HERE
1253 034100 010446      MOV    R4,-(SP)      ;SAVE R4 FOR TYPEOUT
      034102 104402      TYPOC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1254 034104 104401 045725      TYPE    ,SLASH      ; /
1255 034110 104411      RDLIN          ;READ AN ASCII STRING
1256 034112 012601      MOV    (SP)+,R1      ;ADDRESS OF FIRST CHARACTER
1257 034114 004037 036340      JSR    R0,CK.CHR      ;CHECK ONE CHARACTER
      034120 034074      1$          ;ILLEGAL CHARACTER
      034122 034074      1$          ;CARRIAGE RETURN
      034124 034136      4$          ;'/'
      034126 034162      8$          ;'.'
      034130 034170      9$          ;'.'
      034132 034134      3$          ;DIGIT 0-9
1258 034134 005301 3$:    DEC    R1          ;DECREMENT THE INPUT POINTER
1259 034136 034136 4$:
      034136 004037 036600      JSR    R0,CK.NUM      ;CHECK THE NUMBER
      034142 034074      1$          ;ILLEGAL INPUT
      034144 034156      7$          ;TERMINATED WITH A '','' OR 'CR'
      034146 034154      6$          ;TERMINATED WITH A '',''
      034150 034152      5$          ;TERMINATED WITH A '',''
1260 034152 005720 5$:    TST    (R0)+      ;DOUBLE PERIOD
1261 034154 005720 6$:    TST    (R0)+      ;SINGLE PERIOD
1262 034156 010204 7$:    MOV    R2,R4      ;COMMA--SAVE INPUT NUMBER
1263 034160 000414      BR     11$          ;GO TO EXIT
1264 034162 105711 8$:    TSTB   (R1)          ;TERMINATOR AFTER A COMMA?
1265 034164 001343      BNE    1$          ;NO--LOOP
1266 034166 000411      BR     11$          ;YES--EXIT
1267 034170 105711 9$:    TSTB   (R1)          ;TERMINATOR AFTER A PERIOD?
1268 034172 001406      BEQ    10$         ;YES--EXIT
1269 034174 122721 000056      CMPB  #'.,(R1)+      ;NO--DOUBLE PERIOD?
1270 034200 001335      BNE    1$          ;NO--LOOP
1271 034202 105711      TSTB   (R1)          ;YES--TERMINATOR?
1272 034204 001333      BNE    1$          ;NO--LOOP
1273 034206 005720      TST    (R0)+      ;DOUBLE PERIOD
1274 034210 005720 10$:   TST    (R0)+      ;PERIOD
1275 034212 010402 11$:   MOV    R4,R2      ;COMMA--POSITION THE
1276 034214 000302      SWAB   R2          ;NUMBER IN CASE IT
1277 034216 006202      ASR    R2          ;IS THE PRIORITY LEVEL
1278 034220 006202      ASR    R2
1279 034222 006202      ASR    R2
1280 034224 000200      RTS     R0          ;EXIT
1281
1282          ;THIS ROUTINE IS USED TO CHANGE OR MODIFY
1283          ;THE TEST PARAMETERS. IT GIVES THE OPERATOR
1284          ;THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
1285          ;TESTS TO RUN AND HOW MANY TIMES TO
1286          ;REPEAT EACH TEST
1287
1288 034226 104412 GT.PRM: SAVREG          ;SAVE R0 - R5
1289 034230 005037 001326 GT.PR1: CLR    DRVSEL      ;NO DRIVE SELECTED

```

```

1290 034234 104401 034242          TYPE      ,65$          ;;TYPE ASCIZ STRING
      034240 000406          BR          64$          ;;GET OVER THE ASCIZ
      034256          ;;65$: .ASCIZ <CRLF>/DRIVE(S)=/
      64$:
1291 034256 104411          RDLIN          ;READ TTY
1292 034260 012601          MOV      (SP)+,R1      ;ADDRESS OF ASCIZ STRING
1293 034262 004037 036340          JSR      RO,CK.CHR     ;CHECK ONE CHARACTER
      034266 034230          GT.PR1         ;ILLEGAL CHARACTER
      034270 034230          GT.PR1         ;CARRIAGE RETURN
      034272 034230          GT.PR1         ;"/"
      034274 034230          GT.PR1         ;" "
      034276 034230          GT.PR1         ;" "
      034300 034302          1$          ;DIGIT 0-9
1294 034302 005301          1$: DEC      R1
1295 034304          2$:
      034304 012702 000007          MOV      #7,R2          ;UPPER LIMIT OF INPUT
      034310 004037 036414          JSR      RO,CK.DIG     ;CHECK THE DIGIT(S)
      034314 034230          GT.PR1         ;ILLEGAL INPUT
      034316 034230          GT.PR1         ;INPUT TOO LARGE
      034320 034326          3$          ;TERMINATED WITH A " " OR "CR"
      034322 034352          4$          ;TERMINATED WITH A " "
      034324 034352          4$          ;TERMINATED WITH A " "
1296 034326 156237 037054 001326 3$: BISB      ATABIT(R2),DRVSEL ;SET THE DRIVE SELECTED BIT
1297 034334 105741          TSTB      -(R1)        ;WAS THE LINE TERMINATED?
1298 034336 001362          BNE      2$          ;NO-GET THE NEXT DRIVE
1299 034340 005037 001330          CLR      TSTNMS      ;DESELECT ALL TESTS
1300 034344 005037 001332          CLR      TSTNMS+2
1301 034350 000405          BR          GTTST1    ;YES--SELECT TEST
1302 034352 156237 037054 001326 4$: BISB      ATABIT(R2),DRVSEL ;SET THE SELECTED DRIVE BITS
1303 034360 104413          GT.PR2: RESREG      ;RESTORE RO - R5
1304 034362 000207          RTS          PC          ;EXIT
1305
1306 034364          GTTST1:
      034364 104401 034372          TYPE      ,65$          ;;TYPE ASCIZ STRING
      034370 000403          BR          64$          ;;GET OVER THE ASCIZ
      034400          ;;65$: .ASCIZ /TEST=/
      64$:
1307 034400 104411          RDLIN          ;READ AN ASCIZ STRING
1308 034402 012601          MOV      (SP)+,R1      ;POINTER TO R1
1309 034404 122711 000056          CMPB     #'.,(R1)      ;DOUBLE PERIOD?
1310 034410 001007          BNE      1$          ;NO--BRANCH
1311 034412 122761 000056 000001          CMPB     #'.,1(R1)
1312 034420 001003          BNE      1$
1313 034422 105761 000002          TSTB     2(R1)        ;"CR"?
1314 034426 001754          BEQ      GT.PR2      ;YES--EXIT
1315 034430 005037 001330          1$: CLR      TSTNMS      ;NO TEST SELECTED
1316 034434 005037 001332          CLR      TSTNMS+2
1317 034440 005037 001334          CLR      OPNFLG      ;NO TESTS TO BE OPENED
1318 034444 005037 001336          CLR      OPNFLG+2
1319 034450 121127 000123          GTTST2: CMPB     (R1),#'S ;ALL SEEK TESTS?
1320 034454 001004          BNE      1$          ;NO--BRANCH
1321 034456 052737 000777 001330          BIS      #777,TSTNMS ;YES--SELECT TESTS 0-10
1322 034464 000552          BR          GTTST3
1323 034466 121127 000124          1$: CMPB     (R1),#'T ;ALL TIMING TESTS?
1324 034472 001004          BNE      2$          ;NO--BRANCH
1325 034474 052737 036000 001330          BIS      #36000,TSTNMS ;YES--SELECT TESTS 12-15
1326 034502 000543          BR          GTTST3
    
```

1327	034504	121127	000101		2\$:	CMPB	(R1),#'A	:ALL ADDRESSING TESTS?
1328	034510	001004				BNE	3\$:NO--BRANCH
1329	034512	052737	140000	001330		BIS	#140000,TSTNMS	:YES--SELECT TESTS 16 & 17
1330	034520	000534				BR	GTTST3	
1331	034522	121127	000104		3\$:	CMPB	(R1),#'D	:DATA TEST?
1332	034526	001004				BNE	4\$:NO--BRANCH
1333	034530	052737	000001	001332		BIS	#1,TSTNMS+2	:YES--SELECT TEST 20
1334	034536	000525				BR	GTTST3	
1335	034540	121127	000105		4\$:	CMPB	(R1),#'E	:EXERCISER TEST?
1336	034544	001004				BNE	5\$:NO--BRANCH
1337	034546	052737	000002	001332		BIS	#2,TSTNMS+2	:YES--SELECT TEST 21
1338	034554	000516				BR	GTTST3	
1339	034556	004037	036264		5\$:	JSR	R0,CK.OCT	:OCTAL DIGIT?
1340	034562	000514				BR	GTTST4	:NO--BRANCH
1341	034564	010205				MOV	R2,R5	:YES--SAVE IT
1342	034566	005201				INC	R1	:MOVE TO NEXT CHARACTER
1343	034570	004037	036264			JSR	R0,CK.OCT	:OCTAL DIGIT
1344	034574	000405				BR	6\$:NO--BRANCH
1345	034576	005201				INC	R1	:MOVE TO NEXT CHARACTER
1346	034600	006305				ASL	R5	:SCALE HIGH DIGIT
1347	034602	006305				ASL	R5	
1348	034604	006305				ASL	R5	
1349	034606	060502				ADD	R5,R2	:COMBINE HIGH & LOW DIGITS
1350	034610				6\$:			
1353	034610	020227	000022			CMP	R2,#22	:VALID TEST NUMBER?
1354	034614	003263				BGT	GTTST1	:NO--BRANCH
1355	034616	010237	035010			MOV	R2,9\$:SAVE THE TEST NUMBER
1356	034622	010204				MOV	R2,R4	:CONVERT TEST NUMBER INTO AN INDEX
1357	034624	042704	000017			BIC	#17,R4	:CLEAR UNWANTED BITS
1358	034630	006204				ASR	R4	:SHIFT THE BITS
1359	034632	006204				ASR	R4	
1360	034634	006204				ASR	R4	
1361	034636	006302				ASL	R2	
1362	034640	056264	001526	001330		BIS	BITS(R2),TSTNMS(R4)	:SELECT TEST
1363	034646	121127	000055			CMPB	(R1),#' -	:TEST STRING?
1364	034652	001060				BNE	GTTST4	:NO--BRANCH
1365	034654	005201				INC	R1	:YES--MOVE TO NEXT CHARACTER
1366	034656	004037	036264			JSR	R0,CK.OCT	:OCTAL DIGIT?
1367	034662	000640				BR	GTTST1	:NO--BRANCH
1368	034664	010205				MOV	R2,R5	:YES--SAVE IT
1369	034666	005201				INC	R1	:MOVE TO NEXT CHARACTER
1370	034670	004037	036264			JSR	R0,CK.OCT	:OCTAL DIGIT?
1371	034674	000405				BR	7\$:NO--BRANCH
1372	034676	005201				INC	R1	:YES--MOVE TO NEXT CHARACTER
1373	034700	006305				ASL	R5	:SCALE HIGH DIGIT
1374	034702	006305				ASL	R5	
1375	034704	006305				ASL	R5	
1376	034706	060502				ADD	R5,R2	:COMBINE HIGH & LOW DIGIT
1377	034710				7\$:			
1380	034710	020227	000022			CMP	R2,#22	:VALID TEST NUMBER?
1381	034714	003223				BGT	GTTST1	:NO--BRANCH
1382	034716	023702	035010			CMP	9\$,R2	:IS THE FIRST NUMBER OF THE
1383								:STRING SMALLER THAN THE LAST?
1384	034722	002220				BGE	GTTST1	:NO--BRANCH
1385	034724	010246				MOV	R2,-(SP)	:SAVE ENDING TEST NUMBER
1386	034726	013702	035010			MOV	9\$,R2	:GET STARTING TEST NUMBER
1387	034732	012637	035010			MOV	(SP)+,9\$:STORE ENDING TEST NUMBER

1388	034736	006337	035010		ASL	9\$:SHIFT ENDING TEST NUMBER
1389	034742	006302			ASL	R2		:SHIFT TEST NUMBER
1390	034744	010204		8\$:	MOV	R2,R4		:COPY TEST NUMBER INTO R4
1391	034746	042704	000037		BIC	#37,R4		:CLEAR LOWER BITS
1392	034752	006204			ASR	R4		:SHIFT THE TEST NUMBER
1393	034754	006204			ASR	R4		
1394	034756	006204			ASR	R4		
1395	034760	006204			ASR	R4		
1396	034762	056264	001526	001330	BIS	BITS(R2),TSTNMS(R4)		:SELECT THE TEST
1397	034770	062702	000002		ADD	#2,R2		:INCREMENT THE TEST NUMBER
1398	034774	020237	035010		CMP	R2,9\$:SEE IF FINISHED
1399	035000	101761			BLOS	8\$:BR IF NOT
1400	035002	162702	000002		SUB	#2,R2		:CORRECT TEST NUMBER
1401	035006	000402			BR	GTTST4		:CONTINUE
1402	035010	000000		9\$:	.WORD	0		:STORE TEST NUMBER HERE
1403	035012	005201		GTTST3:	INC	R1		:MOVE TO NEXT CHARACTER
1404	035014	121127	000056	GTTST4:	CMPB	(R1),#'		: 'PERIOD'?
1405	035020	001441			BEQ	GTTST5		:YES--BRANCH
1406	035022	005737	001330		TST	TSTNMS		:ANY TEST SELECTED THIS CYCLE?
1407	035026	001005			BNE	1\$:BR IF YES
1408	035030	0057	001332		TST	TSTNMS+2		:ANY TEST SELECTED THIS CYCLE ?
1409	035034	001002			BNE	1\$:BR IF YES
1410	035036	000137	034364		JMP	GTTST1		:NO
1411	035042	121127	000057	1\$:	CMPB	(R1),#'/		: 'OPEN'?
1412	035046	001004			BNE	2\$:NO--BRANCH
1413	035050	056264	001526	001334	BIS	BITS(R2), OPNFLG(R4)		:YES--SET BITS FOR TEST TO OPEN
1414	035056	000405			BR	3\$		
1415	035060	121127	000054	2\$:	CMPB	(R1),#'		: 'COMMA'?
1416	035064	001402			BEQ	3\$:BR IF YES
1417	035066	000137	034364		JMP	GTTST1		:NO
1418	035072	005201		3\$:	INC	R1		:MOVE TO NEXT CHARACTER
1419	035074	105711			TSTB	(R1)		: 'CR'?
1420	035076	001402			BEQ	4\$:BR IF 'CR'
1421	035100	000137	034450		JMP	GTTST2		:NO--GO GET NEXT CHARACTER
1422	035104	005737	001334	4\$:	TST	OPNFLG		:ANY TESTS TO OPEN ?
1423	035110	001042			BNE	OPNTST		:BR IF YES
1424	035112	005737	001336		TST	OPNFLG+2		:ANY TEST TO OPEN ?
1425	035116	001037			BNE	OPNTST		:BR IF YES
1426	035120	000137	034364		JMP	GTTST1		:NO--START AGAIN
1427	035124	005201		GTTST5:	INC	R1		:MOVE TO NEXT CHARACTER
1428	035126	121127	000056		CMPB	(R1),#'		: 'PERIOD'?
1429	035132	001414			BEQ	GTTST6		:YES--BRANCH
1430	035134	105711			TSTB	(R1)		: 'CR'?
1431	035136	001402			BEQ	1\$:YES--BRANCH
1432	035140	000137	034364		JMP	GTTST1		
1433	035144	005737	001334	1\$:	TST	OPNFLG		:ANY TESTS TO OPEN ?
1434	035150	001022			BNE	OPNTST		:BR IF YES
1435	035152	005737	001336		TST	OPNFLG+2		:ANY TESTS TO OPEN ?
1436	035156	001017			BNE	OPNTST		:BR IF YES
1437	035160	000137	034360		JMP	GT.PR2		:NO--GO START TESTING
1438	035164	005201		GTTST6:	INC	R1		:MOVE TO NEXT CHARACTER
1439	035166	105711			TSTB	(R1)		: 'CR'?
1440	035170	001402			BEQ	1\$:YES--BRANCH
1441	035172	000137	034364		JMP	GTTST1		:NO--GO ASK FOR TEST
1442	035176	005737	001334	1\$:	TST	OPNFLG		:ANY TESTS TO OPEN ?
1443	035202	001005			BNE	OPNTST		:BR IF YES
1444	035204	005737	001336		TST	OPNFLG+2		:ANY TESTS TO OPEN ?

```

1445 035210 001002
1446 035212 000137 034360      BNE      OPNTST      ;BR IF YES
                                JMP      GT.PR2      ;NO--GO START TESTING
1447
1448                                ;OPEN THE SELECTED TEST FOR CHANGES
1449
1450 035216 104412      OPNTST: SAVREG      ;SAVE R0 - R5
1451 035220 005027      CLR      (PC)+      ;START WITH TEST 0
1452 035222 000000      OPN.CT: .WORD      0      ;COUNT STORED HERE
1453 035224 000411      BR      OPN.2      ;SKIP THE INCREMENT
1454
1455 035226 005237 035222      OPN.1: INC      OPN.CT      ;MOVE TO THE NEXT TEST
1458 035232 022737 000022 035222  CMP      #22,OPN.CT      ;TEST NUMBER TOO BIG?
1459 035240 002003      BGE      OPN.2      ;NO--OPEN THE NEXT TEST
1460 035242 104413      RESREG
1461 035244 000137 034364      JMP      GTTST1      ;RESTORE R0 - R5
                                ;YES--GO ASK FOR MORE TESTS
1462
1463 035250 013705 035222      OPN.2: MOV      OPN.CT,R5      ;SETUP TO USE THE
1464 035254 006305      ASL      R5      ;TEST NUMBER AS AN INDEX
1465 035256 013703 035222      MOV      OPN.CT,R3      ;GET INDEX
1466 035262 042703 000017      BIC      #17,R3      ;CLEAR LOWER TEST BITS
1467 035266 006203      ASR      R3      ;SHIFT TEST NUMBER
1468 035270 006203      ASR      R3
1469 035272 006203      ASR      R3
1470 035274 036563 001526 001334  BIT      BITS(R5),OPNFLG(R3) ;OPEN THIS TEST?
1471 035302 001751      BEQ      OPN.1      ;NO--MOVE TO NEXT TEST
1472 035304 104401 035312      TYPE      ,65$      ;:TYPE ASCIZ STRING
                                BR      64$      ;:GET OVER THE ASCIZ
                                ;:65$: .ASCIZ / TEST /
                                ;:64$:
1473 035322 013746 035222      MOV      OPN.CT,-(SP) ;:SAVE OPN.CT FOR TYPEOUT
                                TYPOS      ;:GO TYPE--OCTAL ASCII
                                .BYTE      2      ;:TYPE 2 DIGIT(S)
                                .BYTE      0      ;:SUPPRESS LEADING ZEROS
1474 035332 104401 001231      TYPE      ,%CRLF      ;CR-LF
1475 035336 016500 002362      MOV      PRMPT(R5),PO ;PICKUP PARAMETER POINTER
1476 035342 011046      MOV      (R0),-(SP) ;SAVE THE VARIABLE INDICATOR
1477 035344 012702 002322      MOV      #PRM,R2      ;FIRST ADDRESS OF TABLE
1478 035350 000405      BR      2$
1479 035352 006216      1$: ASR      (SP)      ;CHECK FOR A VARIABLE
1480 035354 103403      BCS      2$      ;GO MOVE THIS ONE
1481 035356 001404      BEQ      OPNPRM      ;DONE
1482 035360 005722      TST      (R2)+      ;BUMP THE POINTER
1483 035362 000773      BR      1$
1484 035364 012022      2$: MOV      (R0)+,(R2)+ ;MOVE THIS VARIABLE INTO THE
1485 035366 000771      BR      1$      ;COMMON AREA
1486
1487 035370 013716 002322      OPNPRM: MOV      PRM,(SP) ;GET THE VARIABLE INDICATOR
1488 035374 005004      CLR      R4      ;ZERO THE INDEX
1489 035376 006216      1$: ASR      (SP)      ;CHECK FOR A VARIABLE
1490 035400 103403      BCS      3$      ;GO GET IT
1491 035402 001772      BEQ      OPNPRM      ;OUT OF VARIABLES
1492 035404 005724      2$: TST      (R4)+      ;UPDATE THE INDEX
1493 035406 000773      BR      1$
1494 035410 005764 002432      3$: TST      PRMLMT(R4) ;IS THE MAX. MAGNITUDE NEG?
1495 035414 100466      BMI      OPNPAT      ;YES--THEN IT IS THE PATTERN
1496 035416 104401 046657      TYPE      ,BLNKS2 ;TYPE 2 SPACES
1497 035422 016437 002464 035432      MOV      PRMSG(R4),4$ ;TYPE THE NAME OF THIS VARIABLE

```

```

1498 035430 104401
1499 035432 000000
1500 035434 104401 045700
1501 035440 016446 002324
      035444 004737 025260
      035450 004737 025510
1502 035454 104401 045725
1503 035460 104411
1504 035462 012601
1505 035464 004037 036340
      035470 035410
      035472 035404
      035474 035542
      035476 035504
      035500 035512
      035502 035540
1506 035504 105711
1507 035506 001340
1508 035510 000735
1509 035512 105711
1510 035514 001002
1511 035516 000137 036132
1512 035522 122721 000056
1513 035526 001330
1514 035530 105711
1515 035532 001326
1516 035534 000137 036150
1517
1518 035540 005301
1519 035542
      035542 016402 002432
      035546 004037 036414
      035552 035410
      035554 035410
      035556 035564
      035560 036126
      035562 036144
1520 035564 010264 002324
1521 035570 000705
1522
1523
1524
1525 035572 104401 046657
1526 035576 104401 045674
1527 035602 104401 045700
1528 035606 016446 002324
      035612 104402
1529 035614 104401 046660
1530 035620 104411
1531 035622 012601
1532 035624 004037 036340
      035630 035572
      035632 035370
      035634 035666
      035636 035570
      035640 035644
      035642 035664

```

4\$: TYPE .WORD 0
TYPE ,MSG.EQ ;TYPE '='
MOV RPT(R4),-(SP) ;PUT RPT(R4) ON THE STACK
JSR PC,\$SB2D ;CHANGE TO DECIMAL ASCIZ
JSR PC,\$SUPRS ;TYPE WITHOUT LEADING ZEROS
TYPE ,SLASH ;TYPE ' / '
RD LIN
MOV (SP)+,R1 ;READ AN ASCIZ STRING
JSR RO,CK.CHR ;CHECK ONE CHARACTER
3\$;ILLEGAL CHARACTER
2\$;CARRIAGE RETURN
9\$;'
5\$;'
6\$;'
8\$;DIGIT 0-9
5\$: TSTB (R1) ;'CR'?
BNE 3\$;NO--STAY ON THIS VARIABLE
BR 2\$;YES--MOVE TO NEXT VARIABLE
6\$: TSTB (R1) ;IS THERE A 'CR' AFTER THE PERIOD?
BNE 7\$;NO
JMP OPN.N2 ;YES--GO CLOSE THIS TEST
7\$: CMPB #'.,'(R1)+ ;DOUBLE PERIOD?
BNE 3\$;NO--GO ASK FOR THIS VARIABLE
TSTB (R1) ;YES--IS A 'CR' AFTER THE DOUBLE PERIOD?
BNE 3\$;NO--ASK FOR THIS VARIABLE AGAIN
JMP OPN.X2 ;YES--CLOSE ALL TEST
8\$: DEC R1 ;BACK THE POINTER UP BY ONE
9\$: MOV PRMLMT(R4),R2 ;UPPER LIMIT OF INPUT
JSR RO,CK.DIG ;CHECK THE DIGIT(S)
3\$;ILLEGAL INPUT
3\$;INPUT TOO LARGE
10\$;TERMINATED WITH A '.,' OR 'CR'
OPN.N1 ;TERMINATED WITH A '.,'
OPN.X1 ;TERMINATED WITH A '.,'
10\$: MOV R2,RPT(R4) ;SAVE THIS VARIABLE
BR 2\$;MOVE TO NEXT VARIABLE
;OPEN PATTERN FOR CHANGES
OPNPAT: TYPE ,BLNKS2 ;TYPE 2 SPACES
TYPE ,MSG.PAT ;TYPE 'PAT'
TYPE ,MSG.EQ ;TYPE '='
MOV RPT(R4),-(SP) ;SAVE RPT(R4) FOR TYPEOUT
TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE ,BLNKS1 ;TYPE ONE SPACE
RD LIN ;READ ASCIZ STRING
MOV (SP)+,R1 ;PICKUP POINTER
JSR RO,CK.CHR ;CHECK ONE CHARACTER
OPNPAT ;ILLEGAL CHARACTER
OPNPRM ;CARRIAGE RETURN
3\$;'
OP PRM ;'
;DIGIT 0-9

```

INTEG DIVIDE ROUTINE
1533 035644 105711      1$:  TSTB   (R1)      ;'CR' AFTER THE PERIOD?
1534 035646 001531      BEQ    OPN.N2     ;YES--GO CLOSE THIS TEST
1535 035650 122721 000056  CMPB   #'..(R1)+  ;NO--PERIOD?
1536 035654 001346      BNE    OPNPAT     ;NO--LOOP
1537 035656 105711      TSTB   (R1)      ;'CR' AFTER A DOUBLE PERIOD?
1538 035660 001533      BEQ    OPN.X2     ;YES--GO START TESTING
1539 035662 000743      BR     OPNPAT     ;NO--LOOP
1540 035664 005301      2$:  DEC    R1      ;BACKUP THE ASCII POINTER
1541 035666
3$:
      035666 004037 036600      JSR    RO,CK.NUM ;CHECK THE NUMBER
      035672 035572      OPNPAT ;ILLEGAL INPUT
      035674 035702      4$     ;TERMINATED WITH A '..' OR 'CR'
      035676 036126      OPN.N1 ;TERMINATED WITH A '..'
      035700 036144      OPN.X1 ;TERMINATED WITH A '..'
1542 035702 010264 002324  4$:  MOV    R2,RPT(R4) ;SAVE THE INPUT NUMBER
1543 035706 006002      ROR    R2         ;OPEN PATTERN 0?
1544 035710 103227      BCC   OPNPRM     ;NO--START AT BEGINNING OF PARAMETER TABLE
1545 035712 104412      SAVREG ;SAVE R0 - R5
1546
1547
1548      ;OPEN WORDS IN PATTERN #0 FOR CHANGES
1549 035714 005000      OPNWDS: CLR   R0      ;START WITH WORD 0
1550 035716 012704 003560      MOV    #PATO,R4
1551 035722
1$:
      035722 104401 035730      TYPE   ,65$      ;;TYPE ASCII STRING
      035726 000403      BR     64$      ;;GET OVER THE ASCII
;;65$: .ASCIIZ / WD/
64$:
1552 035736
      035736 010046      MOV    RO,-(SP)   ;PUT R0 ON THE STACK
      035740 004737 025260      JSR    PC,$SB2D  ;CHANGE TO DECIMAL ASCIIZ
      035744 004737 025510      JSR    PC,$SUPRS ;TYPE WITHOUT LEADING ZEROS
1553 035750 104401 045700      TYPE   ,MSG.EQ   ;TYPE '='
1554 035754 011446      MOV    (R4),-(SP) ;SAVE (R4) FOR TYPEOUT
      035756 104402      TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1555 035760 104411      RDLIN ;READ ASCII STRING
1556 035762 012601      MOV    (SP)+,R1  ;PICKUP THE POINTER
1557 035764 004037 036340      JSR    RO,CK.CHR ;CHECK ONE CHARACTER
      035770 035722      1$     ;ILLEGAL CHARACTER
      035772 036024      5$     ;CARRIAGE RETURN
      035774 036006      3$     ;'/'
      035776 036024      5$     ;'..'
      036000 036040      6$     ;'...'
      036002 036004      2$     ;DIGIT 0-9
1558 036004 005301      2$:  DEC    R1      ;BACKUP THE ASCII POINTER
1559 036006
3$:
      036006 004037 036600      JSR    RO,CK.NUM ;CHECK THE NUMBER
      036012 035722      1$     ;ILLEGAL INPUT
      036014 036022      4$     ;TERMINATED WITH A '..' OR 'CR'
      036016 036060      7$     ;TERMINATED WITH A '..'
      036020 036072      9$     ;TERMINATED WITH A '..'
1560 036022 010214      4$:  MOV    R2,(P )  ;SAVE THE INPUT
1561 036024 005724      5$:  TST   (R4)+     ;MOVE TO NEXT WORD
1562 036026 005200      INC   R0         ;INCREMENT THE COUNT
1563 036030 022700 000020      CMP   #16.,R0   ;COUNT TO LARGE?
1564 036034 003532      BGT   1$        ;NO--BRANCH
1565 036036 000726      BR    OPNWDS     ;YES--BRANCH
1566 036040 105711      6$:  TSTB   (R1)      ;'CR' AFTER THE PERIOD?

```

```

1567 036042 001407          BEQ      8$          ;YES--GO CLOSE THIS TEST
1568 036044 122721 000056    CMPB    #'.,(R1)+    ;NO--PERIOD?
1569 036050 001324          BNE     1$          ;NO--BRANCH ILLEGAL INPUT STRING
1570 036052 105711          TSTB   (R1)         ;'CR' AFTER THE 'PERIOD-PERIOD'?
1571 036054 001407          BEQ     10$         ;YES--GO START TESTING
1572 036056 000721          BR     1$          ;NO--LOOP
1573 036060 010224          7$:    MOV     R2,(R4)+ ;SAVE THE INPUT
1574 036062 004737 036104    8$:    JSR     PC,CLSWDS ;CLOSE THE DATA PATTERN
1575 036066 104413          RESREG ;RESTORE R0 - R5
1576 036070 000420          BR     OPN.N2      ;MOVE TO NEXT TEST
1577 036072 010224          9$:    MOV     R2,(R4)+ ;SAVE THE INPUT
1578 036074 004737 036104    10$:   JSR     PC,CLSWDS ;CLOSE THE DATA PATTERN
1579 036100 104413          RESREG ;RESTORE R0 - R5
1580 036102 000422          BR     OPN.X2      ;START TESTING
1581
1582          ;CLOSE PATTERN #0 AND SAVE CHANGED WORDS
1583
1584 036104 012701 003560    CLSWDS: MOV    #PA10,R1 ;FIRST ADDRESS OF DATA PATTERN
1585 036110 005200          1$:    INC     R0       ;COUNT THE LAST WORD THAT WAS STORED
1586 036112 022700 000017    CMP     #15.,R0     ;END OF TABLE
1587 036116 002402          BLT    2$          ;YES--EXIT
1588 036120 012124          MOV    (R1)+,(R4)+ ;COPY
1589 036122 000772          BR     1$          ;LOOP
1590 036124 000207          2$:    RTS     PC     ;RETURN
1591
1592 036126 010264 002324    OPN.N1: MOV    R2,RPT(R4) ;SAVE THIS VARIABLE
1593 036132 005726          OPN.N2: TST    (SP)+   ;CLEAN OFF THE STACK
1594 036134 004737 036204    JSR    PC,CLOSE    ;CLOSE THIS TEST
1595 036140 000137 035226    JMP    OPN.1       ;GO OPEN THE NEXT TEST
1596
1597 036144 010264 002324    OPN.X1: MOV    R2,RPT(R4) ;SAVE THIS VARIABLE
1598 036150 005726          OPN.X2: TST    (SP)+   ;CLEAN OFF THE STACK
1599 036152 004737 036204    1$:    JSR    PC,CLOSE ;CLOSE THIS TEST
1600 036156 005725          2$:    TST    (R5)+   ;UPDATE THE INDEX
1601 036160 020527 000034    CMP    R5,#16*2    ;INDEX TO BIG?
1602 036164 002403          BLT    3$          ;NO--BRANCH
1603 036166 104413          RESREG ;RESTORE R0 - R5
1604 036170 000137 034360    JMP    GT.PR2      ;GO TO EXIT
1605 036174 036503 001526    3$:    BIT    BITS(R5),R3 ;IS THIS TEST OPEN FOR CHANGE?
1606 036200 001364          BNE    1$          ;YES--GO CLOSE IT
1607 036202 000765          BR     2$          ;NO--MOVE TO NEXT TEST
1608
1609          ;CLOSE CURRENT TEST THAT WAS OPEN FOR CHANGES
1610
1611 036204 104412          CLOSE: SAVREG ;SAVE R0 - R5
1612 036206 012700 002322    MOV    #PRM,R0     ;'FROM' ADDRESS
1613 036212 016501 002362    MOV    PRMPT(R5),R1 ;'TO' ADDRESS
1614 036216 012002          MOV    (R0)+,R2    ;'FROM' INDICATOR
1615 036220 012103          MOV    (R1)+,R3    ;'TO' INDICATOR
1616 036222 012704 000001    MOV    #1,R4       ;TEST BIT START A 'RPT'
1617 036226 030402          1$:    BIT    R4,R2     ;PARAMETER TO BE MOVED?
1618 036230 001403          BEQ    2$          ;NO--BRANCH
1619 036232 030403          BIT    R4,R3       ;A PLACE TO PUT IT?
1620 036234 001404          BEQ    3$          ;NO--BRANCH
1621 036236 011011          MOV    (R0),(R1)   ;YES--MOVE 'FROM' TO 'TO'
1622 036240 030403          2$:    BIT    R4,R3   ;'TO' PARAMETER?
1623 036242 001401          BEQ    3$          ;NO--BRANCH
    
```



```

1624 036244 005721          TST      (R1)+      ;YES--UPDATE THE POINTER
1625 036246 005720          3$: TST      (R0)+      ;UPDATE FROM POINTER
1626 036250 006304          ASL      R4          ;POSITION THE TEST BIT
1627 036252 032704 002000  BIT      #BIT10,R4  ;DONE?
1628 036256 001763          BEQ      1$          ;NO--BRANCH
1629 036260 104413          RESREG   1$          ;RESTORE R0 - R5
1630 036262 000207          RTS      PC          ;RETURN
1631
1632          ;THIS ROUTINE IS USED TO CHECK IF AN
1633          ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
1634          ;CALL
1635          ;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
1636          ;      JSR      RO,CK.OCT   ;CHECK THE CHARACTER
1637          ;      RETURN1   ;CHARACTER IS NOT BETWEEN 0-7
1638          ;      RETURN2   ;CHARACTER IS IN R2 AS A
1639          ;      ;      ;OCTAL DIGIT
1640
1641 036264 121127 000060  CK.OCT: CMPB    (R1),#'0    ;LESS THAN ZERO?
1642 036270 103407          BLO     1$          ;YES -- BRANCH
1643 036272 121127 000067  CMPB    (R1),#'7    ;GREATER THAN SEVEN?
1644 036276 101004          BHI     1$          ;YES -- BRANCH
1645 036300 111102          MOVB   (R1),R2     ;GET THE CHARACTER
1646 036302 042702 177770  BIC     #'^C7,R2   ;STRIP AWAY THE ASCII
1647 036306 005720          TST    (R0)+      ;ADJUST FOR RETURN
1648 036310 000200          1$:  RTS      RO          ;RETURN
1649
1650          ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
1651          ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
1652          ;CALL
1653          ;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
1654          ;      JSR      RO,CK.DEC   ;CHECK THE CHARACTER
1655          ;      RETURN1   ;NOT BETWEEN 0 AND 9
1656          ;      RETURN2   ;BETWEEN 0 AND 9
1657          ;      ;      ;R2 = DIGIT
1658
1659 036312 121127 000060  CK.DEC: CMPB    (R1),#'0    ;LESS THAN ZERO?
1660 036316 103407          BLO     1$          ;YES -- BRANCH
1661 036320 121127 000071  CMPB    (R1),#'9    ;GREATER THAN NINE?
1662 036324 10 004          BHI     1$          ;YES -- BRANCH
1663 036326 111102          MOVB   (R1),R2     ;GET THE CHARACTER
1664 036330 042702 000060  BIC     #'0,R2     ;STRIP AWAY THE ASCII
1665 036334 005720          TST    (R0)+      ;ADJUST FOR RETURN
1666 036336 000200          1$:  RTS      RO          ;RETURN
1667
1668          ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
1669          ;DETERMINE WHAT IT IS.
1670          ;CALL
1671          ;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
1672          ;      JSR      RO,CK.CHR   ;CHECK CHARACTER
1673          ;      RETURN  ADR1   ;UNKNOWN CHARACTER
1674          ;      RETURN  ADR2   ;CARRIAGE RETURN * (R1)=ADR+1
1675          ;      RETURN  ADR3   ;SLASH * (R1)=ADR+1
1676          ;      RETURN  ADR4   ;COMMA * (R1)=ADR+1
1677          ;      RETURN  ADR5   ;PERIOD * (R1)=ADR+1
1678          ;      RETURN  ADR6   ;DIGIT BETWEEN 0 AND 9.
1679          ;      ;      ;R2 = DIGIT * (R1)=ADR+1
1680

```

```

1681 036340 105711          CK.CHR: TSTB   (R1)          ;'CARRIAGE RETURN'?
1682 036342 001420          BEQ    4$          ;YES -- BRANCH
1683 036344 121127 000057  CMPB   (R1),#'/'   ;'SLASH'?
1684 036350 001414          BEQ    3$          ;YES -- BRANCH
1685 036352 121127 000054  CMPB   (R1),#','   ;'COMMA'?
1686 036356 001410          BEQ    2$          ;YES -- BRANCH
1687 036360 121127 000056  CMPB   (R1),#'.   ;'PERIOD'?
1688 036364 001404          BEQ    1$          ;YES -- BRANCH
1689 036366 004037 036312  JSR    RO,CK.DEC   ;'DIGIT'?
1690 036372 000406          BR     5$          ;NO -- BRANCH
1691 036374 005720          TST   (RO)+       ;DIGIT BETWEEN 0-9
1692 036376 005720          1$:   TST   (RO)+       ;PERIOD
1693 036400 005720          2$:   TST   (RO)+       ;COMMA
1694 036402 005720          3$:   TST   (RO)+       ;SLASH
1695 036404 005720          4$:   TST   (RO)+       ;CARRIAGE RETURN
1696 036406 005201          INC   R1          ;MOVE POINTER TO NEXT CHARACTER
1697 036410 011000          5$:   MOV   (RO),RO    ;UNKNOWN CHARACTER
1698 036412 000200          RTS    RO          ;RETURN
1699
1700          ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
1701          ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
1702          ;CALL
1703          ;      MOV   #ADR,R1          ;ADDRESS OF ASCII STRING
1704          ;      MOV   #NUM,R2        ;MAX. MAGNITUDE OF INPUT NUMBER
1705          ;      JSR   RO,CK.DIG      ;CHECK DIGITS
1706          ;      RETURN ADR1         ;ILLEGAL CHARACTER -- R2=?
1707          ;      RETURN ADR2         ;INPUT NUMBER TOO LARGE -- R2=?
1708          ;      RETURN ADR3         ;'COMMA' -- R2 = NUMBER
1709          ;      RETURN ADR4         ;'PERIOD' -- R2 = NUMBER
1710          ;      RETURN ADR5         ;'PERIOD-PERIOD' -- R2 = NUMBER
1711
1712 036414 010446          CK.DIG: MOV   R4,-(SP)   ;SAVE R4
1713 036416 010346          MOV   R3,-(SP)   ;SAVE R3
1714 036420 010246          MOV   R2,-(SP)   ;SAVE THE MAX. SIZE ON THE STACK
1715 036422 005002          CLR   R2          ;START WITH 0
1716 036424 005003          CLR   R3
1717 036426 005004          CLR   R4
1718 036430 004037 036340  JSR   RO,CK.CHR   ;CHECK ONE CHARACTER
1719          036434 036564          9$          ;ILLEGAL CHARACTER
1720          036436 036564          9$          ;CARRIAGE RETURN
1721          036440 036564          9$          ;'/'
1722          036442 036564          9$          ;'.'
1723          036444 036564          9$          ;'0-9'
1724          036446 036450          1$          ;DIGIT 0-9
1725 036450 006303          1$:   ASL   R3          ;2
1726 036452 010346          MOV   R3,-(SP)   ;SAVE *2
1727 036454 006303          ASL   R3          ;4
1728 036456 006303          ASL   R3          ;8
1729 036460 062603          ADD   (SP)+,R3   ;(*8)+(*2)=*10.
1730 036462 060203          ADD   R2,R3      ;UPDATE THE INPUT NUMBER
1731 036464 004037 036340  JSR   RO,CK.CHR   ;CHECK ONE CHARACTER
1732          036470 036564          9$          ;ILLEGAL CHARACTER
1733          036472 036504          2$          ;CARRIAGE RETURN
1734          036474 036564          9$          ;'/'
1735          036476 036512          4$          ;'.'
1736          036500 036510          3$          ;'0-9'
1737          036502 036450          1$          ;DIGIT 0-9

```

```

1726 036504 005301      2$: DEC R1 ;BACKUP THE CHARACTER POINTER
1727 036506 000401      BR 4$ ;CONTINUE
1728 036510 005724      3$: TST (R4)+ ;'PERIOD'
1729 036512 005724      4$: TST (R4)+ ;'COMMA' OR 'CR'
1730 036514 004037 036340 JSR RO,CK.CHR ;CHECK ONE CHARACTER
      036520 036564      9$ ;ILLEGAL CHARACTER
      036522 036554      7$ ;CARRIAGE RETURN
      036524 036564      9$ ;'/'
      036526 036564      9$ ;'..'
      036530 036534      5$ ;'..'
      036532 036544      6$ ;DIGIT 0-9
1731 036534 005724      5$: TST (R4)+ ;'PERIOD-PERIOD'
1732 036536 105711      TSTB (R1) ;'CR'?
1733 036540 001405      BEQ 7$ ;YES--BRANCH
1734 036542 000410      BR 9$
1735 036544 126127 177776 000054 6$: CMPB -2(R1),#' ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
1736 036552 001004      BNE 9$ ;NO--EXIT
1737 036554 020316      7$: CMP R3,(SP) ;INPUT TO LARGE?
1738 036556 101001      BHI 8$ ;YES -- BRANCH
1739 036560 060400      ADD R4,RO ;ADJUST RETURN ADDRESS
1740 036562 005720      8$: TST (R0)+
1741 036564 010302      9$: MOV R3,R2 ;NUMBER TO R2
      TST (SP)+ ;CLEAN MAX. SIZE OFF OF STACK
1742 036566 005726      MOV (SP)+,R3 ;RESTORE R3
1743 036570 012603      MOV (SP)+,R4 ;RESTORE R4
1744 036572 012604      MOV (R0),RO ;GET RETURN ADDRESS
1745 036574 011000      RTS RO ;RETURN
1746 036576 000200
1747
1748 ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
1749 ;AND FORMS AN OCTAL NUMBER IN R2
1750 ;CALL:
1751 ;: MOV #ADR,R1 ;ADDRESS OF ASCIZ STRING
1752 ;: JSR RO,CK.NUM ;GO FORM THE NUMBER
1753 ;: RETURN ADR1 ;ILLEGAL CHARACTER IN THE INPUT STRING
1754 ;: RETURN ADR2 ;'COMMA' OR 'CR'--R2=NUMBER
1755 ;: RETURN ADR3 ;'PERIOD'--R2=NUMBER
1756 ;: RETURN ADR4 ;'PERIOD-PERIOD'--R2=NUMBER
1757
1758 036600 010346      CK.NUM: MOV R3,-(SP) ;SAVE R3
1759 036602 005003      CLR R3 ;START NUMBER AT ZERO
1760 036604 004037 036264 JSR RO,CK.OCT ;OCTAL DIGIT?
1761 036610 000440      BR 6$ ;NO--BRANCH
1762 036612 005201      1$: INC R1 ;MOVE TO NEXT CHARACTER
1763 036614 006303      ASL R3 ;FOR THE OCTAL NUMBER IN R3
1764 036616 103435      BCS 6$ ;DON'T LET IT GET TO BIG
1765 036620 006303      ASL R3
1766 036622 103433      BCS 6$
1767 036624 006303      ASL R3
1768 036626 103431      BCS 6$
1769 036630 060203      ADD R2,R3
1770 036632 004037 036264 JSR RO,CK.OCT ;IS THIS AN OCTAL DIGIT?
1771 036636 000401      BR 2$ ;NO--FIND OUT WHAT IT IS
1772 036640 000764      BR 1$ ;YES--MAKE IT PART OF THE NUMBER
1773 036642 010302      2$: MOV R3,R2 ;SAVE THE OCTAL NUMBER
1774 036644 005003      CLR R3 ;START WITH ZERO INDEX
1775 036646 004037 036340 JSR RO,CK.CHR ;CHECK ONE CHARACTER
      036652 036712      6$ ;ILLEGAL CHARACTER

```

036654	036702		5\$:CARRIAGE RETURN
036656	036712		6\$: '/'
036660	036702		5\$: ' ' .
036662	036666		3\$: ' ' .
036664	036712		6\$: DIGIT 0-9
1776 036666	005723	000056	3\$:	TST (R3)+	: 'PERIOD'
1777 036670	121127			CMPB (R1), #'.	: 'PERIOD-PERIOD'?
1778 036674	001002			BNE 5\$: NO--BRANCH
1779 036676	005201			INC R1	: YES--ADVANCE THE POINTER
1780 036700	005723		4\$:	TST (R3)+	: 'PERIOD-PERIOD'
1781 036702	005723		5\$:	TST (R3)+	: 'COMMA'
1782 036704	105711			TSTB (R1)	: 'CR'?
1783 036706	001001			BNE 6\$: NO--BRANCH
1784 036710	060300			ADD R3, R0	: YES--SAVE THE OCTAL NUMBER
1785 036712	012603		6\$:	MOV (SP)+, R3	: RESTORE R3
1786 036714	011000			MOV (R0), R0	: PICKUP EXIT ADDRESS
1787 036716	000200			RTS R0	: RETURN
1788					
1795					

```
1      .SBTTL SINGLE/DUAL PORT RH/RM DRIVER (REV 6.2) 1980
2
3      ;NEW DRIVE TYPE ID FOR RM02 *****
4      ;10-AUG-77 *****
5      ;10-MAR-78 THE SC, SC5 CHANGES
6      ;NEW DRIVE TYPE ID FOR RM05 *****
7      ;1980 *****
8
9      ;COPYRIGHT (C) 1977
10     ;DIGITAL EQUIPMENT CORP.
11     ;MAYNARD, MA 01754
12     ;AUTHOR(S): JIM LACEY/CHUCK HESS/
13
14     ;*****
15
16     ;STORAGE FOR RMDS, RMER1, RMER2, AND RMMR2 ON AN ERROR '2'
17     ;RMERRS = RMDS
18     ;RMERRS+2 = RMER1
19     ;RMERRS+4 = RMER2
20     ;RMERRS+6 = RMMR2
21
22 036720 000000 000000 000000 RMERRS: .WORD 0,0,0,0
23
24     ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
25     ;DRVACT=0 IF DRIVE IS IDLE
26     ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
27     ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
28
29 036730      000      DRVACT: .BYTE 0      ;DRIVE 0
30 036731      000      .BYTE 0      ;DRIVE 1
31 036732      000      .BYTE 0      ;DRIVE 2
32 036733      000      .BYTE 0      ;DRIVE 3
33 036734      000      .BYTE 0      ;DRIVE 4
34 036735      000      .BYTE 0      ;DRIVE 5
35 036736      000      .BYTE 0      ;DRIVE 6
36 036737      000      .BYTE 0      ;DRIVE 7
37
38     ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
39     ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
40     ;DRVSTA>0 IF DRIVE IS ONLINE
41     ;DRVSTA<0 IF DRIVE IS UNSAFE
42
43 036740      000      DRVSTA: .BYTE 0      ;DRIVE 0
44 036741      000      .BYTE 0      ;DRIVE 1
45 036742      000      .BYTE 0      ;DRIVE 2
46 036743      000      .BYTE 0      ;DRIVE 3
47 036744      000      .BYTE 0      ;DRIVE 4
48 036745      000      .BYTE 0      ;DRIVE 5
49 036746      000      .BYTE 0      ;DRIVE 6
50 036747      000      .BYTE 0      ;DRIVE 7
51
52     ;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
53     ;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
54     ;DRV TYP=7 IF DRIVE IS RM05 *****
55     ;DRV TYP=5 IF DRIVE IS RM02 *****
56     ;DRV TYP=4 IF DRIVE IS RM03
57     ;DRV TYP=-1 IF NOT RM05/3/2
```

```

50
51 036750      000      DRV TYP: .BYTE 0      ;DRIVE 0
54 036751      000      .BYTE 0      ;DRIVE 1
    036752      000      .BYTE 0      ;DRIVE 2
    036753      000      .BYTE 0      ;DRIVE 3
    036754      000      .BYTE 0      ;DRIVE 4
    036755      000      .BYTE 0      ;DRIVE 5
    036756      000      .BYTE 0      ;DRIVE 6
    036757      000      .BYTE 0      ;DRIVE 7
55
56      ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
57      ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
58      ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
59
60 036760      000      DPINT: .BYTE 0      ;DRIVE 0
63 036761      000      .BYTE 0      ;DRIVE 1
    036762      000      .BYTE 0      ;DRIVE 2
    036763      000      .BYTE 0      ;DRIVE 3
    036764      000      .BYTE 0      ;DRIVE 4
    036765      000      .BYTE 0      ;DRIVE 5
    036766      000      .BYTE 0      ;DRIVE 6
    036767      000      .BYTE 0      ;DRIVE 7
64
65      ;TABLE OF PENDING DUAL PORT REQUESTS
66      ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
67      ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
68
69 036770      000      DPRQS: .BYTE 0      ;DRIVE 0
72 036771      000      .BYTE 0      ;DRIVE 1
    036772      000      .BYTE 0      ;DRIVE 2
    036773      000      .BYTE 0      ;DRIVE 3
    036774      000      .BYTE 0      ;DRIVE 4
    036775      000      .BYTE 0      ;DRIVE 5
    036776      000      .BYTE 0      ;DRIVE 6
    036777      000      .BYTE 0      ;DRIVE 7
73
74      ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
75      ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
76      ;"DPB" OF THE I/O OPERATION.
77
78 037000      000000    TRNSWT: .WORD 0
79
80      ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
81      ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
82      ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
83      ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
84      ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
85
86 037002      000000    SRCHWT: .WORD 0
87
88      ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
89      ;ACTDRV=0 IF DRIVER IS INACTIVE
90      ;ACTDRV>0 IF DRIVER IS ACTIVE
91
92 037004      000      ACTDRV: .BYTE 0
93
94      ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
    
```

```

95                                     ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
96                                     ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
97
98 037005      000      ACTSTR: .BYTE  0
99
100                                     ;UNLOAD FLAG (ULDFLG=8 BYTES)
101                                     ;ULDFLG=0 IF NO UNLOAD COMMAND
102                                     ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
103                                     ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
104
105 037006      000      ULDFLG: .BYTE  0          ;DRIVE 0
108 037007      000          .BYTE  0          ;DRIVE 1
      037010      000          .BYTE  0          ;DRIVE 2
      037011      000          .BYTE  0          ;DRIVE 3
      037012      000          .BYTE  0          ;DRIVE 4
      037013      000          .BYTE  0          ;DRIVE 5
      037014      000          .BYTE  0          ;DRIVE 6
      037015      000          .BYTE  0          ;DRIVE 7
109
110                                     ;LOOK AHEAD COUNT (LACNT=8 BYTES)
111                                     ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
112
113 037016      000      LACNT:  .BYTE  0          ;DRIVE 0
116 037017      000          .BYTE  0          ;DRIVE 1
      037020      000          .BYTE  0          ;DRIVE 2
      037021      000          .BYTE  0          ;DRIVE 3
      037022      000          .BYTE  0          ;DRIVE 4
      037023      000          .BYTE  0          ;DRIVE 5
      037024      000          .BYTE  0          ;DRIVE 6
      037025      000          .BYTE  0          ;DRIVE 7
117
118                                     ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
119                                     ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
120                                     ;OPERATION IS COMPLETED AS PER (DPB+14).
121                                     ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
122                                     ;(DPB+14), AFTER AN ERROR.
123
124 037026      000000    SAVEFG: .WORD  0
125
126                                     ;SEEK FLAG (SEEKFG=1 WORD)
127                                     ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
128                                     ;FOR A DATA TRANSFER START A SEARCH COMMAND
129                                     ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
130                                     ;DISREGARD THE WINDOW
131
132 037030      177777    SEEKFG: .WORD  -1
133
134                                     ;TIMEOUT TABLE (TIMER=8 WORDS)
135                                     ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
136
137 037032      177777    TIMER:   .WORD  -1          ;DRIVE 0
140 037034      177777          .WORD  -1          ;DRIVE 1
      037036      177777          .WORD  -1          ;DRIVE 2
      037040      177777          .WORD  -1          ;DRIVE 3
      037042      177777          .WORD  -1          ;DRIVE 4
      037044      177777          .WORD  -1          ;DRIVE 5
      037046      177777          .WORD  -1          ;DRIVE 6
    
```

```

141 037050 177777          .WORD  -1          ;DRIVE 7
142
143      ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
144      ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
145      ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
146 037052 177777      DTUW:  .WORD  -1
147
148      ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
149      ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
150      ;ATTENTION BIT
151
152 037054      001      ATABIT:  .BYTE  1          ;DRIVE 0
153 037055      002      .BYTE  2          ;DRIVE 1
154 037056      004      .BYTE  4          ;DRIVE 2
155 037057      010      .BYTE 10         ;DRIVE 3
156 037060      020      .BYTE 20         ;DRIVE 4
157 037061      040      .BYTE 40         ;DRIVE 5
158 037062      100      .BYTE 100        ;DRIVE 6
159 037063      200      .BYTE 200        ;DRIVE 7
160
161      ;FSRM TO RH11/RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
162      ;CALLING IT FATAL (MCPEMX=1 WORD)
163
164 037064      000003    MCPEMX: .WORD  3
165
166      ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
167      ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
168
169 037066      176700    RMADR:  .WORD  176700
170 037070      000254 000240    RMVEC:  .WORD  254,5*32.
171
172      ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
173 037074      000004    MXLACT: .WORD  4
174
175      ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
176 037076      001000    MXDLTA: .WORD  8.*64.
177
178      ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
179 037100      000200    MNDLTA: .WORD  2*64.
180
181      ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
182 037102      000005    MXWNDW: .WORD  5
183
184      ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
185
186      000000    RMCS1=0          ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
187      000002    RMWC=2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
188      000004    RMBA=4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
189      000006    RMDA=6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
190      000010    RMCS2=10        ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
191      000012    RMD5=12         ;DRIVE STATUS REGISTER (DRIVE REG 01)
192      000014    RMER1=14        ;ERROR REGISTER #1 (DRIVE REG. 02)
193      000016    RMAS=16         ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
194      000020    RMLA=20         ;LOOK AHEAD REGISTER (DRIVE REG. 07)
195      000022    RMDB=22         ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
196      000024    RMMR1=24        ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
    
```



```

197      000026      RMDT=26      ;DRIVE TYPE REGISTER (DRIVE REG. 06)
198      000030      RMSN=30      ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
199      000032      RMOF=32      ;OFFSET REGISTER (DRIVE REG. 11)
200      000034      RMDC=34      ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
201      000036      RMHR=36      ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
202      000040      RMMR2=40     ;MAINTENANCE REGISTER #2
203      000042      RMER2=42     ;ERROR REGISTER #2 (DRIVE REG. 15)
204      000044      RMEC1=44     ;ECC POSITION REGISTER (DRIVE REG. 16)
205      000046      RMEC2=46     ;ECC PATTERN REGISTER (DRIVE REG. 17)
206
207      .SBTTL  RH/RM DRIVER INITIALIZATION CODE
208
209      ;THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
210      ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
211      ;TO THE PROPER STATE FOR EACH DRIVE.
212      ;NOTE: THIS ROUTINE CALLS DRVINT
213
214      ;CALL
215
216      ;       JSR      PC,RMINIT
217      ;       RETURN
218
219      ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
220
221 037104 104412      RMINIT: SAVREG      ;SAVE R0 - R5
222 037106 013746 177776      MOV      PS,-(SP)    ;SAVE THE PRESENT PROCESSOR STATUS
223 037112 012737 000240 177776      MOV      #<5+32.>,PS ;CHANGE THE PRIORITY TO 5
224 037120 004737 044772      JSR      PC,CLRQUE  ;CLEAR ALL REQUEST QUEUES
225 037124 012701 036720      MOV      #RMERRS,R1 ;FIRST ADDRESS TO BE CLEARED
226 037130 012702 037030      MOV      #SEEKFG,R2 ;LAST ADDRESS TO BE CLEARED
227 037134 005021      1$:      CLR      (R1)+      ;CLEAR
228 037136 020102      CMP      R1,R2      ;ARE WE DONE?
229 037140 103775      BLO     1$          ;BR IF NO
230 037142 012702 037052      MOV      #DTUW,R2   ;LAST ADDRESS
231 037146 012721 177777      2$:      MOV      #-1,(R1)+ ;INITIALIZE
232 037152 020102      CMP      R1,R2      ;DONE?
233 037154 101774      BLOS    2$          ;LOOP IF NO
234 037156 005037 036740      CLR      DRVSTA     ;SET ALL DRIVES TO OFFLINE
235 037162 005037 036742      CLR      DRVSTA+2
236 037166 005037 036744      CLR      DRVSTA+4
237 037172 005037 036746      CLR      DRVSTA+6
238 037176 013703 037070      MOV      RMVEC,R3   ;SETUP THE RH/RM VECTOR
239 037202 012723 041740      MOV      #ISR,(R3)+
240 037206 013713 037072      MOV      RMVEC+2,(R3)
241 037212 013704 037066      MOV      RMADR,R4   ;FIRST ADDRESS OF RH/RM
242 037216 012764 000040 000010      MOV      #CLR,RMCS2(R4) ;MASSBUS INIT
243 037224 005001      CLR      R1          ;START WITH DRIVE 0
244 037226 004037 037316      3$:      JSR      R0,DRVINT  ;INIT THE DRIVE
245 037232 000401      BR      4$          ;'DVA' NOT SET OR PARITY ERROR
246 037234 000402      BR      5$          ;NORMAL RETURN
247 037236 105061 036740      4$:      CLRB    DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
248 037242 005201      5$:      INC      R1          ;GO TO NEXT DRIVE
249 037244 042701 177770      BIC      #^C7,R1    ;MASK OUT UNUSED BITS
250 037250 001366      BNE     3$          ;BR IF MORE DRIVES TO GO
251 037252 012701 000007      MOV      #7,R1      ;START WITH DRIVE 7
252 037256 005037 177776      CLR      PS          ;CLEAR THE PROCESSOR STATUS
253 037262 105761 036760      6$:      TSTB    DPINT(R1)  ;WAITING FOR DRIVE TO SWITCH PORTS ?
    
```

```

254 037266 001405          BEQ      8$          ;BR NOT WAITING
255 037270 004737 044426   JSR      PC,SET.IE   ;SET INTERRUPT
256 037274 105761 036760   7$:    TSTB     DPINT(R1) ;DRIVE SWITCHED PORTS ?
257 037300 001375          BNE      7$          ;BR IF NOT
258 037302 005301          8$:    DEC      R1      ;GO TO THE NEXT DRIVE
259 037304 100366          BPL      6$          ;CHECK NEXT DRIVE
260 037306 012637 177776   MOV      (SP)+,PS    ;RESTORE THE PROCESSOR STATUS
261 037312 104413          RESREG   ;RESTORE R0 - R5
262 037314 000207          RTS      PC          ;BYE-BYE
263
264
265          ;DRIVE INITIALIZATION ROUTINE
266          ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
267          ;AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
268          ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
269          ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
270          ;DRVSTA IS SET TO THE PROPER CONDITION.
271          ;CALL
272          ;
273          ;
274          ;
275          ;
276          ;
277          ;
278 037316 010546          DRVINT: MOV     R5,-(SP)   ;SAVE R5
279 037320 105061 036740   DULP:  CLRB    DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
280 037324 105061 036750   CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
281 037330 105061 037006   CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
282 037334 010164 000010   MOV     R1,RMCS2(R4)    ;SELECT A DRIVE
283 037340 112764 000111 000000   MOVB   #11,RMCS1(R4)   ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
284 037346 032764 010000 000010   BIT    #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
285 037354 001403          BEQ     1$            ;NO
286 037356 004737 044426   JSR    PC,SET.IE      ;GO SET 'IE' WITHOUT A 'TRE'
287 037362 000520          BR     6$            ;LEAVE THIS ROUTINE
288
289 037364 105061 036740          1$:   CLRB    DRVSTA(R1)   ;SET DRIVE STATUS TO OFFLINE
290 037370 032764 004000 000000   BIT    #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
291 037376 001750          BEQ    DULP           ;BR IF DRIVE NOT AVAILABLE
292 037400 004037 043736   JSR    RO,RD.RM      ;READ THE DRIVE TYPE REG.
293 037404 000026          RMDT
294 037406 037650          8$
295 037410 012605          MOV    (SP)+,R5      ;ERROR RETURN ADDRESS
296 037412 112761 000004 036750   MOVB   #1,DRVTYP(R1) ;PUT DRIVE TYPE IN R5
297 037420 022705 020024          CMP    #24,R5        ;SET RM03 INDICATOR
298 037424 001431          BEQ    2$            ;SINGLE PORT RM03 ?
299 037426 022705 024024          BEQ    2$            ;BR IF YES
300 037432 001426          CMP    #24024,R5     ;DUAL PORT RM03 ?
301 037434 112761 000005 036750   BEQ    2$            ;BR IF YES
302 037442 022705 020025          MOVB   #5,DRVTYP(R1) ;SET RM02 INDICATOR
303 037446 001420          CMP    #20025,R5     ;SINGLE PORT RM02 ?
304 037450 022705 024025          BEQ    2$            ;BR IF SO
305 037454 001415          CMP    #24025,R5     ;DUAL PORT RM02 ?
306 037456 112761 000007 036750   BEQ    2$            ;BR IF SO
307 037464 022705 020027          MOVB   #7,DRVTYP(R1) ;SET RM05 INDICATOR
308 037470 001407          CMP    #20027,R5     ;SINGLE PORT RM05 ?
309 037472 022705 024027          BEQ    2$            ;BR IF YES
310 037476 001404          CMP    #24027,R5     ;DUAL PORT RM05 ?
          BEQ    2$            ;BR IF YES

```

```

311 037500 112761 177777 036750      MOVB  #-1,DRVTP(R1)  ;SET INDICATOR TO 'OTHER'
312 037506 000446                    BR    6$             ;EXIT
313
314 037510 012746 000121      2$:  MOV  #121,-(SP)    ;DO A 'READ-IN PRESET'
315 037514 004037 044116      JSR  RO,WRT.RM
316 037520 000000                    RMCS1
317 037522 037650                    8$
318 037524 012746 010000      MOV  #BIT12,-(SP)    ;SET FMT16=1
319 037530 004037 044116      JSR  RO,WRT.RM
320 037534 000032                    RMOF
321 037536 037650                    8$
322 037540 004037 043736      JSR  RO,RD.RM       ;READ RMD5
323 037544 000012                    RMD5
324 037546 037650                    8$
325 037550 012605                    MOV  (SP)+,R5       ;AND SAVE IT IN R5
326 037552 100015                    BPL  4$             ;BR IF ATA=0
327 037554 116164 037054 000016  MOVB  ATABIT(R1),RMA5(R4) ;CLEAR ATTENTION BIT
328 037562 004037 043736      JSR  RO,RD.RM       ;FIND OUT WHY ATA=1
329 037566 000014                    RMER1
330 037570 037650                    8$
331 037572 006126                    ROL  (SP)+          ;IS IT UNSAFE?
332 037574 100004                    BPL  4$             ;BR IF NOT
333 037576 112761 177777 036740  MOVB  #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
334 037604 000407                    BR    6$             ;EXIT
335
336 037606 005105      4$:  COM  R5           ;CHECK MOL, DPR, DRY, AND VV
337 037610 042705 167077      BIC  #*C<BIT12:BIT08:BIT07:BIT06>,R5
338 037614 001003                    BNE  6$             ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
339 037616 112761 000001 036740  MOVB  #1,DRVSTA(R1)  ;SET DRIVE STATUS TO ONLINE
340 037624 005720      6$:  TST  (R0)+        ;STEP OVER THE ERROR RETURN
341 037626 000410                    BR    8$             ;EXIT
342 037630 006301      7$:  ASL  R1           ;CHANGE INDEX TO ADDRESS WORDS
343 037632 012761 060000 037032  MOV  #60000,TIMER(R1) ;START 2 SEC TIMER
344 037640 006201                    ASR  R1             ;RESTORE R1
345 037642 112761 177777 036760  MOVB  #-1,DPINT(R1)  ;SET PORT INITIALIZE INDICATOR
346 037650 012605      8$:  MOV  (SP)+,R5       ;RESTORE R5
347 037652 000200                    RTS   RO           ;EXIT
348
349      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
350      ;
351      ;CALL
352      ;
353      ;
354      ;
355      ;
356      ;
357      ;
358      ;
359 037654 013746 177776      RM05: MOV  PS,-(SP)    ;SAVE THE CALLING STATUS
360 037660 013737 037072 177776  MOV  RMVEC+2,PS     ;DON'T ALLOW ANY RM INTERRUPTS
361 037666 112737 000001 037004  MOVB  #1,ACTDRV     ;SET 'ACTIVE DRIVER' FLAG
362 037674 104412                    SAVREG              ;SAVE R0 - R5
363 037676 011002                    MOV  (R0),R2       ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
364 037700 005062 000016      CLR  16(R2)        ;CLEAR THE STATUS/ERROR INDICATOR
365 037704 111201                    MOVB  (R2),R1      ;PICKUP THE DRIVE NUMBER
366 037706 013704 037066      MOV  RMADR,R4      ;UNIBUS ADDRESS OF RMCS1
367 037712 105761 036740      TSTB  DRVSTA(R1)   ;CHECK DRIVES STATUS

```

```

368 037716 003014          BGT      1$          ;BR IF ONLINE
369 037720 105761 037006   TSTB    ULDFLG(R1)  ;UNLOAD COMMAND IN QUEUE?
370 037724 001036          BNE     3$          ;BR IF YES
371 037726 105761 036760   TSTB    DPINT(R1)  ;TRYING TO INIT THE DRIVE
372 037732 001042          BNE     5$          ;BR IF YES
373 037734 004037 037316   JSR     RO,DRVINT  ;GO INIT. THE DRIVE
374 037740 000434          BR      4$          ;ERROR RETURN
375 037742 105761 036740   TSTB    DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
376 037746 003445          BLE     6$          ;BR IF NOT
377 037750 105761 036770   1$:    TSTB    DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
378 037754 001031          BNE     5$          ;BR IF YES
379 037756 010164 000010   MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
380 037762 004037 045070   JSR     RO,DRVQUE  ;PUT THIS REQUEST IN QUEUE
381 037766 000460          BR      9$          ;QUEUE IS FULL
382 037770 122762 000103 000002  CMPB    #103,2(R2)  ;IS THIS REQ. FOR AN UNLOAD?
383 037776 001003          BNE     2$          ;BR IF NO
384 040000 112761 177777 037006   MOVB    #-1,ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
385 040006 105761 036730   2$:    TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
386 040012 001043          BNE     8$          ;BR IF YES
387 040014 004737 040146   JSR     PC,OPT     ;CALL THE OPTIMIZER
388 040020 000440          BR      8$          ;
389 040022 012762 120000 000016  3$:    MOV     #BIT15 BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
390 040030 000434          BR      8$          ;EXIT
391 040032 004737 041226   4$:    JSR     PC,C17   ;GO HANDLE THE PARITY ERROR
392 040036 000431          BR      8$          ;
393 040040 004037 045070   5$:    JSR     RO,DRVQUE  ;PUT REQUEST IN QUEUE
394 040044 000431          BR      9$          ;QUEUE IS FULL
395 040046 032714 000100   BIT     #BIT06,(R4) ;IE BIT SET ?
396 040052 001023          BNE     8$          ;YES
397 040054 004737 044426   JSR     PC,SET.IE  ;SET THE INTERRUPT
398 040060 000420          BR      8$          ;RETURN
399 040062 105761 036740   6$:    TSTB    DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
400 040066 002412          BLT     7$          ;BR IF UNSAFE
401 040070 012762 140000 000016   MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
402 040076 105761 036750   TSTB    DRVTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
403 040102 001007          BNE     8$          ;BR IF OFFLINE
404 040104 012762 100002 000016   MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
405 040112 000403          BR      8$          ;GO TO EXIT
406 040114 012762 110000 000016  7$:    MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
407 040122 104413          8$:    RESREG   ;RESTORE R0 - R5
408 040124 005720          TST     (R0)+      ;SETUP FOR NORMAL RETURN
409 040126 000401          BR      10$       ;FINISH UP, THEN EXIT
410 040130 104413          9$:    RESREG   ;RESTORE R0 - R5
411 040132 005720          10$:   TST     (R0)+     ;CORRECT THE RETURN ADDRESS
412 040134 105037 037004   CLRB   ACTDRV     ;CLEAR 'ACTIVE DRIVER' FLAG
413 040140 012637 177776   MOV     (SP)+,PS   ;RETURN 'PS' TO USER LEVEL
414 040144 000200          RTS     RO        ;RETURN TO CALLER
415
416          .OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
417          :
418          :CALL
419          :
420          :      MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
421          :      JSR     PC,OPT         ;SETUP A COMMAND
422 040146 104412          OPT:   SAVREG   ;SAVE R0 - R5
423 040150 013746 177776   MOV     PS,-(SP)   ;SAVE PROC. STATUS
424 040154 146137 037054 037002  BICB   ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
    
```

```

425 040162 105061 036770      CLRB   DPRQS(R1)      ;RESET THE PORT REQ FLAG ****
426 040166 004737 045144      JSR    PC,GETREQ     ;GET 'DPB' POINTER OF REQUEST
427 040172 005702              TST    R2            ;IS THERE A REQUEST IN QUEUE?
428 040174 001472              BEQ    7$            ;NO--BR TO EXIT
429 040176 010164 000010      MOV    R1,RMCS2(R4)  ;LOAD THE DRIVE ADDRESS *****
430 040202 012764 000111 000000  MOV    #11,RMCS1(R4) ;CLEAR THE DRIVE
431 040210 032764 004000 000000  BIT    #BIT11,RMCS1(R4) ;DVA SET ?
432 040216 001446              BEQ    5$            ;TO PROT REQUEST ,IF NOT
433 040220 105761 036740      10$:  TSTB   DRVSTA(R1)    ;IS DRIVE ONLINE?
434 040224 003014              BGT    1$            ;YES
435 040226 004737 045166      JSR    PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
436 040232 012762 140000 000016  MOV    #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
437 040240 105761 036740      TSTB   DRVSTA(R1)    ;IS DRIVE UNSAFE ?
438 040244 100053              BPL    8$            ;BR TO EXIT IF NOT
439 040246 012762 110000 000016  MOV    #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
440 040254 000447              BR     8$            ;BR TO EXIT
441 040256      1$:
442      :
443      :
444      :
445      :
446      :
447      :
448 040256 122762 000150 000002  MOV    #11,-(SP)     ;LOAD COMMAND ONTO THE STACK
449 040264 002403              JSR    RO,WRT.RM     ;LOAD THE REGISTER
450 040266 004737 040612      RMCS1  ;REGISTER INCREMENT
451 040272 000440              6$
452 040274 005737 037052      BIT    #BIT11,(R4)   ;DRIVE AVAILABLE ?
453 040300 002012              BEQ    9$            ;BR IF NOT
454 040302 005737 037030      CMPB   #150,2(R2)    ;IS THE REQUEST FOR I/O?
455 040306 100404              BLT    2$            ;YES
456 040310 004037 041564      JSR    PC,C14        ;CALL THE COMMAND INITIATOR
457 040314 000427              BR     8$            ;BR TO EXIT
458 040316 000403              TST   DTUW           ;DATA TRANSFER UNDERWAY?
459 040320 004737 040404      2$:  BGE    4$            ;YES--GO START A SEARCH
460 040324 000423              TST   SEEKFG        ;DO IMPLIED SEEKS?
461 040326 004737 040512      BMI    3$            ;YES
462 040332 000420              JSR    RO,LA         ;NO--DO LOOK AHEAD
463 040334 112761 177777 036770  5$:  BR     8$            ;RETURN HERE ON A PARITY ERROR
464 040342 010103              BR     4$            ;GO START A SEARCH
465 040344 006303              JSR    PC,C11        ;START A DATA TRANSFER
466 040346 012763 060000 037032  4$:  BR     8$            ;START A SEARCH
467 040354 000402              BR     8$            ;GO TO THE EXIT
468 040356 004737 041226      6$:  MOVB   #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
469 040362 032714 000100      7$:  MOV    R1,R3         ;SET UP TO ADDRESS WORDS
470 040366 001002              ASL   R3             ;CONVERT TO WORD INDEX
471 040370 004737 044426      MOV    #60000,TIMER(R3) ;START 10 SEC TIMER
472 040374 012637 177776      8$:  BR     7$            ;EXIT
473 040400 104413              JSR    PC,C17        ;PROCESS THE PARITY ERROR
474 040402 000207              BIT    #BIT06,(R4)   ;SEE IF 'IE' ALREADY SET
475      :
476      :
477      :
478      :
479      :
480      :
481      :
      BNE   8$          ;BR IF SET
      JSR   PC,SET.IE  ;SET 'IE' WITHOUT A 'TRE'
      MOV   (SP)+,PS   ;RESTORE PROC. STATUS
      RESREG ;RESTORE R0 - R5
      RTS   PC
;COMMAND INITIATOR
;CALL
;MOV #DRVNUM,R1 ;DRIVE NUMBER
;MOV #DPB,R2 ;ADDRESS OF DPB
;JSR PC,C17 ;C17- C11,C13, OR C14
  
```

```

482
483
484
485
486
487 040404 004737 045166
488 040410 010237 037000
489 040414 010203
490 040416 013704 037066
491 040422 010164 000010
492 040426 062703 000004
493 040432 062704 000002
494 040436 012324
495 040440 012324
496 040442 012346
497 040444 004037 044116
498 040450 000006
499 040452 041226
500 040454 012346
501 040456 004037 044116
502 040462 000034
503 040464 041226
504 040466 016246 000002
505 040472 004037 044116
506 040476 000000
507 040500 041226
508 040502 010137 037052
509 040506 000137 041170
510 040512 013704 037066
511 040516 010164 000010
512 040522 016246 000012
513 040526 004037 044116
514 040532 000034
515 040534 041226
516 040536 116203 000010
517
518
519
520 040542 010346
521 040544 042716 177740
522 040550 116266 000011 000001
523 040556 004037 044116
524 040562 000006
525 040564 041226
526 040566 012746 000131
527 040572 004037 044116
528 040576 000000
529 040600 041226
530 040602 156137 037054 037002
531 040610 000567
532 040612 013704 037066
533 040616 010164 000010
534 040622 116203 000002
535 040626 122703 000131
536 040632 001007
537 040634 016246 000010
538 040640 004037 044116

```

```

:
:
:
:
:WHERE:
:C11=DATA TRANSFER
:C12=SEARCH REQUESTED BY DATA XFER
:C14=NOT DATA TRANSFER
:
C11: JSR PC,POPQUE ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
      MOV R2,TRNSWT ;PUT REQ. IN TRANSFER WAIT QUEUE
      MOV R2,R3 ;DPB ADDRESS TO R3
      MOV RMADR,R4 ;RMCS1 ADDRESS
      MOV R1,RMCS2(R4) ;SELECT DRIVE
      ADD #4,R3 ;DESIRED WORD COUNT
      ADD #2,R4 ;RMWC ADDRESS
      MOV (R3)+,(R4)+ ;LOAD WORD COUNT
      MOV (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
      MOV (R3)+,-(SP) ;LOAD SECTOR AND TRACK
      JSR R0,WRT.RM ;CALL THE LOAD(WRITE) ROUTINE
      RMDA ;INDEX OF REGISTER TO LOAD
      C17 ;ERROR RETURN ADDRESS
      MOV (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
      JSR R0,WRT.RM
      RMDC
      C17
      MOV 2(R2),-(SP) ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
      JSR R0,WRT.RM
      RMCS1
      C17
      MOV R1,DTUW ;SET 'DATA TRANSFER UNDERWAY'
      JMP C15
C13: MOV RMADR,R4 ;RMCS1 ADDRESS
      MOV R1,RMCS2(R4) ;SELECT DRIVE
      MOV 12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
      JSR R0,WRT.RM
      RMDC
      C17
      MOV 10(R2),R3 ;PICKUP SECTOR ADDRESS
      SLB MXWINDW,R3 ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
      BGE 1$
      ADD #32,R3
      MOV R3,-(SP) ;COMBINE THE ADJUSTED SECTOR WITH
      BIC #177740,(SP) ;CHOP OF ALL EXEPTED BITS ,IF ANY
      MOV 11(R2),1(SP) ;THE DESIRED TRACK
      JSR R0,WRT.RM ;LOAD DESIRED TRACK & SECTOR
      RMDA
      C17
      MOV #131,-(SP) ;START A SEARCH
      JSR R0,WRT.RM
      RMCS1
      C17
      BISB ATABIT(R1),SRCHWT ;SET 'SEARCH WAIT' KEY
      BR C15
C14: MOV RMADR,R4 ;RMCS1 ADDRESS
      MOV R1,RMCS2(R4) ;SELECT DRIVE
      MOV 2(R2),R3 ;PICKUP THE REQUESTED COMMAND
      CMPB #131,R3 ;IS IT A SEARCH COMMAND?
      BNE 1$ ;BR IF NO
      MOV 10(R2),-(SP) ;LOAD DESIRED TRACK & SECTOR
      JSR R0,WRT.RM

```

539	040644	000006			RMDA		
540	040646	041226			C17		
541	040650	000403			BR	2\$:GO LOAD CYLINDER
542	040652	122703	000105	1\$:	CMPB	#105,R3	:IS IT A SEEK COMMAND
543	040656	001007			BNE	3\$:BR IF NO
544	040660	016246	000012	2\$:	MOV	12(R2),-(SP)	:LOAD DESIRED CYLINDER
545	040664	004037	044116		JSR	RO,WRT.RM	
546	040670	000034			RMDC		
547	040672	041226			C17		
548	040674	000546			BR	C16	
549	040676	122703	000115	3\$:	CMPB	#115,R3	:IS IT AN 'OFFSET' COMMAND?
550	040702	001013			BNE	4\$:BR IF NO
551	040704	004037	043736		JSR	RO,RD.RM	:MERGE THE OFFSET VALUE INTO RMOF
552	040710	000032			RMOF		:BUT DON'T CHANGE THE UPPER
553	040712	041226			C17		
554	040714	116216	000001		MOVB	1(R2),(SP)	:BYTE WHEN LOADING THE
555	040720	004037	044116		JSR	RO,WRT.RM	:REGISTER (RMOF)
556	040724	000032			RMOF		
557	040726	041226			C17		
558	040730	000530			BR	C16	:GO START THE COMMAND
559	040732	122703	000107	4\$:	CMPB	#107,R3	:IS IT A 'RECALIBRATE' COMMAND?
560	040736	001525			BEQ	C16	:BR IF YES
561	040740	122703	000117		CMPB	#117,R3	:IS IT A RETURN TO CENTER?
562	040744	001522			BEQ	C16	:BR IF YES
563	040746	122703	000103		CMPB	#103,R3	:IS IT AN 'UNLOAD' COMMAND?
564	040752	001016			BNE	5\$:BR IF NO
565	040754	112761	000001	036730	MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE INDICATOR
566	040762	105061	036740		CLRB	DRVSTA(R1)	:PUT DRIVE STATUS TO OFFLINE
567	040766	112761	000001	037006	MOVB	#1,ULDFLG(R1)	:SET 'UNLOAD IN PROGRESS' FLAG
568	040774	010346			MOV	R3,-(SP)	:START THE 'UNLOAD' COMMAND
569	040776	004037	044116		JSR	RO,WRT.RM	
570	041002	000000			RMCS1		
571	041004	041226			C17		
572	041006	000207			RTS	PC	:RETURN TO USER
573	041010	122703	000143	5\$:	CMPB	#143,R3	:IS IT A 'SET FORMAT' COMMAND?
574	041014	001014			BNE	6\$:BR IF NO
575	041016	004037	043736		JSR	RO,RD.RM	:READ THE OFFSET REGISTER
576	041022	000032			RMOF		
577	041024	041226			C17		
578	041026	116266	000001	000001	MOVB	1(R2),1(SP)	:COMBINE 'FMT16','ECI', AND 'HCI'
579	041034	004037	044116		JSR	RO,WRT.RM	:LOAD 'FMT16','ECI', AND/OR 'HCI'.
580	041040	000032			RMOF		
581	041042	041226			C17		
582	041044	000436			BR	12\$	
583	041046	122703	000141	6\$:	CMPB	#141,R3	:IS IT A 'GET REGISTER' COMMAND?
584	041052	001023			BNE	10\$:BR IF NO
585	041054	016203	000006	7\$:	MOV	6(R2),R3	:POINTS TO 1ST ADDRESS OF WHERE
586							:TO PUT THE REGISTER(S)
587	041060	116237	000010	041076	MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.
588	041066	116205	000011		MOVB	11(R2),R5	:INDEX OF LAST REG. TO MOVE
589	041072	004037	043736	8\$:	JSR	RO,RD.RM	:READ RH/RM REGISTER
590	041076	000000		9\$:	RMCS1		:INDEX OF REG. TO READ
591	041100	041226			C17		
592	041102	012623			MOV	(SP)+,(R3)+	:GET THE CONTENTS OF RH/RM REG.
593	041104	023705	041076		CMP	9\$,R5	:LAST REG. BEEN READ?
594	041110	001414			BEQ	12\$:GET OUT IF YES
595	041112	062737	000002	041076	ADD	#2,9\$:INCREASE THE INDEX BY 2

```

596 041120 000764          BR      8$          ;LOOP--MORE TO READ
597 041122 122703 000145 10$:  CMPB   #145,R3      ;IS IT A "SELECT DRIVE" COMMAND?
598 041126 001405          BEQ     12$          ;BR IF YES
599 041130 010346          MOV     R3,-(SP)    ;LOAD THE COMMAND
600 041132 004037 0441'6  JSR    RO,WRT.RM
601 041136 000000          RMCS1
602 041140 041226          C17
603 041142 004737 045166  JSR    PC,POPQUE   ;REMOVE REQ. FROM QUEUE
604 041146 052762 000200 000016  BIS    #BIT07,16(R2) ;SET THE "DONE" BIT
605 041154 005737 037026  TST    SAVEFG     ;SAVE THE RH/RM REGISTERS?
606 041160 100002          BPL    13$          ;BR IF NO
607 041162 004737 044310  JSR    PC,SVRH70   ;YES--GO SAVE THE REGISTERS
608 041166 000207          RTS    PC          ;RETURN TO USER
609 041170 006301          C15:  ASL    R1
610 041172 012761 060000 037032  MOV    #60000,TIMER(R1) ;SET A ONE SECOND TIMER
611 041200 006201          ASR    R1
612 041202 112761 000001 036730  MOVB   #1,DRVACT(R1) ;SET THE DRIVE ACTIVE
613 041210 000207          RTS    PC          ;RETURN TO THE USER
614 041212 010346          C16:  MOV    R3,-(SP)    ;LOAD THE COMMAND
615 041214 004037 044116  JSR    RO,WRT.RM
616 041220 000000          RMCS1
617 041222 041226          C17
618 041224 000761          BR     C15
619 041226 032764 010000 000010  C17:  BIT    #BIT12,RMCS2(R4) ;DRIVE NON-EXISTENT ?
620          ;
621 041234 005702          ;:  BNE    C18          ;BR IF YES
622          ;
623 041236 001001          ;:  TST    R2          ;ANYTHING IN QUEUE ?
624 041240 000207          ;:  BEQ    C17B       ;BR IF NOT
625 041242 012762 104000 000016  ;:  BNE    2$          ;BR IF QUEUE IS THERE
626          ;
627 041250 012746 000111  ;:  RTS    PC          ;OTHERWISE EXIT
628 041254 004037 044116  C17B: MOV    #BIT15:BIT11,16(R2) ;SET "PARITY" ERROR INDICATOR
629 041260 000000          ;:  JSR    PC,SVRH70   ;GO SAVE THE RH/RM REGISTERS
630 041262 041326          ;:  MOV    #11,-(SP)   ;DO A "DRIVE CLEAR"
631 041264 004737 045050  ;:  JSR    RMCS1
632 041270 105061 036770  ;:  JSR    PC,EMPTYQ   ;EMPTY THE QUEUE
633 041274 105061 037006  CLRB   DPRQS(R1)   ;CLEAR THE PORT REQUEST FLAG
634 041300 105061 036730  CLRB   ULDFLG(R1)  ;CLEAR THE UNLOAD 'V' QUEUE FLAG
635 041304 020237 037000  CLRB   DRVACT(R1)  ;DRIVE IS IDLE
636          ;
637 041310 001005          ;:  CMP    R2,TRNSWT  ;IF THIS DRIVE HAD AN I/O REQUEST
638 041312 005037 037000  ;:  CMP    R1,DTUW    ;IF THIS DRIVE HAD AN I/O REQUEST
639 041316 012737 177777 037052  ;:  BNE    1$          ;IN PROGRESS CLEAR ALL OF THE FLAGS
640 041324 000207          ;:  CLR    TRNSWT
641 041326 104412          ;:  MOV    #-1,DTUW
642 041330 032764 010000 000010  C18:  SAVREG ;SAVE R0 - R5
643          ;
644 041336 005001          ;:  BIT    #BIT12,RMCS2(R4) ;IS "NED" SET ?
645 041340 005003          ;:  BNE    1$          ;BR IF YES
646 041342 105761 036730  ;:  CLR    R1
647          ;
648 041346 001003          ;:  CLR    R3
649 041350 105761 036770  ;:  TSTB   DRVACT(R1) ;DRIVE ACTIVE?
650 041354 001443          ;:  BEQ    5$          ;BR IF NO
651 041356 013702 037000  ;:  BNE    22$         ;BR IF IN ACTIVE
652 041362 020137 037052  ;:  TSTB   DPRQS(R1)  ;PORT REQUEST
        ;:  BEQ    5$          ;BR IF NOT
        ;:  MOV    TRNSWT,R2 ;GET THE "TRANSFER WAIT" QUEUE
        ;:  CMP    R1,DTUW  ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?

```



```

653 041366 001402          BEQ      2$          ;BR IF YES
654 041370 004737 045144   JSR      PC,GETREQ   ;GET THE DPB POINTER
655 041374 005702          TST      R2          ;QUEUE ENTRY FOR DRIVE ?
656 041376 001413          BEQ      4$          ;BR IF NOT
657 041400 032764 010000 000010 BIT      #BIT12,RMCS2(R4) ;'NED' SET ?
658 041406 001404          BEQ      3$          ;BR IF NOT
659 041410 012762 100002 000016 MOV      #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
660 041416 000403          BR       4$          ;CONTINUE
661 041420 012762 102000 000016 3$: MOV      #BIT15!BIT10,16(R2) ;SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
662          ;
663 041426 012763 177777 037032 4$: JSR      PC,SVRH70   ;SAVE RH/RM REGISTERS
664 041434 105061 036730   MOV      #-1,TIMER(R3) ;STOP THE TIMER
665 041440 105061 036770   CLRB    DRVACT(R1)    ;SET 'DRIVE ACTIVE' TO IDLE
666 041444 020137 037052   CLRB    DPRQS(R1)    ;CLEAR PORT REQUEST FLAG
667 041450 001005          CMP      R1,DTUW     ;IS THIS DRIVE SETUP FOR A TRANSFER
668 041452 012737 177777 037052 BNE     5$          ;BR IF NOT
669 041460 005037 037000   MOV      #-1,DTUW   ;RESET THE INDICATOR
670 041464 105061 037006 5$: CLR      TRNSWT     ;CLEAR THE TRANSFER QUEUE
671 041470 032764 010000 000010 CLRB    ULDFLG(R1)   ;CLEAR UNLOAD FLAG
672          ;
673 041476 005201          BNE     6$          ;BR IF YES
674 041500 062703 000002   INC     R1          ;MOVE TO THE NEXT DRIVE
675 041504 042701 177770   ADD     #2,R3
676 041510 001314          BIC     #^C7,R1
677 041512 012737 177777 037052 BNE     1$          ;BR IF MORE DRIVES
678 041520 005037 037000   MOV      #-1,DTUW   ;NO DATA TRANSFERS UNDERWAY
679 041524 004737 044772   CLR     TRNSWT     ;CLEAR THE 'TRANSFER WAIT' QUEUE
680 041530 012764 000040 000010 JSR     PC,CLRQUE   ;CLEAR ALL OF THE REQUEST QUEUES
681 041536 000406          MOV      #CLR,RMCS2(R4) ;DO A MASSBUS INIT.
682 041540 004737 045050 6$: BR       7$          ;CONTINUE
683 041544 105061 036740   JSR     PC,EMPTYQ   ;CLEAR THE DRIVE'S QUEUE
684 041550 105061 036750   CLRB    DRVSTA(R1)  ;SET DRIVE TO OFFLINE
685 041554 004737 044426 7$: CLRB    DRVTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
686 041560 104413          JSR     PC,SET.IE   ;SET 'IE' WITHOUT 'TRE'
687 041562 000207          RESREG ;RESTORE R0 - R5
688          RTS      PC ;RETURN
689          ;LOOK AHEAD ROUTINE
690          ;CALL
691          ;
692          ;
693          ;
694          ;
695          ;
696          ;
697          ;
698          ;
699 041564 013704 037066 LA: MOV     RMADR,R4   ;GET RMCS1'S ADDRESS
700 041570 010164 000010   MOV     R1,RMCS2(R4) ;SELECT DRIVE
701 041574 004037 043736   JSR     R0,RD.RM    ;READ DRIVE STATUS
702 041600 000012          RMDS   4$          ;ERROR RETURN ADDRESS
703 041602 041732          BIC     #^C020200,(SP) ;ON CYLINDER ?
704 041604 042716 157577   CMP     #200,(SP)+  ;PIP=0,DRY=1?
705 041610 022726 000200   BNE     3$          ;NO
706 041614 001044          INCB   LACNT(R1)    ;INCREMENT THE LOOK AHEAD COUNT
707 041616 105261 037016   CMPB   LACNT(R1),MXLACT ;EXCEED MAX?
708 041622 126137 037016 037074 BGT     2$          ;BR IF YES
709 041630 003033          ;
    
```

```

710 041632 116203 000010      MOVB    10(R2),R3      ;GET DESIRED SECTOR ADDRESS AND
711 041636 000303              SWAB     R3           ;MULT. BY 64--ALIGN WITH
712 041640 006203              ASR     R3           ;LOOK AHEAD REGISTER
713 041642 006203              ASR     R3
714 041644 012737 000340 177776  MOV     #340,PS      ;PRIORITY LEVEL '7'
715 041652 004037 043736 6$:   JSR     R0,RD.RM     ;READ LOOK AHEAD REGISTER
716 041656 000020              RMLA    4$
717 041660 041732              4$
718 041662 021664 000020      CMP     (SP),RMLA(R4) ;CORRECT LA NUMBER ?
719 041666 001402              BEQ     7$           ;YES
720 041670 005726              TST     (SP)+        ;NO,CLEAR STACK
721 041672 000415              BR      3$
722 041674 162603 7$:   SUB     (SP)+,R3     ;CALCULATE THE DELTA
723 041676 002002              BGE     1$
724 041700 062703 004000      ADD     #<32.*64.>,R3 ;MAKE THE DELTA POSITIVE
725 041704 023703 037076 1$:   CMP     MXDLTA,R3   ;CHECK THE DELTA TO SEE
726 041710 002406              BLT     3$           ;IF IT IS WITHIN THE
727 041712 023703 037100      CMP     MNDLTA,R3   ;WINDOW---IF YES, ZERO
728 041716 002003              BGE     3$           ;THE LOOK AHEAD COUNT
729 041720 105061 037016 2$:   CLRB   LACNT(R1)   ;AND TAKE THE I/O EXIT
730 041724 005720              TST     (R0)+
731 041726 005720 3$:   TST     (R0)+        ;ADJUST THE RETURN ADDRESS
732 041730 000002              BR      5$           ;EXIT
733 041732 004037 041226 4$:   JSR     PC,C17      ;PROCESS THE ERROR
734 041736 000200 5$:   RTS     R0         ;RETURN
735
736 ;INTERRUPT SERVICE ROUTINE
737
738 041740 112737 000001 037004 ISR:  MOVB   #1,ACTDRV    ;SET 'ACTIVE DRIVER' FLAG
739 041746 104412              SAVREG              ;SAVE R0 - R5
740 041750 013704 037066      MOV    RMADR,R4     ;ADDRESS OF RHSCS1
741 041754 013701 037052      MOV    DTUW,R1      ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
742 041760 002403              BLT    1$           ;BR IF NO DATA TRANSFER UNDERWAY
743 041762 004737 042004      JSR    PC,TD        ;CALL TRANSFER DONE
744 041766 000402              BR     2$           ;EXIT
745 041770 004737 042154 1$:   JSR    PC,SC        ;CALL SPECIAL CONDITIONS
746 041774 104413 2$:   RESREG              ;RESTORE R0 - R5
747 041776 105037 037004      CLRB   ACTDRV      ;CLEAR 'ACTIVE DRIVER' FLAG
748 042002 000002              RTI
749
750 ;TRANSFER DONE ROUTINE
751
752 042004 105061 036730 037052 TD:  CLRB   DRVACT(R1)  ;SET DRIVE ACTIVE INDICATOR TO IDLE
753 042010 012737 177777 037052  MOV    #-1,DTUW     ;NO DATA TRANSFERS UNDERWAY
754 042016 006301              ASL    R1
755 042020 012761 177777 037032  MOV    #-1,TIMER(R1) ;CANCEL TIMEOUT
756 042026 006201              ASR    R1
757 042030 013702 037000      MOV    TRNSWT,R2   ;GET 'DPB' ADDRESS FROM THE
758 042034 005037 037000      CLR    TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
759 042040 052762 000200 000016  BIS    #BIT07,16(R2) ;SET DONE
760 042046 010164 000010      MOV    R1,RMCS2(R4) ;SELECT THE DRIVE
761 042052 004037 043736      JSR    R0,RD.RM     ;TRANSFER ERROR(TRE-1)?
762 042056 000000              RMCS1
763 042060 041226              C17
764 042062 006126              ROL    (SP)+
765 042064 100417              BMI    3$           ;BR IF YES
766 042066 005737 037026      TST    SAVEFG      ;SAVE THE RH/RM REGISTERS?

```

```

767 042072 100002          BPL      1$          ;BR IF NO
768 042074 004737 044310   JSR      PC,SVRH70    ;YES--SAVE THE REGISTERS
772 042100          1$:
774 042100 004737 045144   JSR      PC,GETREQ    ;GET DPB POINTER
775 042104 005702          TST      R2          ;ENTRY FOR DRIVE ?
776 042106 001403          BEQ      2$          ;BR IF NOT
777 042110 004737 040146   JSR      PC,OPT      ;CALL OPTIMIZER
778 042114 000417          BR       SC          ;CHECK OTHER DRIVES
779          ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
780 042116 012714 000113   2$:  MOV     #113,(R4)  ;RELEASE THE DRIVE
781 042122 000414          BR       SC          ;CHECK FOR OTHER DRIVES
782 042124 052762 100100 000016 3$:  BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
783 042132 004737 045050   JSR      PC,EMPTYQ   ;EMPTY THE 'DRIVE'S WAIT' QUEUE
784 042136 004737 044310   JSR      PC,SVRH70   ;SAVE THE RH/RM REGISTERS
785 042142 012714 040111   MOV     #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
786 042146 012714 000113   MOV     #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
787 042152 000400          BR       SC          ;CHECK FOR OTHER DRIVES
788
789
806
807          ;SPECIAL CONDITION ROUTINE
808
809 042154 116403 000016   SC:  MOVB   RMAS(R4),R3 ;READ 'RMAS'
810 042160 001014          BNE     2$          ;BR IF ANY 'ATA' BITS SET
811 042162 004037 043736   JSR      RO,RD.RM    ;READ CONTROL AND STATUS REGISTER
812 042166 000000          RMCS1
813 042170 041326          C18
814          ;
815 042172 106126          ;EXIT IF FAIL TO READ
816 042174 100405          ROLB   (SP)+        ;IS 'IE'=1?
817 042176 004037 045234   BMI    1$          ;YES, NO DRIVES TO CHECK
818 042204 004737 044426   JSR      RO,ES.SAV   ;SAVE THE ADDRESS IN '$ESCAPE'
819 042210 000207          EMT     1           ;REPORT AN ILLEGAL INTERRUPT
820 042212 005046          JSR      PC,SET.IE  ;SET INTERRUPT ENABLE
821 042214 110316          1$:  RTS     PC          ;RETURN
822 042216 012703 000001   2$:  CLR     -(SP)     ;PROCESS ALL DRIVES THAT HAVE
823 042222 005001          MOVB   R3,(SP)     ;AN 'ATA'=1
824 042224 030316          CLR     #1,R3
825 042226 001005          SC3:  BIT     R3,(SP) ;ATA=1?
826 042230 005201          BNE    SC5         ;YES
827 042232 106303          SC4:  INC     R1     ;MOVE TO THE NEXT DRIVE
828 042234 001373          ASLB   R3
829 042236 005726          BNE    SC3         ;BR IF MORE TO CHECK?
830 042240 000207          TST   (SP)+        ;CLEAN OFF THE STACK
831 042242 105761 036760   SC5:  RTS     PC          ;RETURN TO USER
832 042246 001402          TSTB  DPINT(R1)    ;INITIALIZING THE DRIVE ?
833 042250 000137 043166   BEQ    1$          ;BR IF NOT
834 042254 105761 036770   1$:  JMP     SC13       ;PROCESS THE DRIVE
835 042260 001402          TSTB  DPRQS(R1)    ;PORT REQUEST OUTSTANDING ?
836 042262 000137 043166   BEQ    2$          ;BR IF NOT
837 042266 105761 036740   2$:  JMP     SC13       ;START THE OUTSTANDING COMMAND
838 042272 003023          TSTB  DRVSTA(R1)   ;CHECK THE DRIVE STATUS
839 042274 105761 037006   BGT    5$          ;BR IF ONLINE
840 042300 003420          TSTB  ULDFLG(R1)  ;UNLOAD IN PROGRESS?
841 042302 004737 045144   BLE    5$          ;BR IF NOT
842 042306 004737 044310   JSR      PC,GETREQ  ;GET DPB POINTER
                          JSR      PC,SVRH70  ;SAVE THE RH/RM REGISTERS

```

```

843 042312 004737 043115      JSR      PC,SC12      ;SAVE RMD5, RMER1, RMER2, AND RMMR2
844                                ;ALSO DO A DRIVE INIT (DRVINT)
845 042316 105761 036740      TSTB     DRVSTA(R1)  ;DID DRIVE COME ONLINE?
846 042322 003414                BLE      6$          ;NO
847 042324 032737 040000 036720 BIT      #BIT14,RMERRS ;WAS THERE AN ERROR?
848 042332 001000                BNF      3$          ;BR IF ERROR
849                                JMP      SC11         ;NO ERROR
850 042334 013705 036722 3$:   MOV      RMERRS+2,R5 ;YES -- PICKUP RMER1 AND
851 042340 000504                BR       SC6A        ;GO PROCESS THE ERROR
852 042342 105761 036730 5$:   TSTB     DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
853 042346 001033                BNE      SC6         ;BR IF EITHER
854 042350 004737 043116      JSR      PC,SC12      ;SAVE RMD5, RMER1, RMER2, AND RMMR2
855                                ;ALSO DO A DRVINT
856 042354 105761 036760 6$:   TSTB     DPINT(R1)   ;TRYING TO INIT THE DRIVE ?
857 042360 001323                BNE      SC4         ;BR IF YES, CHECK ON MORE DRIVES
858 042362 105761 036740      TSTB     DRVSTA(R1) ;CHECK ON DRIVE'S STATUS
859 042366 100412                BMI      7$          ;BR IF UNSAFE
860 042370 032737 020000 036724 BIT      #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
861 042376 001013                BNE      8$          ;BR IF YES
862                                ;
863 042400 012746 000111      MOV      #113,-(SP)   ;RELEASE COMMAND
864 042404 004037 044116      MOV      #111,-(SP)   ;DRIVE CLEAR
865 042410 000000                JSR      RO,WRT.RM    ;WRITE THE COMMAND INTO RMCS1
866 042412 042756                RMCS1          ;REGISTER INDEX
867 042414 011605                SC8           ;PARITY EXIT ADDRESS
868 042416 004037 045234 7$:   MOV      (SP),R5     ;PICKUP (RMAS) BEFORE THE ERROR CALL
869 042424 000701                JSR      RO,ES.SAV    ;SAVE THE ADDRESS IN '$ESCAPE'
870 042426 000701                EMT      2          ;REPORT THE UNEXPECTED ATTENTION
871 042426 000701                BR       SC4         ;GO CHECK FOR MORE ATA'S
872 042426 004037 045234 8$:   JSR      RO,ES.SAV    ;SAVE THE ADDRESS IN '$ESCAPE'
873 042432 104005                EMT      5          ;REPORT THE ADDRESS PLUG CHANGE
874 042434 000675                BR       SC4         ;CHECK FOR MORE DRIVES
875 042436 006301 037032 5C6:  ASL      R1          ;SETUP TO ADDRESS WORDS
876 042440 012761 177777 037032 MOV      #-1,TIMER(R1) ;STOP THE TIMER
877 042446 006201                ASR      R1          ;RESTORE THE DRIVE ADDRESS
878 042450 004737 045144      JSR      PC,GETREQ    ;GET THE DPB POINTER FROM THE QUEUE
879 042454 010164 000010      MOV      R1,RMCS2(R4) ;SELECT DRIVE
880 042460 000137 043006      JMP      SC11         ;PROCESS THE SEARCH
881 042464 004037 043736      JSR      RO,RD.RM     ;READ THE RM'S STATUS REG.
882 042470 000012                RMD5          ;
883 042472 042756                SC8           ;
884 042474 011605                MOV      (SP),R5     ;AND PUT IT IN R5
885 042476 006126                ROL      (SP)+       ;WAS THERE AN ERROR?
886 042500 100407                BMI      1$          ;BR IF ERROR
887 042502 105761 036730      TSTB     DRVACT(R1)  ;CHECK DRIVE'S STATE
888 042506 003137                BGT      SC11        ;BR IF DRIVE ACTIVE WITH ORDER
889 042510 052762 100210 000016 BIS      #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
890 042516 000470                BR       SC7         ;
891 042520 004037 043736 1$:   JSR      RO,RD.RM     ;READ ERROR REGISTER #1
892 042524 000014                RMER1         ;
893 042526 042756                SC8           ;
894 042530 012605                MOV      (SP)+,R5    ;AND SAVE IT IN R5
895 042532 004737 044310      JSR      PC,SVRH70    ;SAVE RH/RM REGISTERS
896 042536 012746 000111      MOV      #111,-(SP)  ;ISSUE A DRIVE CLEAR
897 042542 004037 044116      JSR      RO,WRT.RM    ;
898 042546 000000                RMCS1         ;
899 042550 042756                SC8           ;

```

```

897 042552 006105          SC6A:  ROL    R5          ;WAS 'UNSAFE' CONDITION =1?
898 042554 100406          BMI    1$          ;BR IF YES
899 042556 005702          TST   R2          ;ANYTHING IN QUEUE ?
900 042560 001447          BEQ   SC7          ;BR IF NOT
901 042562 052762 100240 000016  BIS   #BIT15!BIT07.BIT05,16(R2) ;INFORM USER OF ERROR
902 042570 000443          BR    SC7
903 042572 004037 043736 1$:  JSR   RO,RD.RM    ;READ DRIVE STATUS REG. #1
904 042576 000012          RMDS
905 042600 042756          SC8
906 042602 011605          MOV   (SP),R5     ;SAVE RMDS IN R5
907 042604 006126          ROL   (SP)+       ;'ERR'=1?
908 042606 100011          BPL   2$          ;BR IF NO--UNSAFE CLEARED
909 042610 112761 177777 036740  MOVB  #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
910 042616 004737 044310  JSR   PC,SVRH70   ;SAVE RH/RM REGISTERS
911 042622 052762 110000 000016  BIS   #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
912 042630 000423          BR    SC7
913 042632 032705 010000 2$:  BIT   #BIT12,R5    ;'MOL' = 1 ?
914 042636 001015          BNE   3$          ;BR IF YES
915 042640 112761 177777 036730  MOVB  #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
916 042646 112761 000001 036740  MOVB  #1,DRVSTA(R1) ;ONLINE
917 042654 006301          ASL   R1
918 042656 012761 072460 037032  MOV   #30000.,TIMER(R1) ;START 30 SECOND TIMER
919 042664 006201          ASR   R1
920 042666 000137 042230  JSR   SC4
921 042672 052762 100220 000016 3$:  BIS   #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
922 042700 105061 036730  SC7:  CLRB  DRVACT(R1)  ;DRIVE IS IDLE
923          ; JSR   PC,EMPTYQ ;DUMP THE QUEUE
924 042704 004737 045166  JSR   PC,POPQUE  ;REMOVE THE QUFLE
925 042710 105761 037006  TSTB  ULDFLG(R1) ;UNLOAD IN RMOGRESS OR QUEUE?
926 042714 003002          BGT   1$          ;BR IF NOT
927 042716 105061 037006  CLRB  ULDFLG(R1)  ;CLEAR UNLOAD FLAG
928 042722 116164 037054 000016 1$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
929 042730 105761 036740  TSTB  DRVSTA(R1)  ;IS THE DRIVE UNSAFE ?
930 042734 100406          BMI   2$          ;BR IF IT IS
931          ; MOV   #113,-(SP) ;RELEASE COMMAND
932 042736 012746 000111  MOV   #111,-(SP) ;DRIVE CLEAR COMMAND
933 042742 004037 044116  JSR   RO,WRT.RM  ;WRITE THE COMMAND INTO RPCS1
934 042746 000000          RMCS1
935 042750 042756          SC8
936 042752 000137 042230 2$:  JMP   SC4
937 042756 105761 036730  SC8:  TSTB  DRVACT(R1)  ;IS DRIVE IDLE?
938 042762 001405          BEQ   1$          ;YES
939 042764 004737 045144  JSR   PC,GETREQ  ;GET DPB POINTER
940 042770 004737 041226  JSR   PC,C17     ;PROCESS THE PARITY ERROR
941 042774 000402          BR    2$          ;CONTINUE
942 042776          1$:
943          ; JSR   PC,C17 ;PROCESS THE PARITY ERROR
944 042776 004737 041250  JSR   PC,C17B    ;PROCESS THE UNCORRECTABLE PARITY ERROR
945 043002 000137 042230 2$:  JMP   SC4
946 043006 105761 037006  SC11: TSTB  ULDFLG(R1) ;'UNLOAD IN PROGRESS'?
947 043012 003402          BLE   1$          ;BR IF NO
948 043014 105061 037006  CLRB  ULDFLG(R1)  ;CLEAR UNLOAD FLAG
949 043020 105061 036730 1$:  CLRB  DRVACT(R1)  ;SET DRIVE IDLE
950 043024 136137 037054 037002  BITB  ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
951          ; AN I/O COMMAND?
952 043032 001012          RNE   2$          ;BR IF YES
953 043034 004737 045166  JSR   PC,POPQJE ;REMOVE REQUEST FROM QUEUE
  
```

```

954 043040 052762 000200 000016      BIS      #BIT07,16(R2)      ;SET 'DONE' BIT
955 043046 005737 037026      TST      SAVEFG          ;SAVE THE REGISTERS?
956 043052 100002      BPL      2$              ;BR IF NO
957 043054 004737 044310      JSR      PC,SVRH70       ;YES--SAVE ALL OF THE RH/RM REG'S
958 043060 116164 037054 000016 2$:      MOVB     ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
959 043066 146137 037054 037002      BICB     ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
960 043074 006301      ASL      R1              ;WORD INDEX
961 043076 012761 177777 037032      MOV      #-1,TIMER(R1)   ;STOP CLOCK
962 043104 006201      ASR      R1              ;RESTOPE R1
963 043106 004737 040146      JSR      PC,OPT          ;START A REQUEST
964 043112 000137 042230      JMP      SC4             ;CHECK FOR MORE DRIVES
965 043116 010164 000010      SC12:    MOV      R1,RMCS2(R4)   ;SELECT DRIVE
966 043122 016437 000012 036720      MOV      RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
967 043130 016437 000014 036722      MOV      RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
968 043136 016437 000042 036724      MOV      RMER2(R4),RMERRS+4
969 043144 016437 000040 036726      MOV      RMMR2(R4),RMERRS+6
970 043152 004037 037316      JSR      RO,DRVINT       ;INIT. THE STATE OF THE DRIVE
971 043156 000401      BR       1$             ;TAKE ERROR EXIT
972 043160 000207      RTS      PC              ;RETURN
973 043162 005726      1$:      TST      (SP)+         ;POP PC OFF OF THE STACK
974 043164 000674      BR       SC8            ;PROCESS THE PARITY ERROR
975 043166 006301      SC13:    ASL      R1              ;SETUP TO ADDRESS WORDS
976 043170 012761 177777 037032      MOV      #-1,TIMER(R1)   ;STOP THE TIMER
977 043176 006201      ASR      R1              ;
978 043200 010164 000010      MOV      R1,RMCS2(R4)   ;SELECT THE DRIVE
979 043204 116164 037054 000016      MOVB     ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
980 043212 105761 036760      1$:      TSTB     DPINT(R1)       ;INITIALIZING THE DRIVE ?
981 043216 001424      BEQ      2$             ;BR IF NOT
982 043220 105061 036760      CLRB     DPINT(R1)       ;CLEAR THE INIT INDICATOR
983 043224 004037 037316      JSR      RO,DRVINT       ;GO INIT THE DRIVE
984 043230 000240      NOP                          ;DUMMY PARITY ERROR RETURN
985 043232 105761 036740      TSTB     DRVSTA(R1)      ;DRIVE ONLINE ?
986 043236 003014      BGT      2$             ;BR IF YES -- START ORDER
987 043240 005702      TST      R2              ;QUEUE ENTRY FOR THE DRIVE
988 043242 001426      BEQ      3$             ;BR IF NOT
989 043244 004737 045144      JSR      PC,GETREQ       ;GET DPB ADDRESS
990 043250 052772 140000 000016      BIS      #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
991 043256 004737 044310      JSR      PC,SVRH70       ;SAVE THE REGISTERS
992      ;      JSR      PC,EMPTYQ     ;EMPTY THE REQUEST QUEUE
993 043262 004737 045166      JSR      PC,POPQUE       ;REMOVE THE QUEUE
994 043266 000414      BR       3$
995 043270 032764 064000 000000 2$:      BIT      #BIT11,RMCS1(R4) ;DVA SET ?
996 043276 001006      BNE     4$              ;SET THEN CALL OPT
997 043300 006301      ASL      R1
998 043302 012761 060000 037032      MOV      #60000,TIMER(R1)
999 043310 006201      ASR      R1
1000 043312 000402      BR       3$
1001 043314 004737 040146      4$:      JSR      PC,OPT          ;START THE PENDING REQUEST
1002 043320 000137 042230      3$:      JMP      SC4             ;PROCESS OTHER DRIVES
1003
1004      ;RM TIMER ROUTINE
1005      ;CALL
1006      ;      MOV      #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
1007      ;      JSR      PC,RMTMR   ;CALL RM05 TIME ROUTINE
1008
1009 043324 005737 037004      RMTMR:  TST      ACTDRV       ;CHECK 'ACTDRV & ACTSTR'
1010 043330 001027      BNE     4$              ;IF NON ZERO EXIT

```

```

1011 043332 112737 000001 037005      MOVB   #1,ACTSTR      ;SET 'ACTSTR'
1012 043340 104412                SAVREG                ;SAVE R0 - R5
1013 043342 005001                CLR    R1              ;START WITH DRIVE 0
1014 043344 005003                CLR    R3
1015 043346 005763 037032          1$:   TST   TIMER(R3)   ;IS THE TIMER RUNNING?
1016 043352 002406                BLT    2$              ;BR IF NO
1017 043354 166663 000002 037032          SUB    2(SP),TIMER(R3);COUNT THE INTERVAL
1018 043362 003002                BGT    2$              ;BR IF NO SOFTWARE TIMEOUT
1019 043364 004737 043414          JSR    PC,STO         ;CALL SOFTWARE TIMEOUT ROUTINE
1020 043370 005201                2$:   INC    R1          ;MOVE TO NEXT DRIVE
1021 043372 005723                TST   (R3)+
1022 043374 022701 000010          CMP    #8.,R1         ;OUT OF DRIVES?
1023 043400 003362                BGT    1$             ;BR IF NO
1024 043402 104413                3$:   RESREG                ;RESTORE R0 - R5
1025 043404 105037 037005          CLRB  ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1026 043410 012616                4$:   MOV    (SP)+,(SP)   ;ADJUST THE STACK
1027 043412 000207                RTS    PC              ;RETURN
1028
1029                ;SOFTWARE TIMEOUT ROUTINE
1030
1031                ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1032                ;OR GREATER
1033
1034                ;CALL:
1035                ;   STO
1036                ;   MOV    #DRVNUM,R1      ;DRIVE NUMBER
1037                ;   JSR    PC,STO         ;CALL
1038                ;   RETURN
1039 043414 010146                STO:   MOV    R1,-(SP)   ;SAVE R1
1040 043416 010246                MOV    R2,-(SP)   ;SAVE R2
1041 043420 010346                MOV    R3,-(SP)   ;SAVE R3
1042 043422 010446                MOV    R4,-(SP)   ;SAVE R4
1043 043424 013704 037066                MOV    RMADR,R4    ;GET ADDRESS OF 'RMCS1'
1044 043430 010164 000010                MOV    R1,RMCS2(R4);SELECT THE DRIVE
1045 043434 004037 043736                JSR    R0,RD.RM    ;READ 'DRIVE STATUS REG'
1046 043440 000012                RMDS
1047                ;
1048 043442 043724                STOS
1049 043444 105726                STO9
1050 043446 100436                TSTB  (SP)+        ;IS 'DRY'=1?
1051 043450 105761 036760                BMI   STO2         ;BR IF YES
1052 043454 001033                STO1: TSTB  DPINT(R1) ;TRYING TO INTIALIZE THE DRIVE ?
1053 043456 105761 036770                BNE  STO2         ;BR IF YES
1054 043462 001030                TSTB  DPRQS(R1)   ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1055 043464 013702 037000                BNE  STO2         ;BR IF YES
1056 043470 020137 037052                MOV   TRNSWT,R2   ;PICKUP TRANSFER WAIT QUEUE
1057 043474 001404                CMP   R1,DTUW     ;TRANSFER UNDERWAY ON THIS DRIVE?
1058 043476 000137 043724                BEQ   1$          ;BR IF YES
1059 043502 004737 045144                JMP   STO9        ;IF NOT DON'T BOTHER DRIVES
1060 043506 052762 101000 000016 1$:   JSR   PC,GETREQ   ;GET DPB ADDRESS
1061 043514 004737 044310                JSR   PC,SVRH70   ;SAVE RH/RM REGISTERS
1062                ;
1063 043520 105061 036730                MOV   #CLR,RMCS2(R4);'INIT' THE MASS BUS
1064 043524 105061 037006                CLRB  DRVACT(R1)  ;DRIVE IS IDLE
1065 043530 005037 037000                CLRB  ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
1066 043534 012737 177777 037052                CLR   TRNSWT     ;CLEAR DPB ADDRESS
1067 043542 000470                MOV   #-1,DTUW   ;CLEAR THE TRANSFER DRIVE #
                BR    STO9                ;DON'T BOTHER OTHER DRIVES

```

```

1068 043544 116405 000016      ST02:  MOV#  RMA5(R4),R5      ;READ ATTENTION REG
1069 043550 136105 037054      BIT#  ATABIT(R1),R5      ;IS ATTENTION FOR THIS DRIVE UP ?
1070 043554 001007      BNE   ST03              ;YES
1071 043556 105761 036760      TST#  DPINT(R1)         ;TRYING TO INTIALIZE THE DRIVE ?
1072 043562 001021      BNE   ST06              ;BR IF YES - DRIVE NOT ONLINE
1073 043564 105761 036770      TST#  DPRQS(R1)         ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1074 043570 001035      BNE   ST07              ;BR IF YES - NO RESPONSE TO REQUEST
1075 043572 000454      BR    ST09              ;OTHER WISE EXIT
1076 043574 105761 036760      ST03:  TST#  DPINT(R1)         ;INITIALIZING THE DRIVE ?
1077 043600 001003      BNE   1$                ;BR IF INIT PENDING
1078 043602 105761 036770      TST#  DPRQS(R1)         ;PORT REQUEST PENDING ?
1079 043606 001446      BEQ   ST09              ;BR IF NOT
1080 043610 012763 177777 037032  1$:   MOV   #-1,TIMER(R3)      ;STOP THE TIMER
1081 043616 000442      BR    ST09              ;EXIT
1082 043620 004737 041326      ST05:  JSR   PC,C18        ;GO HANDLE THE PARITY ERROR
1083 043624 000437      BR    ST09
1084 043626 105061 036760      ST06:  CLR#  DPINT(R1)         ;CLEAR THE INITIALIZE INDICATOR
1085 043632 105061 036740      CLR#  DRVSTA(R1)        ;SET UNIT OFFLINE
1086 043636 012763 177777 037032  MOV   #-1,TIMER(R3)      ;STOP THE TIMER
1087 043644 004737 045144      JSR   PC,GETREQ         ;GET THE DPB ADDRESS
1088 043650 005702      TST   R2                ;REQUEST IN QUEUE ?
1089 043652 001424      BEQ   ST09              ;BR IF NOT
1090 043654 052762 140000 000016  BIS   #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
1091 043662 000414      BR    ST08
1092 043664 012763 177777 037032  ST07:  MOV   #-1,TIMER(R3)      ;STOP THE TIMER
1093 043672 105061 036770      CLR#  DPRQS(R1)         ;CLEAR PORT REQUEST INDICATOR
1094 043676 004737 045144      JSR   PC,GETREQ         ;GET DPB ADDRESS
1095 043702 005702      TST   R2                ;QUEUE ENTRY FOR DRIVE ?
1096 043704 001407      BEQ   ST09              ;BR IF NONE
1097 043706 012762 100004 000016  MOV   #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
1098 043714 004737 045050      ST08:  JSR   PC,EMPTYQ     ;CLEAR THE QUEUE FOR THE DRIVE
1099 043720 004737 044310      JSR   PC,SVR#70        ;SAVE THE REGISTERS
1100 043724 012604      ST09:  MOV   (SP)+,R4       ;RESTORE R4
1101 043726 012603      MOV   (SP)+,R3         ;RESTORE R3
1102 043730 012602      MOV   (SP)+,R2         ;RESTORE R2
1103 043732 012601      MOV   (SP)+,R1         ;RESTORE R1
1104 043734 000207      RTS   PC                ;RETURN
1105
1106      ;ROUTINE TO READ A RH/RM REGISTER
1107      ;CALL
1108      ;
1109      JSR   RO,RD.RM       ;GO READ A REGISTER
1110      INDEX      ;REG. INDEX FROM BASE
1111      ERRADR     ;ERROR ADDRESS--PROCESS ERROR STARTING
1112      ;AT THIS ADDRESS
1113      ;CONTENTS OF REG. IS ON THE STACK
1114      RETURN
1115 043736 013737 037064 044104  RD.RM:  MOV   MCPEM#RD.RM2    ;MAX. RETRYS ALLOWED
1116 043744 011646      MOV   (SP),-(SP)       ;SAVE RO FOR RETURN
1117 043746 013737 037066 043762  MOV   RMADR,RD.ADR     ;FORM THE DESIRED ADDRESS
1118 043754 062037 043762      ADD   (RO)+,RD.ADR     ;USING THE BASE AND THE INDEX
1119 043760 013727      RD.RM1: MOV   @((PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RM DRIVE
1120 043762 000000      RD.ADR: .WORD 0        ;ADDRESS IS FORMED HERE
1121 043764 000000      RD.WRD: .WORD 0        ;REG. CONTENTS PUT HERE
1122 043766 013766 043764 000002  MOV   RD.WRD,2(SP)     ;RETURN IT TO THE USER
1123 043774 013746 037066      MOV   RMADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
1124 044000 062716 000010      ADD   #RMCS2,(SP)      ;FORM THE ADDRESS OF RMCS2

```



```

1125 044004 032736 010000          BIT    #BIT12,@(SP)+    ;CHECK THE 'MED' BIT
1126 044010 001037          BNE    RD.RM3          ;BR IF DRIVE NON-EXISTENT
1127 044012 017746 173050          MOV    @RMADR,-(SP)    ;READ RMCS1
1128 044016 032716 020000          BIT    #BIT13,(SP)    ;DID MCPE SET?
1129 044022 001002          BNE    1$             ;BR IF YES
1130 044024 022620          CMP    (SP)+,(RO)+    ;ADJUST FOR RETURN
1131 044026 000432          BR     RD.RM4         ;EXIT
1132 044030          1$:
      044030 004037 045234          JSR    RO,ES.SAV      ;SAVE THE ADDRESS IN '$ESCAPE'
      044034 104003          EMT    3              ;REPORT 'MCPE' ERROR
1133 044036 005737 037052          TST    DTUW          ;DATA TRANSFER UNDERWAY?
1134 044042 100405          BMI    2$             ;NO
1135 044044 032716 040000          BIT    #BIT14,(SP)    ;'TRE' = 1 ?
1136 044050 001402          BEQ    2$             ;NO
1137 044052 005726          TST    (SP)+         ;YES--CLEAN OFF THE STACK AND
1138 044054 000415          BR     RD.RM3         ;TAKE THE FATAL ERROR EXIT
1139 044056 052716 040000          2$: BIS    #BIT14,(SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1140 044062 000316          SWAB   (SP)           ;POSITION BEFORE WRITING
1141 044064 013737 037066 044100      MOV    RMADR,3$       ;FORM ADDRESS OF HIGH BYTE
1142 044072 005237 044100          INC    3$             ;
1143 044076 112637          MOVB   (SP)+,@(PC)+   ;WRITE THE HIGH BYTE OF RMCS1
1144 044100 000000          3$: .WORD 0           ;ADDRESS STORAGE
1145 044102 005327          DEC    (PC)+         ;EXCEEDED MAX. RETRYS
1146 044104 000003          RD.RM2: .WORD 3      ;
1147 044106 002324          BGE    RD.RM1         ;BR IF NO
1148 044110 011000          RD.RM3: MOV    (RO),RO ;FATAL ERROR EXIT
1149 044112 012616          MOV    (SP)+,(SP)    ;
1150 044114 000200          RD.RM4: RTS    RO    ;
1151
1152          ;ROUTINE TO WRITE A REGISTER
1153          ;
1154          ;CALL
1155          ;
1156          ;      MOV    DATA,-(SP)    ;DATA TO BE LOADED ON THE STACK
1157          ;      JSR    RO,WRT.RM    ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1158          ;      INDEX ERRADR      ;INDEX OF THE REGISTER TO BE LOADED
1159          ;      RETURN           ;ADDRESS TO RETURN TO ON AN ERROR
1160          ;      ;ERROR FREE RETURN
1161 044116 013737 037064 044274 WRT.RM: MOV    MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
1162 044124 016637 000002 044204      MOV    2(SP),WRT.WD  ;SAVE THE WORD TO WRITE
1163 044132 012616          MOV    (SP)+,(SP)    ;ADJUST THE STACK
1164 044134 012037 044206          MOV    (RO)+,WRT.AD  ;GET INDEX OF REGISTER TO BE WRITTEN
1165 044140 001015          BNE    1$             ;BR IF NOT RMCS1
1166 044142 122737 000150 044204      CMPB   #150,WRT.WD   ;IS THE COMMAND FOR DATA TRANSFERS?
1167 044150 002411          BLT    1$             ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1168 044152 004037 043736          JSR    RO,RD.RM      ;NO---COMBINE A16&A17, & PSEL WITH
1169 044156 000000          RMCS1 ;THE COMMAND BEFORE SENDING IT TO
1170 044160 044300          WRT.R3 ;THE RH/RM
1171 044162 000316          SWAB   (SP)           ;
1172 044164 042716 177770          BIC    #^C7,(SP)     ;
1173 044170 112637 044205          MOVB   (SP)+,WRT.WD+1 ;
1174 044174 063737 037066 044206      1$: ADD    RMADR,WRT.AD  ;FORM THE ADDRESS OF THE DISK REG.
1175 044202 012737          WRT.R1: MOV    (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
1176 044204 000000          WRT.WD: .WORD 0      ;WORD TO WRITE GOES HERE
1177 044206 000000          WRT.AD: .WORD 0      ;ADDRESS IS FORMED HERE
1178 044210 013746 037066          MOV    RMADR,-(SP)   ;PUT THE ADDRESS ON THE STACK
1179 044214 062716 000010          ADD    #RMCS2,(SP)   ;FORM THE ADDRESS OF RMCS2

```

```

1180 044220 032736 010000 BIT #BIT12,(SP)+ ;CHECK THE 'MED' BIT
1181 044224 001025 BNE WRT.R3 ;BR IF DRIVE NON-EXISTENT
1182 044226 004037 043736 JSR RO,RO.RM ;CHECK FOR PARITY ERROR ON WRITE
1183 044232 000014 RMER1
1184 044234 044300 WRT.R3
1185 044236 032726 000010 BIT #BIT03,(SP)+
1186 044242 001420 BEQ WRT.R4 ;BR IF 'PAR=0'
1187 044244 016037 177776 044256 MOV -2(RO),1$ ;PICKUP THE INDEX
1188 044252 004037 043736 JSR RO,RO.RM ;READ THE REG.
1189 044256 000000 1$: .WORD 0 ;REG. INDEX
1190 044260 044300 WRT.R3 ;RETURN TO THIS ADDRESS ON ERROR
1191 044262 004037 045234 JSR RO,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
044266 104004 EMT 4 ;REPORT THE PARITY ON WRITE ERROR
1192 044270 005726 TST (SP)+ ;CLEAR OFF THE STACK
1193 044272 005327 DEC (PC)+ ;DECREMENT THE ERROR COUNT
1194 044274 000003 WRT.R2: .WORD 3 ;RETRY COUNTER
1195 044276 002341 BGE WRT.R1 ;TRY AGAIN IF NOT FINISHED
1196 044300 011000 WPT.R3: MOV (RO),RO ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1197 044302 000401 BR WRT.R5 ;EXIT
1198 044304 005720 WRT.R4: TST (RO)+ ;ADJUST FOR ERROR FREE EXIT
1199 044306 000200 WRT.R5: RTS RO

1200 ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1201 ;CALL
1202 ;
1203 ;
1204 ; MOV #DPBNUM,R2 ;DPB POINTER TO R2
1205 ; JSR PC,SVRH70 ;SAVE THE DRIVES REG'S
1206 ;
1207 SVRH70: SAVREG ;SAVE R0 - R5
1208 044312 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE ?
1209 044314 001442 BEQ 6$ ;BR IF NONE
1210 044316 013704 037066 MOV RMADR,R4
1211 044322 111264 000010 MOV#B (R2),RMCS2(R4) ;SELECT DRIVE
1212 044326 016203 000014 MOV 14(R2),R3 ;GET THE ERROR TABLE POINTER
1213 044332 001433 BEQ 6$ ;EXIT IF NO ADDRESS
1214 044334 005037 044370 CLR 3$ ;COUNTER & POINTER
1215 044340 023727 044370 000022 1$: CMP 3$,#RMDB ;REACHED THE BUFFER REGISTER ?
1216 044346 001006 BNE 2$ ;BR IF NOT
1217 044350 032764 000200 000010 BIT #BIT07,RMCS2(R4) ;'OR' SET ?
1218 044356 001002 BNE 2$ ;BR IF SET
1219 044360 005023 CLR (R3)+ ;STORE RMDB AS ZEROES
1220 044362 000405 BR 4$ ;CONTINUE
1221 044364 004037 043736 2$: JSR RO,RO.RM ;READ THE SELECTED REGISTER
1222 044370 000000 3$: .WORD 0 ;REGISTER INDEX
1223 044372 044416 5$ ;ERROR RETURN ADDRESS
1224 044374 012623 MOV (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
1225 044376 023727 044370 000046 4$: CMP 3$,#RMEC2 ;REACHED THE END ?
1226 044404 001406 BEQ 6$ ;BR IF YES
1227 044406 062737 000002 044370 ADD #2,3$ ;INCREMENT THE REGISTER INDEX
1228 044414 000751 BR 1$ ;CONTINUE READING THE REGISTERS
1229 044416 004737 041226 5$: JSR PC,C17 ;PROCESS THE UNCORRECTABLE PARITY ERROR
1230 044422 104413 6$: RESREG ;RESTORE R0 - R5
1231 044424 000207 RTS PC ;RETURN
1232 ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE'
1233 ;CALL
1234 ;
1235 ; MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
  
```

```

1236      ;      JSR      PC,SET.IE      ;SET 'IE'
1237      ;      RETURN
1238
1239 044426 010446      SET.IE: MOV      R4,-(SP)      ;SAVE R4
1240 044430 013704 037066  MOV      RMADR,R4      ;PICKUP ADDRESS OF RMCS1
1241 044434 010164 000010  MOV      R1,RMCS2(R4)  ;SELECT DRIVE
1242 044440 011446      MOV      (R4),-(SP)    ;READ RMCS1
1243 044442 052716 040000  BIS      #BIT14,(SP)   ;SET THE 'TRE' BIT OF THE WORD READ
1244 044446 000316      SWAB      (SP)        ;ADJUST FOR DATO
1245 044450 112714 000100  MOVB     #BIT06,(R4)   ;SET 'IE'
1246 044454 032764 010000 000010  BIT      #BIT12,RMCS2(R4) ;IS 'NED'=1?
1247 044462 001002      BNE      1$           ;YES--CLEAR 'TRE'
1248 044464 005726      TST      (SP)+        ;CLEAN OFF THE STACK
1249 044466 000402      BR       2$
1250 044470 112664 000001  1$:  MOVB     (SP)+,1(R4) ;CLEAR 'TRE'
1251 044474 012604      2$:  MOV      (SP)+,R4   ;RESTORE R4
1252 044476 000207      RTS      PC          ;RETURN TO CALLER
1253
1254      ;QUEUE COUNT
1255 044500      000      QCNT: .BYTE 0      ;DRIVE 0
1256 044501      000      .BYTE 0      ;DRIVE 1
1257 044502      000      .BYTE 0      ;DRIVE 2
1258 044503      000      .BYTE 0      ;DRIVE 3
1259 044504      000      .BYTE 0      ;DRIVE 4
1260 044505      000      .BYTE 0      ;DRIVE 5
1261 044506      000      .BYTE 0      ;DRIVE 6
1262 044507      000      .BYTE 0      ;DRIVE 7
1263
1264      ;QUEUE INPUT POINTERS
1265
1266 044510 044572      QINPT: .WORD  QDRV0      ;DRIVE 0
1267 044512 044612      .WORD  QDRV1      ;DRIVE 1
1268 044514 044632      .WORD  QDRV2      ;DRIVE 2
1269 044516 044652      .WORD  QDRV3      ;DRIVE 3
1270 044520 044672      .WORD  QDRV4      ;DRIVE 4
1271 044522 044712      .WORD  QDRV5      ;DRIVE 5
1272 044524 044732      .WORD  QDRV6      ;DRIVE 6
1273 044526 044752      .WORD  QDRV7      ;DRIVE 7
1274
1275      ;QUEUE OUTPUT POINTERS
1276
1277 044530 044572      QOUTPT: .WORD  QDRV0      ;DRIVE 0
1278 044532 044612      .WORD  QDRV1      ;DRIVE 1
1279 044534 044632      .WORD  QDRV2      ;DRIVE 2
1280 044536 044652      .WORD  QDRV3      ;DRIVE 3
1281 044540 044672      .WORD  QDRV4      ;DRIVE 4
1282 044542 044712      .WORD  QDRV5      ;DRIVE 5
1283 044544 044732      .WORD  QDRV6      ;DRIVE 6
1284 044546 044752      .WORD  QDRV7      ;DRIVE 7
1285
1286 044550 044572      QSTART: .WORD  QDRV0      ;DRIVE 0 START ADDRESS
1287 044552 044612      QSTOP:  .WORD  QDRV1      ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1288 044554 044632      .WORD  QDRV2      ;STOP DRIVE 1--START DRIVE 2
1289 044556 044652      .WORD  QDRV3      ;STOP DRIVE 2--START DRIVE 3
1290 044560 044672      .WORD  QDRV4      ;STOP DRIVE 3--START DRIVE 4
1291 044562 044712      .WORD  QDRV5      ;STOP DRIVE 4--START DRIVE 5
1292 044564 044732      .WORD  QDRV6      ;STOP DRIVE 5--START DRIVE 6
  
```

```

1293 044566 044752          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
1294 044570 044772          .WORD  QTERM         ;STOP DRIVE 7
1295
1296          ;DRIVE REQUEST QUEUES
1297
1298 044572          QDRV0:  .BLKW  10
1299 044612          QDRV1:  .BLKW  10
1300 044632          QDRV2:  .BLKW  10
1301 044652          QDRV3:  .BLKW  10
1302 044672          QDRV4:  .BLKW  10
1303 044712          QDRV5:  .BLKW  10
1304 044732          QDRV6:  .BLKW  10
1305 044752          QDRV7:  .BLKW  10
1306          044772          QTERM=.
1307
1308          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1309
1310          ;CALL
1311          ;      JSR      PC,CLRQUE
1312
1313 044772 104412          CLRQUE: SAVREG          ;SAVE R0 - R5
1314 044774 012702 044500      MOV      #QCN*,R2          ;ZERO THE QUEUE COUNTS
1315 045000 005022          CLR      (R2)+          ;DRIVES 0 & 1
1316 045002 005022          CLR      (R2)+          ;DRIVES 2 & 3
1317 045004 005022          CLR      (R2)+          ;DRIVES 4 & 5
1318 045006 005022          CLR      (R2)+          ;DRIVES 6 & 7
1319 045010 012703 000010      MOV      #8.,R3          ;MOVE THE STARTING
1320 045014 012701 044550      MOV      #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
1321 045020 012122          1$:  MOV      (R1)+,(R2)+      ;THE QUEUE INPUT POINTER
1322 045022 005303          DEC      R3
1323 045024 001375          BNE     1$
1324 045026 012703 000010      MOV      #8.,R3          ;MOVE THE STARTING ADDRESS
1325 045032 012701 044550      MOV      #QSTART,R1      ;OF THE QUEUE INTO THE
1326 045036 012122          2$:  MOV      (R1)+,(R2)+      ;QUEUE OUTPUT POINTER
1327 045040 005303          DEC      R3
1328 045042 001375          BNE     2$
1329 045044 104413          RESREG          ;RESTORE R0 - R5
1330 045046 000207          RTS      PC
1331
1332          ;EMPTY THE QUEUE SPECIFIED BY R1
1333
1334          ;CALL
1335          ;      MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
1336          ;      JSR      PC,EMPTYQ
1337
1338 045050 105061 044500      EMPTYQ: CLRB      QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
1339 045054 006301          ASL      R1
1340 045056 016161 044510 044530      MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
1341 045064 006201          ASR      R1
1342 045066 000207          RTS      PC
1343
1344          ;ROUTINE TO PUT A REQUEST IN QUEUE
1345
1346          ;CALL
1347          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER
1348          ;      MOV      #DPB,R2          ;ADDRESS OF PARAMETER BLOCK
1349          ;      JSR      RO,DRVQJE          ;GO PUT REQUEST IN QUEUE

```

```

1350          :          RETURN1          :RETURN HERE IF QUEUE IS FULL
1351          :          RETURN2          :RETURN HERE IF REQUEST IS IN QUEUE
1352
1353 045070 122761 000010 044500 DRVQUE: CMPB    #10,QCNT(R1)  ;IS QUEUE FULL?
1354 045076 001421          BEQ      2$          ;BR IF YES-TAKE RETURN1
1355 045100 105261 044500          INCB    QCNT(R1)    ;INCREMENT QUEUE COUNT
1356 045104 006301          ASL     R1
1357 045106 010271 044510          MOV     R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1358 045112 062761 000002 044510          ADD     #2,QINPT(R1) ;UPDATE THE QUEUE PC'NTER
1359 045120 026161 044510 044552          CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1360 045126 001003          BNE    1$          ;BR IF NO
1361 045130 016161 044550 044510          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1362 045136 006201          1$: ASR     R1
1363 045140 005720          TST    (R0)+        ;TAKE RETURN 2
1364 045142 000200          2$: RTS     R0          ;RETURN TO USER
1365
1366          ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1367          :
1368          :CALL
1369          :          MOV     #DRVNUM,R1    ;DRIVE NUMBER TO R1
1370          :          JSR     PC,GETREQ    ;GO GET THE REQUEST
1371          :          RETURN    ;R2='DPB' ADDRESS OF THE REQUEST
1372          :          ;R2=0 IF NO REQUEST IN QUEUE
1373
1374 045144 005002          GETREQ: CLR     R2
1375 045146 105761 044500          TSTB   QCNT(R1)    ;IS THERE ANY RFQUEST IN QUEUE?
1376 045152 001404          BEQ    2$          ;NO
1377 045154 006301          1$: ASL     R1
1378 045156 017102 044530          MOV     @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
1379 045162 006201          ASR    R1
1380 045164 000207          2$: RTS     PC          ;RETURN TO USER
1381
1382          ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1383          :
1384          :CALL
1385          :          MOV     #DRVNUM,R1    ;DRIVE NUMBER TO R1
1386          :          JSR     PC,POPQUE    ;CALL TO REMOVE REQUEST
1387          :          RETURN    ;R2=ADDRESS OF DPB REMOVED
1388
1389 045166 00361 044500          POPQUE: DECB   QCNT(R1) ;DECREMENT QUEUE COUNT
1390 045172 001421          ASL    R1
1391 045174 001421 044530          MOV     @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
1392 045200 001421 044530          CLR    @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
1393 045204 062761 000002 044530          ADD     #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
1394 045212 026161 044530 044552          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
1395 045220 001003          BNE    1$          ;NO--BR TO EXIT
1396 045222 016161 044550 044530          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1397 045230 006201          1$: ASR    R1
1398 045232 000207          RTS    PC          ;RETURN TO USER
1399
1401          ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
1402          ;REPORTS AN ERROR DIRECTLY.
1403          :
1404          :CALL
1405          :          JSR     R0,ES.SAV    ;:THE ERROR CALL
1406          :          ERROR   N          ;:THE RETURN IS PAST THE ERROR CALL
1407          :          RETURN

```

1408
 1409 045234 012037 045250
 1410 045240 013746 001222
 1411 045244 005037 001222
 1412 045250 000000
 1413 045252 012637 001222
 1414 045256 000200
 1416

```

ES.SAV: MOV (R0)+,1$ ;GET THE ERROR CALL
          MOV $ESCAPE,-(SP) ;SAVE THE ADDRESS IN '$ESCAPE'
          CLR $ESCAPE ;CLEAR THE ESCAPE RETURN
1$: .WORD 0 ;THE ERROR CALL IS MOVED HERE
          MOV (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
          RTS R0 ;RETURN
  
```

```

1          .SBTTL  GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
2
3          ;THIS ROUTINE IS USED TO ENSURE THE BUS ADDRESS
4          ;OF THE RH/RM IS SETUP TO READ THE PROPER VALUE.
5          ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
6          ;REQUIRED.
7          ;NOTE: THIS ROUTINE DESTROYS R0-R4
8          ;CALL
9
10         :
11         :       JSR      PC,GETADR
12         :       RETURN
13
14 045260 005737 001322  GETADR: TST      BUSADR      ;INPUT FROM TTY REQUESTED?
15 045264 001427                BEQ      5$          ;NO--BRANCH
16 045266 005037 001322  CLR      BUSADR      ;YES--CLEAR THE REQUEST FLAG
17 045272 012700 001470  1$:  MOV     #RH.ADR,R0 ;FIRST ADDRESS
18 045276 012703 045436  MOV     #MRMCS1,R3  ;'RMCS1='
19 045302 011004                MOV     (R0),R4     ;PRESENT RMCS1 ADDRESS
20 045304 004037 034070  JSR     R0,GETNUM   ;GET NEW RMCS1
21 045310 000402                BR      2$          ;COMMA
22 045312 000767                BR      1$          ;PERIOD
23 045314 000412                BR      4$          ;DOUBLE PERIOD
24 045316 010420 045446  2$:  MOV     R4,(R0)+   ;SAVE NEW RMCS1
25 045320 012703                MOV     #MRMVEC,R3 ;'RMVEC='
26 045324 011004                MOV     (R0),R4     ;PRESENT RH/RM VECTOR ADDRESS
27 045326 004037 034070  JSR     R0,GETNUM   ;GET NEW RMVEC
28 045332 000402                BR      3$          ;COMMA
29 045334 000756                BR      1$          ;PERIOD
30 045336 000401                BR      4$          ;DOUBLE PERIOD
31 045340 010420 000004  3$:  MOV     R4,(R0)+   ;SAVE NEW RMVEC
32 045342 010410 000004  4$:  MOV     R4,(R0)    ;SAVE INPUT
33 045344 013701 000004  5$:  MOV     ERRVEC,R1   ;SAVE THE ERROR VECTOR
34 045350 012737 045404 000004  MOV     #6$,ERRVEC  ;SETUP FOR TRAP
35 045356 005777 134106                TST     @RH.ADR     ;CHECK FOR RH/RM
36 045362 010137 000004  MOV     R1,ERRVEC   ;RESTORE ERROR VECTOR
37 045366 012700 001470  MOV     #RH.ADR,R0  ;FIRST ADDRESS OF NEW PARAMETERS
38 045372 012701 037066  MOV     #RMADR,R1   ;FIRST ADDRESS OF WHERE TO PUT THEM
39 045376 012021                MOV     (R0)+,(R1)+ ;BUS ADDRESS
40 045400 012021                MOV     (R0)+,(R1)+ ;VECTOR ADDRESS
41 045402 000207                RTS     PC          ;RETURN
42 045404 010137 000004  6$:  MOV     R1,ERRVEC   ;RESTORE ERROR VECTOR
43 045410 022626                CMP     (SP)+,(SP)+ ;CLEAN OFF THE STACK
44 045412 104010                EMT     10
45 045414 005737 000042  TST     @#42        ;IS THERE A MONITOR?
46 045420 001724                BEQ     1$          ;NO--GO ASK FOR ADDRESS
47 045422 005037 001326  CLR     DRVSEL      ;YES--NO DRIVES SELECTED
48 045426 005037 021172  CLR     $EOPCT      ;NO PASSES
49 045432 000137 021020  JMP     $EOP        ;GO TO END OF PROGRAM
50 045436      200      122      115  MRMCS1: .ASCIZ <CRLF>/RMCS1=/
51 045446      200      122      115  MRMVEC: .ASCIZ <CRLF>/RMVEC-/
52

```



```

58 045526      000      .BYTE  0      ;(10) SECTOR ADDRESS OR
59              ;FIRST REG. INDEX
60 045527      000      .BYTE  0      ;(11) TRACK ADDRESS OR
61              ;LAST REG. INDEX
62 045530      000000    .WORD  0      ;(12) CYLINDER ADDRESS
63 045532      045556    .WORD  RM.REG ;(14) ERROR TABLE POINTER
64              ;POINTS TO THE FIRST OF TWENTY
65              ;LOCATIONS OF WHERE THE DRIVER
66              ;IS TO STORE THE RH/RM
67              ;REGISTERS ON AN ERROR. IF LEFT
68              ;ZERO REGISTERS ARE NOT SAVED.
69 045534      000000    .WORD  0      ;(16) STATUS/ERROR INDICATOR
70              ;BIT15=1=>ERROR OCCURRED
71              ;BIT07=1=>DONE
72              ;BIT14-BIT09 AND BIT06-BIT03
73              ;INDICATE TYPE OF ERROR
74
75 045536      000      DTADPB: .BYTE  0      ;(0) DRIVE NUMBER
76 045537      000      .BYTE  0      ;(1) OFFSET VALUE OR FMT16, ECT, AND HCI
77 045540      000      .BYTE  0      ;(2) COMMAND
78 045541      000      .BYTE  0      ;(3) PSEL AND A17 AND A16
79 045542      000000    .WORD  0      ;(4) WORD COUNT (MUST BE NEG.)
80 045544      052776    .WORD  BUFFER ;(6) BUFFER ADDRESS OR
81              ;REGISTER TABLE POINTER
82 045546      000      .BYTE  0      ;(10) SECTOR ADDRESS OR
83              ;FIRST REG. INDEX
84 045547      000      .BYTE  0      ;(11) TRACK ADDRESS OR
85              ;LAST REG. INDEX
86 045550      000000    .WORD  0      ;(12) CYLINDER ADDRESS
87 045552      045556    .WORD  RM.REG ;(14) ERROR TABLE POINTER
88              ;POINTS TO THE FIRST OF TWENTY
89              ;LOCATIONS OF WHERE THE DRIVER
90              ;IS TO STORE THE RH/RM
91              ;REGISTERS ON AN ERROR. IF LEFT
92              ;ZERO REGISTERS ARE NOT SAVED.
93 045554      000000    .WORD  0      ;(16) STATUS/ERROR INDICATOR
94              ;BIT15=1=>ERROR OCCURRED
95              ;BIT07=1=>DONE
96              ;BIT14-BIT09 AND BIT06-BIT03
97              ;INDICATE TYPE OF ERROR
98
99              ;SAVE RH/RM REGISTERS HERE ON ERROR
100
101 045556      000000    RM.REG: .WORD  0      ;RMCS1 (776700) CONTROL & STATUS #1
102 045560      000000    .WORD  0      ;RMWC (776702) WORD COUNT
103 045562      000000    .WORD  0      ;RMBA (776704) BUS ADDRESS
104 045564      000000    .WORD  0      ;RMDA (776706) DESIRED SECTOR/TRACK
105 045566      000000    .WORD  0      ;RMCS2 (776710) CONTROL & STATUS #2
106 045570      000000    .WORD  0      ;RMD5 (776712) DISK STATUS
107 045572      000000    .WORD  0      ;RMER1 (776714) ERROR REG. #1
108 045574      000000    .WORD  0      ;RMAS (776716) ATTENTION SUMMARY
109 045576      000000    .WORD  0      ;RMLA (776720) LOOK AHEAD
110 045600      000000    .WORD  0      ;RMDB (776722) DATA BUFFER
111 045602      000000    .WORD  0      ;RMMR1 (776724) MAINTAINABILITY
112 045604      000000    .WORD  0      ;RMDT (776726) DRIVE TYPE
113 045606      000000    .WORD  0      ;RMSN (776730) SERIAL NUMBER
114 045610      000000    .WORD  0      ;RMOF (776732) OFFSET

```

115 045612 000000
116 045614 000000
117 045616 000000
118 045620 000000
119 045622 000000
120 045624 000000
121
122

.EVEN

.WORD 0
.WORD 0
.WORD 0
.WORD 0
.WORD 0
.WORD 0

;RMDC (776734) DESIRED CYLINDER
;RMHR (776736) CURRENT CYLINDER
;RMMR2 (776740) ERROR REG #2
;RMR2 (776742) ERROR REG #3
;RMEC1 (776744) ECC POSI'ION
;RMEC2 (776746) ECC PATTERN

```

1
2
3 045626 122 000
4 045630 106 103 000
5 045633 114 103 000
6 045636 111 103 000
7 045641 106 124 000
8 045644 114 124 000
9 045647 111 124 000
10 045652 106 124 047
11 045656 114 124 047
12 045662 111 124 047
13 045666 106 123 000
14 045671 114 123 000
15 045674 120 101 124
16 045700 075 000
17 045702 200 103 117
18
19 045725 040 057 040
20 045731 200 125 116
21 045747 040 117 106
22 045760 040 117 116
23 045770 040 116 117
24 046005 040 125 116
25 046015 040 116 117
26 046036 040 111 123
27 046056 122 115 060
28 046063 122 115 060
29 046070 122 115 060
30 046075 200 104 122
31 046125 116 117 116
32 046132 054 040 000
33 046135 200 116 117
34 046222 200 012 124
35 046242 115 102 101
36 046253 200 124 105
37
38 046261 200 012 122
39 046312 200 012 117
40 046357 200 012 101
41 046424 200 012 115
42 046450 200 040 052
43 046464 200 040 052
44
45 046500 200 115 111
46 046506 200 115 101
47 046514 200 101 126
48 046522 060 040 125
49 046527 040 102 105
50 046556 040 101 102
51 046605 040 123 105
52 046625 040 123 105
53 046642 040 116 117
54 046655 040
55 046656 040
56 046657 040
57 046660 040 000

.SBTTL ASCIZ MESSAGES
MSG.R: .ASCIZ /R/
MSG.FC: .ASCIZ /FC/
MSG.LC: .ASCIZ /LC/
MSG.IC: .ASCIZ /IC/
MSG.FT: .ASCIZ /FT/
MSG.LT: .ASCIZ /LT/
MSG.IT: .ASCIZ /IT/
MES.FT: .ASCIZ /FT'/
MES.LT: .ASCIZ /LT'/
MES.IT: .ASCIZ /IT'/
MSG.FS: .ASCIZ /FS/
MSG.LS: .ASCIZ /LS/
MSG.PAT: .ASCIZ /PAT/
MSG.EQ: .ASCIZ /=/
MSG.CS: .ASCIZ <CRLF>/CONTROL SWITCHES=/

SLASH: .ASCIZ @ / @
UNSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
UNTOFF: .ASCIZ / OFFLINE/
UNTON: .ASCIZ / ONLINE/
NOTPRS: .ASCIZ / NOT PRESENT/
NOTSAF: .ASCIZ / UNSAFE/
NOTRM: .ASCIZ @ NOT AN RM05/3/2@
LODEV: .ASCIZ / IS LOAD DEVICE/
$RM02: .ASCIZ /RM02/
$RM03: .ASCIZ /RM03/
$RM05: .ASCIZ /RM05/
DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED /
NONE: .ASCIZ /NONE/
COMMA: .ASCIZ /, /
NOCLOK: .ASCIZ <CRLF>/NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED/
TSTDRV: .ASCIZ <CRLF><LF>/TESTING DRIVE/
SERIAL: .ASCIZ /MBA SN# /
MSGTST: .ASCIZ <CRLF>/TEST/

ROTATE: .ASCIZ <CRLF><LF>/ROTATIONAL SPEED TIMES/
ONECYL: .ASCIZ <CRLF><LF>/ONE CYLINDER SEEK TIMES/<CRLF>/ * FORWARD/
ACCESS: .ASCIZ <CRLF><LF>/ACCESS TIME MEASUREMENT/<CRLF>/ * FORWARD/
MXSEEK: .ASCIZ <CRLF><LF>/MAXIMUM SEEK TIMES/
FWD: .ASCIZ <CRLF>/ * FORWARD/
REV: .ASCIZ <CRLF>/ * REVERSE/

MSGMIN: .ASCIZ <CRLF>/MIN=/
MSGMAX: .ASCIZ <CRLF>/MAX=/
MSGAVG: .ASCIZ <CRLF>/AVG=/
MSGOUS: .ASCIZ /O US/
MBELOW: .ASCIZ / BELOW THE MINIMUM OF /
MAPOVE: .ASCIZ / ABOVE THE MAXIMUM OF /
MSG EA: .ASCIZ / SEARCHES TIMED/
MSGNUM: .ASCIZ / SEEKS TIMED/
MSGNON: .ASCIZ / NOT TIMED/
BLNKS4: .ASCIZ / /
BLNKS3: .ASCIZ / /
BLNKS2: .ASCIZ / /
BLNKS1: .ASCIZ / /

```

58	046662	101	114	114	MSG7XA: .ASCIZ @ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM05/3@
59	046737	101	114	114	MSG7XB: .ASCIZ /ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM02/
60	047012	101	114	114	MSG10X: .ASCIZ /ALLOWABLE ONE CYLINDER SEEK LIMIT/
61	047054	101	114	114	MSG11X: .ASCIZ /ALLOWABLE ACCESS TIME LIMIT/
62	047110	101	114	114	MSG12X: .ASCIZ /ALLOWABLE MAXIMUM SEEK TIME LIMIT/
63					



```
1  
2 .SBTTL ERROR HEADER (EM) MESSAGES  
3 047152 122 110 057 EM1: .ASCIZ @RH/RM INTERRUPT OCCURRED (RMAS = 0)@  
4 047216 125 116 105 EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/  
5 047254 115 101 123 EM3: .ASCIZ /MASSBUS PARITY ERROR(MCPE=1)/  
6 047311 115 101 123 EM4: .ASCIZ /MASSBUS PARITY ERROR(PAR=1)/  
7 047345 101 104 104 EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/  
8 047401 122 110 057 EM10: .ASCIZ @RH/RM FAILED TO RESPOND TO ADDRESSING@  
9 047447 104 122 111 EM11: .ASCIZ /DRIVE SELECTED IS NOT ONLINE/  
10 047504 111 115 120 EM12: .ASCIZ /IMPROPER HEADER DATA/  
11 047531 104 101 124 EM13: .ASCIZ /DATA COMPARE FAILURE/  
12 047556 104 111 123 EM17: .ASCIZ /DISK ERROR IN TIMING TEST/  
13 047610 103 114 117 EM20: .ASCIZ /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/  
14 047657 104 111 123 EM23: .ASCIZ /DISK ERROR DURING SEEK/  
15 047706 123 105 105 EM24: .ASCIZ /SEEK NOT COMPLETE WITHIN 120 MS/  
16 047746 122 110 057 EM41: .ASCIZ @RH/RM ERROR@  
17 047762 106 101 124 EM46: .ASCIZ /FATAL WRITE CHECK ERROR/  
18
```

```
1  
2 .SBTTL STATUS/ERROR INDICATOR MESSAGES  
3 050012 117 106 106 MSGB14: .ASCIZ /OFFLINE OR UNSAFE DRIVE REQUESTED/  
4 050054 125 116 114 MSGB13: .ASCIZ /UNLOADED DRIVE REQUESTED/  
5 050105 120 105 122 MSGB12: .ASCIZ /PERSISTENT UNSAFE/  
6 050127 120 101 122 MSGB11: .ASCIZ /PARITY ERROR OCCURRED/  
7 050155 106 101 124 MSGB10: .ASCIZ /FATAL PARITY ERROR/  
8 050200 123 117 106 MSGB09: .ASCIZ /SOFTWARE TIMEOUT ON THIS DRIVE/  
9 050237 123 117 106 MSGB08: .ASCIZ /SOFTWARE TIMEOUT ON ANOTHER DRIVE/  
10 050301 105 122 122 MSGB06: .ASCIZ "ERROR OCCURRED DURING I/O OPERATION"  
11 050345 105 122 122 MSGB05: .ASCIZ "ERROR OCCURRED DURING NON-I/O OPERATION"  
12 050415 125 116 123 MSGB04: .ASCIZ /UNSAFE OCCURRED/  
13 050435 101 125 124 MSGB03: .ASCIZ /AUTOMATIC RECALIBRATE SEQUENCE OCCURRED/  
14 050505 104 122 111 MSGB02: .ASCIZ /DRIVE HAS NOT RESPONDED TO PORT REQUEST/  
15 050555 104 122 111 MSGB01: .ASCIZ /DRIVE HAS BECOME NON-EXISTENT/  
16
```


DATA TABLE (DT)

.SBTTL DATA TABLE (DT)

1							
2							
3	052170	001132	001204		DT1:	.WORD	\$ERRPC,\$REG3
4	052174	001132	001200	001204	DT2:	.WORD	\$ERRPC,\$REG1,\$REG3,RMERRS,RMERRS+2,RMERRS+6,RMERRS+4
5	052212	001212	001132	043762	DT3:	.WORD	\$TMPO,\$ERRPC,RD.ADR,RD.WRD
6	052222	001212	001132	044206	DT4:	.WORD	\$TMPO,\$ERRPC,WRT.ADR,WRT.WD,RD.WRD
7	052234	001212	001132	001200	DT5:	.WORD	\$TMPO,\$ERRPC,\$REG1,\$REG5,RMERRS,RMERRS+2,RMERRS+6,RMERRS+4
8	052254	001470	001132		DT10:	.WORD	RM.ADR,\$ERRPC
9	052260	001202	001132		DT11:	.WORD	\$REG2,\$ERRPC
10	052264	001212	001132	001176	DT12:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
11	052302	001364	001370	001366	DT12A:	.WORD	CYL.DS,TRK.DS,SEC.DS,CYL.RD,TRK.RD,SEC.RD
12	052316	001212	001132	001176	DT13:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
13	052334	001140	001142	001206	DT13A:	.WORD	\$GDDAT,\$BDDAT,\$REG4,\$GDADR,\$BDADR
14	052346	001212	001132	001350	DT17:	.WORD	\$TMPO,\$ERRPC,CHKDRV,RM.REG,RM.REG+12,RM.REG+14,RM.REG+40,RM.REG+42
15	052366	001212	001132	001176	DT21:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS
16	052402	001200	001142	001206	DT21A:	.WORD	\$REG1,\$BDDAT,\$REG4,\$REG1
17	052412	001212	001132	001350	DT23:	.WORD	\$TMPO,\$ERRPC,CHKDRV,CYL.DS,RM.REG,RM.REG+10,RM.REG+12
18	052430	045572	045616	045620	DT23A:	.WORD	RM.REG+14,RM.REG+40,RM.REG+42,RM.REG+34,RM.REG+36
19	052442	001212	001132	001176	DT41:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV
20	052452	001212	001132	001176	DT42:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,RM.REG,RM.REG+10,RM.REG+12
21	052470	001212	001132	001176	DT43:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,RM.REG,RM.REG+10,RM.REG+12
22	052506	045572	045616	045620	DT43A:	.WORD	RM.REG+14,RM.REG+40,RM.REG+42
23	052514	001212	001132	001176	DT44:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
24	052532	045556	045566	045570	DT44A:	.WORD	RM.REG,RM.REG+10,RM.REG+12,RM.REG+36,RM.REG+34,RM.REG+06
25	052546	045572	045616	045620	DT44B:	.WORD	RM.REG+14,RM.REG+40,RM.REG+42
26	052554	001212	001132	001176	DT45:	.WORD	\$TMPO,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
27	052572	045556	045566	045570	DT45A:	.WORD	RM.REG,RM.REG+10,RM.REG+12,RM.REG+36,RM.REG+34,RM.REG+06
28	052606	045572	045616	045620	DT45B:	.WORD	RM.REG+14,RM.REG+40,RM.REG+42,RM.REG+2,RM.REG+4,RM.REG+22
29							

Line	Address	Value	Label	Format	Count	Notes
1			.SBTTL	DATA FORMAT (DF) TABLE		
2						
3	052622	000001	DF1:	.WORD	1	:NUMBER OF DATA HEADERS :NUMBER OF WORDS IN DATA TABLE :ALL 3 NUMBERS ARE OCTAL
4	052624	002		.BYTE	2	
5	052625	000		.BYTE	0	
6						
7	052626	000001	DF2:	.WORD	1	
8	052630	007		.BYTE	7	
9	052631	000		.BYTE	0	
10						
11	052632	000001	DF3:	.WORD	1	
12	052634	004		.BYTE	4	
13	052635	000		.BYTE	0	
14						
15	052636	000001	DF4:	.WORD	1	
16	052640	005		.BYTE	5	
17	052641	000		.BYTE	0	
18						
19	052642	000001	DF10:	.WORD	1	
20	052644	002		.BYTE	2	
21	052645	000		.BYTE	0	
22						
23	052646	000001	DF11:	.WORD	1	
24	052650	002		.BYTE	2	
25	052651	000		.BYTE	0	
26						
27	052652	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	052654	007		.BYTE	7	
29	052655	160		.BYTE	160	
30	052656	051146		.WORD	DH12A	
31	052660	006		.BYTE	6	
32	052661	000		.BYTE	0	
33						
34	052662	000002	DF13:	.WORD	2	
35	052664	007		.BYTE	7	
36	052665	160		.BYTE	160	
37	052666	051225		.WORD	DH13A	
38	052670	005		.BYTE	5	
39	052671	004		.BYTE	4	
40						
41	052672	000000	DF14:	.WORD	0	
42	052674	005		.BYTE	5	
43	052675	004		.BYTE	4	
44						
45	052676	000001	DF17:	.WORD	1	
46	052700	010		.BYTE	^D8	
47	052701	000		.BYTE	0	
48						
49	052702	000002	DF21:	.WORD	2	
50	052704	006		.BYTE	6	
51	052705	060		.BYTE	60	
52	052706	051446		.WORD	DH21A	
53	052710	004		.BYTE	4	
54	052711	014		.BYTE	14	
55						
56	052712	000000	DF22:	.WORD	0	
57	052714	004		.BYTE	4	

:WORD 3 IS DECIMAL

:WORD 3 IS DECIMAL

58	052715	014		.BYTE	14	
59						
60	052716	000002	DF23:	.WORD	2	
61	052720	007		.BYTE	7	
62	052721	010		.BYTE	10	
63	052722	051572		.WORD	DH23A	;WORD 4 IS DECIMAL
64	052724	005		.BYTE	5	
65	052725	000		.BYTE	0	
66						
67						
68	052726	000001	DF41:	.WORD	1	
69	052730	004		.BYTE	4	
70	052731	000		.BYTE	0	
71						
72	052732	000001	DF42:	.WORD	1	
73	052734	007		.BYTE	7	
74	052735	000		.BYTE	0	
75						
76	052736	000002	DF43:	.WORD	2	
77	052740	007		.BYTE	7	
78	052741	000		.BYTE	0	
79	052742	051762		.WORD	DH43A	
80	052744	003		.BYTE	3	
81	052745	000		.BYTE	0	
82						
83	052746	000003	DF44:	.WORD	3	
84	052750	007		.BYTE	7	
85	052751	160		.BYTE	160	
86	052752	052010		.WORD	DH44A	
87	052754	006		.BYTE	6	
88	052755	000		.BYTE	0	
89	052756	052064		.WORD	DH44B	
90	052760	003		.BYTE	3	
91	052761	000		.BYTE	0	
92						
93	052762	000003	DF45:	.WORD	3	
94	052764	007		.BYTE	7	
95	052765	160		.BYTE	160	
96	052766	052010		.WORD	DH44A	
97	052770	006		.BYTE	6	
98	052771	000		.BYTE	0	
99	052772	052112		.WORD	DH45A	
100	052774	006		.BYTE	6	
101	052775	000		.BYTE	0	
102						
103			.SBTTL	START OF READ/WRITE BUFFER		
104	052776		BUFFER:			
105						
106		000200	.END		200	

ABASE = 000000	AT3 = 000010	C17 = 041226	DH17 = 051273	DT12 = 052264
ACCESS = 046357	AT4 = 000020	C17B = 041250	DH2 = 050630	DT12A = 052302
ACDW1 = 000000	AT5 = 000040	C18 = 041326	DH21 = 051370	DT13 = 052316
ACDW2 = 000000	AT6 = 000100	CKSCTR = 032362	DH21A = 051446	DT13A = 052334
ACPUOP = 000000	AT7 = 000200	CKSWR = 104407	DH23 = 051505	DT17 = 052346
ACTDRV = 037004	AUNIT = 000000	CK.CHR = 036340	DH23A = 051572	DT2 = 052174
ACTSTR = 037005	AUSWR = 000000	CK.DEC = 036312	DH3 = 050715	DT21 = 052366
ADDW0 = 000000	AVECT1 = 000000	CK.DIG = 036414	DH4 = 050752	DT21A = 052402
ADDW1 = 000000	AVECT2 = 000000	CK.NUM = 036600	DH41 = 051637	DT23 = 052412
ADDW10 = 000000	A16 = 000400	CK.OCT = 036264	DH42 = 051675	DT23A = 052430
ADDW11 = 000000	A17 = 001000	CLKSTA = 001340	DH43A = 051762	DT3 = 052212
ADDW12 = 000000	BA1 = 000010	CLOSE = 036204	DH44A = 052010	DT4 = 052222
ADDW13 = 000000	BASFLG = 001464	CLR = 000040	DH44B = 052064	DT41 = 052442
ADDW14 = 000000	BITS = 001526	CLREJF = 032314	DH45A = 052112	DT42 = 052452
ADDW15 = 000000	BIT0 = 000001	CLRQUE = 044772	DISPLA = 001156	DT43 = 052470
ADDW2 = 000000	BIT00 = 000001	CLSWDS = 036104	DISPRE = 000174	DT43A = 052506
ADDW3 = 000000	BIT01 = 000002	CNTCLR = 026072	DLT = 100000	DT44 = 052514
ADDW4 = 000000	BIT02 = 000004	CNTRLC = 001320	DMD = 000001	DT44A = 052532
ADDW5 = 000000	BIT03 = 000010	COMMA = 046132	DORTI = 031272	DT44B = 052546
ADDW6 = 000000	BIT04 = 000020	CONT = 000100	DPB.A = 045456	DT45 = 052554
ADDW7 = 000000	BIT05 = 000040	COUNT = 031336	DPB.B = 045476	DT45A = 052572
ADDW8 = 000000	BIT06 = 000100	CR = 000015	DPB.C = 045516	DT45B = 052606
ADDW9 = 000000	BIT07 = 000200	CRLF = 000200	DPINT = 036760	DT5 = 052234
ADEVCT = 000000	BIT08 = 000400	CYL.DS = 001364	DPR = 000400	DULP = 037320
ADEVN = 000000	BIT09 = 001000	CYL.RD = 001356	DPRQS = 036770	DVA = 004000
AENV = 000000	BIT1 = 000002	C.SWR = 001314	DRIVES = 046075	EBL = 020000
AENVN = 000000	BIT10 = 002000	DATCMP = 033022	DRQ = 004000	ECH = 000100
AFATAL = 000000	BIT11 = 004000	DCK = 100000	DRVACT = 036730	ECI = 004000
AMADR1 = 000000	BIT12 = 010000	DDISP = 177570	DRVCAL = 027670	ECRC = 001000
AMADR2 = 000000	BIT13 = 020000	DECSEC = 032232	DRVCLR = 000111	EECC = 000020
AMADR3 = 000000	BIT14 = 040000	DECSK = 007630	DRVCL1 = 027710	EMPTYQ = 045050
AMADR4 = 000000	BIT15 = 100000	DELTA = 001446	DRVINT = 037316	EMTVEC = 000030
AMAMS1 = 000000	BIT2 = 000004	DFLT = 002514	DRVMSK = 001352	EM1 = 047152
AMAMS2 = 000000	BIT3 = 000010	DF1 = 052622	DRVOK = 006560	EM10 = 047401
AMAMS3 = 000000	BIT4 = 000020	DF10 = 052642	DRVQUE = 045070	EM11 = 047447
AMAMS4 = 000000	BIT5 = 000040	DF11 = 052646	DRVSEL = 001326	EM12 = 047504
AMSGAD = 000000	BIT6 = 000100	DF12 = 052652	DRVSTA = 036740	EM13 = 047531
AMSGLG = 000000	BIT7 = 000200	DF13 = 052662	DRV TYP = 036750	EM17 = 047556
AMSGTY = 000000	BIT8 = 000400	DF14 = 052672	DRY = 000200	EM2 = 047216
AMTYP1 = 000000	BIT9 = 001000	DF17 = 052676	DSWR = 177570	EM20 = 047610
AMTYP2 = 000000	BLNKS1 = 046660	DF2 = 052626	DTADPB = 045536	EM23 = 047657
AMTYP3 = 000000	BLNKS2 = 046657	DF21 = 052702	DTE = 010000	EM24 = 047706
AMTYP4 = 000000	BLNKS3 = 046656	DF22 = 052712	DTG = 000020	EM3 = 047254
AOE = 001000	BLNKS4 = 046655	DF23 = 052716	DT0 = 020000	EM4 = 047311
APASS = 000000	BPTVEC = 000014	DF3 = 052632	DTUW = 037052	EM41 = 047746
APRIOR = 000000	BUFFER = 052776	DF4 = 052636	DT00 = 000001	EM46 = 047762
APTCSU = 000040	BUSADR = 001322	DF41 = 052726	DT01 = 000002	EM5 = 047345
APTENV = 000001	BYPASS = 001346	DF42 = 052732	DT02 = 000004	ERINDX = 030526
APTSIZ = 000200	CALL.A = 027132	DF43 = 052736	DT03 = 000010	ERR = 040000
APTSPO = 000100	CALL.B = 027264	DF44 = 052746	DT04 = 000020	ERROR = 104000
ASWREG = 000000	CALL.C = 027466	DF45 = 052762	DT05 = 000040	ERRVEC = 000004
ATA = 100000	CHKDRV = 001350	DH1 = 050613	DT06 = 000100	ERR.CT = 001462
ATABIT = 037054	C11 = 040404	DH10 = 051021	DT07 = 000200	ESRC = 004000
ATESTN = 000000	C13 = 040512	DH11 = 051040	DT08 = 000400	ES.SAV = 045234
ATO = 000001	C14 = 040612	DH12 = 051057	DT1 = 052170	EXIT.A = 014120
AT1 = 000002	C15 = 041170	DH12A = 051146	DT10 = 052254	EXIT0 = 007202
AT2 = 000004	C16 = 041212	LH13A = 051225	DT11 = 052260	EXIT1 = 007410

EXIT10	012746	INCTRK	032152	MSGB10	050155	OPNFLG	001334	PRM14	003402
EXIT11	013212	IOTVEC=	000020	MSGB11	050127	OPNPAT	035572	PRM15	003412
EXIT12	014134	IR =	000100	MSGB12	050105	OPNPRM	035370	PRM16	003422
EXIT13	014636	ISR	041740	MSGB13	050054	OPNTST	035216	PRM17	003434
EXIT14	015376	IT	002340	MSGB14	050012	OPNWDS	035714	PRM2	003160
EXIT15	016136	ITEM41	002242	MSGMAX	046506	OPN.CT	035222	PRM20	003444
EXIT16	016560	LA	041564	MSGMIN	046500	OPN.N1	036126	PRM21	003500
EXIT17	017224	LACNT	037016	MSGNON	046642	OPN.N2	036132	PRM22	003510
EXIT2	007652	LC	002330	MSGNUM	046625	OPN.X1	036144	PRM3	003200
EXIT20	020004	LDCMD	027066	MSGSEA	046605	OPN.X2	036150	PRM4	003220
EXIT21	020622	LF =	000012	MSGTST	046253	OPN.1	035226	PRM5	003240
EXIT22	021016	LKS	001514	MSG.CS	045702	OPN.2	035250	PRM6	003260
EXIT3	010072	LKV	001510	MSG.EQ	045700	OPT	040146	PRM7	003300
EXIT4	010504	LODEV	046036	MSG.FC	045630	OR =	000200	PRO =	000000
EXIT5	010730	LODFLT	026434	MSG.FS	045666	PACK =	000123	PR1 =	000040
EXIT6	011220	LODPRM	026672	MSG.FT	045641	PAR =	000010	PR2 =	000100
EXIT7	011616	LOP.CK	030466	MSG.IC	045636	PAT	002346	PR3 =	000140
FC	002326	LPB	001524	MSG.IT	045647	PAT.PT	003520	PR4 =	000200
FER =	000020	LPS	001522	MSG.LC	045633	PAT0	003560	PR5 =	000240
FILBUF	032256	LPTAVL	001324	MSG.LS	045671	PAT1	003620	PR6 =	000300
FILRAN	033340	LP.AVL	026114	MSG.LT	045644	PAT10	004260	PR7 =	000340
FMT16 =	010000	LS	002344	MSG.PA	045674	PAT11	004320	PS =	177776
FS	002342	LSIT =	000002	MSG.R	045626	PAT12	004360	PSEL =	002000
FT	002334	LST =	002000	MSGOUS	046522	PAT13	004420	PSW =	177776
FWD	046450	LSTRK	001372	MSG10X	047012	PAT14	004460	PTRN15	003476
F1 =	000002	LT	002336	MSG11X	047054	PAT15	004520	PWRVEC=	000024
F2 =	000004	MABOVE	046556	MSG12X	047110	PAT2	003660	QCNT	044500
F3 =	000010	MBELOW	046527	MSG7XA	046662	PAT3	003720	QDRV0	044572
F4 =	000020	MCLK =	004000	MSG7XB	046737	PAT4	003760	QDRV1	044612
F5 =	000040	MCPE =	020000	MUR =	001000	PAT5	004020	QDRV2	044632
GETADR	045260	MCPEMX	037064	MWP =	000010	PAT6	004060	QDRV3	044652
GETNUM	034070	MDF =	000100	MXDLTA	037076	PAT7	004120	QDRV4	044672
GE'REG=	000141	MES.FT	045652	MXF =	001000	PAT8	004160	QDRV5	044712
GE'REQ	045144	MES.IT	045662	MXLACT	037074	PAT9	004220	QDRV6	044732
GE'SWR	033776	MES.LT	045656	MXSEEK	046424	PDA =	000400	QDRV7	044752
GO =	000001	MI =	000004	MXSTAL	001460	PGE =	002000	QINPT	044510
GTSWR =	104406	MNDLTA	037100	MXWWDW	037102	PGM =	001000	QOUTPT	044530
GTTST1	034364	MOC =	000400	NBA =	100000	PHA =	000200	QSTART	044550
GTTST2	034450	MOH =	020000	NC1	002356	PIP =	020000	QSTOP	044552
GTTST3	035012	MOL =	010000	NC2	002360	PIRQ =	177772	QTERM =	044772
GTTST4	035014	MPE =	000400	NED =	010000	PIRQVE=	000240	RANADR	033614
GTTST5	035124	MRD =	002000	NEM =	004000	PKB	001504	RANCK	033362
GTTST6	035164	MRMCS1	045436	NOCLOK	046135	PKC	001506	RANPAT	033560
GT.PRM	034226	MRMVEC	045446	NONE	046125	PKCS	001502	RDCHR =	104410
GT.PR1	034230	MS =	000040	NOOP =	000101	PKV	001476	RDLIN =	104411
GT.PR2	034360	MSC =	000002	NOTPRS	045770	PLFS =	002000	RDY =	000200
HCE =	000200	MSEN =	010000	NOTRM	046015	POPQUE	045166	RD.ADR	043762
HCI =	002000	MSER =	000200	NOTSAF	046005	PRM	002322	RD.RM	043736
HCRC =	000400	MSGAVG	046514	OBCK =	100000	PRMLMT	002432	RD.RM1	043760
HT =	000011	MSGB01	050555	OBEN =	040000	PRMSG	002464	RD.RM2	044104
IAE =	002000	MSGB02	050505	OCC =	100000	PRMPT	002362	RD.RM3	044110
IC	002332	MSGB03	050435	OFD =	000200	PRM0	003116	RD.RM4	044114
IE =	000100	MSGB04	050415	OFFSET=	000115	PRM1	003132	RD.WRD	043764
ILF =	000001	MSGB05	050345	OM =	000001	PRM10	003320	READ =	000171
ILR =	000002	MSGB06	050301	ONECYL	046312	PRM11	003336	READHD=	000173
INCCYL	032202	MSGB08	050237	OPE =	020000	PRM12	003356	READIN=	000121
INCSK	007566	MSGB09	050200	OPI =	020000	PRM13	003372	RECAL -	000107

RELEAS=	000113	SC5	042242	STRT2A	004616	TEST7	011362	T7B	001616
RESREG=	104413	SC6	042436	ST.CLK	026156	TICKMS	001342	T7B1	001636
RESTAR	006540	SC6A	042552	ST.LCL	026366	TICKUS	001344	ULDFLG	037006
RESVEC=	000010	SC7	042700	ST.PCL	026324	TIMER	037032	UNLOAD=	000103
REV	046464	SC8	042756	SVADR	001436	TIM.DN	001412	UNS	= 040000
REX	= 010000	SEARCH=	000131	SVRH70	044310	TIM.PT	001430	UNSTAT	045731
RHVEC	001472	SEC.DS	001366	SVSTAT	001354	TIM.UP	001374	UNTOFF	045747
RH.ADR	001470	SEC.RD	001342	SWR	001154	TKVEC	= 000060	UNTON	045760
RMADR	037066	SEEK	= 000105	SWREG	000176	TPB	001520	UPE	= 020000
RMAS	= 000016	SEEKFG	037030	SW0	= 000001	TPS	001516	US1	= 000001
RMBA	= 000004	SEKCNT	001444	SW00	= 000001	TPS50	014132	US2	= 000002
RMCS1	= 000000	SEKTMR	001442	SW01	= 000002	TPS60	014114	US4	= 000004
RMCS2	= 000010	SELDRV=	000145	SW02	= 000004	TPVEC	= 000064	VERIFY	030746
RMDA	= 000006	SERIAL	046242	SW03	= 000010	TP50	014124	VV	= 000100
RMDB	= 000022	SETBUF	032732	SW04	= 000020	TP60	014106	WC	= 000040
RMDC	= 000034	SETFOR=	000143	SW05	= 000040	TRAPVE=	000034	WCE	= 040000
RMDS	= 000012	SETVEC	005760	SW06	= 000100	TRCKWC	001450	WCEFLG	001432
RMDT	= 000026	SET.IE	044426	SW07	= 000200	TRE	= 040000	WCF	= 000040
RMEC1	= 000044	SKI	= 040000	SW08	= 000400	TRK.DS	001370	WD	= 000010
RMEC2	= 000046	SLASH	045725	SW09	= 001000	TRK.RD	001360	WLE	= 004000
RMERRS	036720	SPTYP	031470	SW1	= 000002	TRNSWT	037000	WRCKD	= 000151
RMER1	= 000014	SP10	001726	SW10	= 002000	TRTVEC=	000014	WRCKHD=	000153
RMER2	= 000042	SP11	001734	SW11	= 004000	TSTDV	046222	WRITE	= 000161
RMHR	= 000036	SP12	001742	SW12	= 010000	TSTNMS	001330	WRL	= 004000
RMINIT	037104	SP7A	001676	SW13	= 020000	TST0	007016	WRTHD	= 000163
RMLA	= 000020	SP7A1	001712	SW14	= 040000	TST1	007204	WRT.AD	044206
RMMR1	= 000024	SP7B	001704	SW15	= 100000	TST10	011620	WRT.RM	044116
RMMR2	= 000040	SP7B1	001720	SW2	= 000004	TST10A	012310	WRT.R1	044202
RMOF	= 000032	SRCHWT	037002	SW3	= 000010	TST10B	012702	WRT.R2	044274
RMR	= 000004	SRCH00	031110	SW4	= 000020	TST11	012750	WRT.R3	044300
RMSN	= 000030	SRTDRV	006232	SW5	= 000040	TST12	013214	WRT.R4	044304
RMTMR	043324	SRTINT	005726	SW6	= 000100	TST13	014136	WRT.R5	044306
RMVEC	037070	SRVCLK	026422	SW7	= 000200	TST14	014640	WRT.WD	044204
RMWC	= 000002	STACK	= 001100	SW8	= 000400	TST15	015400	XXDP	001466
RM.REG	045556	STALL	030664	SW9	= 001000	TST16	016140	\$APTHD	001100
RM05	037654	STALLO	001434	TAB.XY=	001114	TST17	016562	\$ATYC	021310
ROTATE	046261	STALL1	001452	TAP	= 040000	TST2	007412	\$ATY1	021264
RPT	002324	STALL2	001454	TBITVE=	000014	TST20	017226	\$ATY3	021272
RSTRT1	006504	STALL3	001456	TD	042004	TST21	020006	\$ATY4	021302
RSTRT2	006530	START	004624	TEST0	007166	TST22	020624	\$AUTOB	001150
RTC	= 000117	START1	004570	TEST1	007374	TST3	007654	\$BASE	001310
R6	=X000006	START2	004612	TEST10	012052	TST4	010074	\$BDADR	001136
R7	=X000007	START3	004560	TEST11	013134	TST5	010506	\$BDDAT	001142
SAVCSW	001316	START4	004602	TEST12	013612	TST6	010732	\$BELL	001224
SAVEFG	037026	STATBL	001750	TEST13	014346	TST7	011222	\$CHARC	022716
SAVREG=	104412	STKLMT=	177774	TEST14	015050	TYPDS	= 104405	\$CKSWR	023672
SC	042154	STO	043414	TEST15	015610	TYPE	= 104401	\$CMTAG	001114
SCOPE	= 000004	STO1	043450	TEST16	016260	TYPERR	022000	\$CM1	= 000006
SCTRWC=	177400	STO2	043544	TEST17	016702	TYPOC	= 104402	\$CM2	= 000014
SCO	= 000100	STO3	043574	TEST2	007546	TYPON	= 104404	\$CM3	= 000006
SC1	= 000200	STO5	043620	TEST20	017454	TYPOS	= 104403	\$CM4	= 000003
SC11	043006	STO6	043626	TEST21	020302	TYPTIM	031576	\$CNTLC	024604
SC12	043116	STO7	043664	TEST22	020736	T10	001646	\$CNTLG	024616
SC13	043166	STO8	043714	TEST3	010010	T11	001656	\$CNTLU	024611
SC2	= 000400	STO9	043724	TEST4	010244	T12	001666	\$CPUOP	001262
SC3	042224	STRTMR	031274	TEST5	010642	T7A	001606	\$CRLF	001231
SC4	042230	STRT1A	004574	TEST6	011066	T7A1	001626	\$DBLK	023364

\$DB2D	025314	\$GET42	021234	\$MTYP1	001265	\$RTNAD	021256	\$TPB	001166
\$DECVL	025474	\$GTSWR	023762	\$MTYP2	001271	\$\$AVRE	025100	\$TPFLG	001173
\$DEVCT	001244	\$HD =	000000	\$MTYP3	001275	\$\$B2D	025260	\$TPS	001164
\$DEVN	001312	\$HIBTS	001100	\$MTYP4	001301	\$\$SCOPE	024646	\$TRAP	025174
\$DIV	025652	\$HINUM	025646	\$MXCNT	025076	\$\$SETUP=	000167	\$TRAP2	025216
\$DOAGN	021254	\$ICNT	001120	\$NULL	001170	\$\$STUP =	177777	\$TRP =	000014
\$DTBL	023354	\$INTAG	001151	\$NWTST=	000001	\$\$SUPRS	025510	\$TRPAD	025230
\$ENDAD	021244	\$ITEMB	001130	\$SOCNT	023144	\$\$SVLAD	025026	\$TSTM	001104
\$ENDCT	021200	\$LFLG	001232	\$SOMODE	023146	\$\$SWPC =	000220	\$TSTNM	001116
\$ENULL	021260	\$LFLG	021527	\$SOVER	025062	\$\$SWR =	167000	\$TTYIN	024560
\$ENV	001254	\$LONUM	025650	\$SPASS	001242	\$\$SWREG	001256	\$TYPDS	023150
\$ENVM	001255	\$LPADR	001122	\$SPASTM	001106	\$\$SWRMK=	000000	\$TYPE	022366
\$EOP	021020	\$LPERR	001124	\$SQUES	001230	\$TESTN	001240	\$TYPEC	022600
\$EOPCT	021172	\$MADR1	001266	\$SRAND	025550	\$TIMES	001220	\$TYPEX	022720
\$ERFLG	001117	\$MADR2	001272	\$SRDCHR	024234	\$TKB	001162	\$TYPOC	022746
\$ERMAX	001131	\$MADR3	001276	\$SRDLIN	024324	\$TKCNT	023374	\$TYPON	022762
\$ERROR	021532	\$MADR4	001302	\$RDSZ =	000024	\$TKINT	023404	\$TYPOS	022722
\$ERRPC	001132	\$MAIL	001234	\$REGAD	001174	\$TKQEN=	023404	\$UNIT	001246
\$ERRTB	002002	\$MAMS1	001264	\$REG0	001176	\$TKQIN	023376	\$UNITM	001110
\$ERTTL	001126	\$MAMS2	001270	\$REG1	001200	\$TKQOU	023400	\$USWR	001260
\$ESCAP	001222	\$MAMS3	001274	\$REG2	001202	\$TKQSR	023402	\$VECT1	001304
\$ETABL	001254	\$MAMS4	001300	\$REG3	001204	\$TKS	001160	\$VECT2	001306
\$ETEND	001314	\$MBADR	001102	\$REG4	001206	\$TKSRV	023454	\$XOFF =	000023
\$FATAL	001236	\$MFLG	021526	\$REG5	001210	\$TMP0	001212	\$XON =	000021
\$FFLG	021530	\$MNEW	024634	\$RESRE	025136	\$TMP1	001214	\$XTSTR	024660
\$FILLC	001172	\$MSGAD	001250	\$RM02	046056	\$TMP2	001216	\$\$GET4=	000000
\$FILLS	001171	\$MSGGLG	001252	\$RM03	046063	\$TN =	000023	\$OFILL	023145
\$GDADR	001134	\$MSGTY	001234	\$RM05	046070	\$TNPWR	025424	\$.X =	001100
\$GDDAT	001140	\$MSWR	024623						

. ABS. 052776 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 54064 WORDS (212 PAGES)
 DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
 CZRMVA.BIC,CZRMVA/C=CZRMVA.DOC,CZRMVA,SYSMAC/M

SETABL	5-0#													
SETEND	4-319	5-0#												
SFATAL	5-0#	14-3*												
SFFLG	14-3	14-3#	14-3*	14-3*	14-3*									
SFILLC	5-0#	14-103	14-103	14-103	14-103									
SFILLS	5-0#	14-103	14-103	14-103	14-103									
\$GDADR	5-0#	14-:55*	14-:19*	22-13										
\$GDDAT	5-0#	14-:57*	14-:21*	22-13										
\$GET42	14-1#													
\$GTSWR	14-109#	14-115	14-115											
\$HD	4-45	4-45	4-45											
\$HIBTS	4-319#													
\$HNUM	10-21*	11-319	11-512	13-294*	14-123	14-123	14-123#	14-123*	14-:03	14-:09*	14-:12*	14-:56	14-:72	
\$ICNT	5-0#	10-74*	14-111	14-111	14-111	14-111*	14-111*							
\$INTAG	5-0#	14-109	14-109	14-109	14-109	14-109	14-109*							
\$ITEMB	5-0#	14-5	14-5	14-5	14-5*	14-26								
\$LF	5-0#	14-5	14-5	14-103	14-103	14-109	14-109	14-109						
\$LFLG	14-3#	14-3*												
\$LONUM	10-21*	13-295*	14-123	14-123	14-123#	14-123*	14-616	14-:04	14-:08*	14-:11*	14-:55	14-:70	14-:71	
\$LPADR	5-0#	10-21*	11-58*	11-90*	11-99*	11-131*	11-160*	11-229*	11-264*	11-313*	11-382*	11-529*	12-36*	12-60*
	12-161*	12-175*	12-235*	12-249*	12-312*	12-326*	13-26*	13-83*	13-161*	13-290*	13-319*	13-378*	14-111	14-111
	14-111	14-111*	14-111*											
\$LPERR	5-0#	10-21*	11-44*	11-67*	11-97*	11-107*	11-114*	11-129*	11-139*	11-142*	11-158*	11-173*	11-194*	11-227*
	11-238*	11-262*	11-274*	11-304*	11-328*	11-330*	11-380*	11-438*	11-439*	11-527*	12-29*	12-154*	12-228*	12-305*
	13-20*	13-34*	13-38*	13-40*	13-47*	13-53*	13-77*	13-92*	13-102*	13-120*	13-126*	13-156*	13-235*	13-243*
	13-72*	13-288*	13-304*	13-319*	13-329*	13-335*	13-346*	13-352*	13-362*	13-373*	14-5	14-111	14-111	14-111
	14-111*	14-539	14-656											
\$MADR1	5-0#													
\$MADR2	5-0#													
\$MADR3	5-0#													
\$MADR4	5-0#													
\$MAIL	4-319	4-319	5-0#	10-21	10-22	10-66	11-44	11-67	11-97	11-129	11-158	11-227	11-:62	11-304
	11-380	11-527	12-29	12-154	12-228	12-305	13-20	13-77	13-156	13-288	13-373	14-5	14-103	14-111
\$MAMS1	5-0#													
\$MAMS2	5-0#													
\$MAMS3	5-0#													
\$MAMS4	5-0#													
\$MBADR	4-319#													
\$MFLG	14-3	14-3#	14-3*	14-3*										
\$MNEW	14-109	14-109#												
\$MSGAD	5-0#	14-3	14-3*											
\$MSGLG	5-0#	14-3*												
\$MSGTY	5-0#	14-3	14-3	14-3*	14-3*									
\$MSWR	14-109	14-109#												
\$MTYP1	5-0#													
\$MTYP2	5-0#													
\$MTYP3	5-0#													
\$MTYP4	5-0#													
\$MXCNT	14-111	14-111	14-111	14-111#										
\$NULL	5-0#	14-103	14-103	14-103										
\$NWTST	11-44	11-44	11-44#	11-44#	11-67	11-67	11-67#	11-67#	11-97	11-97	11-97#	11-97#	11-129	11-129
	11-129#	11-129#	11-158	11-158	11-158#	11-158#	11-227	11-227	11-227#	11-227#	11-262	11-262	11-262#	11-262#
	11-304	11-304	11-304#	11-304#	11-380	11-380	11-380#	11-380#	11-527	11-527	11-527#	11-527#	12-29	12-29
	12-29#	12-29#	12-154	12-154	12-154#	12-154#	12-228	12-228	12-228#	12-228#	12-305	12-305	12-305#	12-305#
	13-20	13-20	13-20#	13-20#	13-77	13-77	13-77#	13-77#	13-156	13-156	13-156#	13-156#	13-288	13-288
	13-288#	13-288#	13-373	13-373	13-373#	13-373#								

SOCNT 14-105# 14-105• 14-105•

C 16

SEQ 0197

STKSRV 14-109 14-109#

E 16

S:0 0199

.SASTA 14-3 14-3

G 16

SEQ 0201

.SX	4-319	4-319#				
A16	4-88#					
A17	4-89#					
ABASE	5-0	5-0				
ACCESS	6-0	18-40#				
ACDW1	5-0					
ACDW2	5-0					
ACPUOP	5-0	5-0				
ACTDRV	15-92#	15-361*	15-412*	15-738*	15-747*	15-:09
ACTSTR	15-98#	15-:11*	15-:25*			
ADDW0	5-0					
ADDW1	5-0					
ADDW10	5-0					
ADDW11	5-0					
ADDW12	5-0					
ADDW13	5-0					
ADDW14	5-0					
ADDW15	5-0					
ADDW2	5-0					
ADDW3	5-0					
ADDW4	5-0					
ADDW5	5-0					
ADDW6	5-0					
ADDW7	5-0					
ADDW8	5-0					
ADDW9	5-0					
ADEVCT	5-0	5-0				
ADEVN	5-0	5-0				
AENV	5-0	5-0				
AENVN	5-0	5-0				
AFATAL	5-0	5-0				
AMADR1	5-0	5-0				
AMADR2	5-0	5-0				
AMADR3	5-0	5-0				
AMADR4	5-0	5-0				
AMAMS1	5-0	5-0				
AMAMS2	5-0	5-0				
AMAMS3	5-0	5-0				
AMAMS4	5-0	5-0				
AMSGAD	5-0	5-0				
AMSGLG	5-0	5-0				
AMSGTY	5-0	5-0				
AMTYP1	5-0	5-0				
AMTYP2	5-0	5-0				
AMTYP3	5-0	5-0				
AMTYP4	5-0	5-0				
AOE	4-160#					
APASS	5-0	5-0				
APRIOR	5-0					
APTCSU	14-3#	14-103				
APTENV	14-3	14-3#	14-5		14-103	
APTSIZ	10-21	14-3#				
APTSP0	14-3	14-3#	14-103			
ASWREG	5-0	5-0				
ATO	4-208#					
AT1	4-209#					

AT2

4-210#

I 16

SEQ 0203

AT3	4-211#													
AT4	4-212#													
AT5	4-213#													
AT6	4-214#													
AT7	4-215#													
ATA	4-147#													
ATABIT	10-157	14-<96	14-=02	15-152#	15-327	15-424	15-530	15-928	15-950	15-958	15-959	15-979	15-:69	
AATESTN	5-0	5-0												
AUNIT	5-0	5-0												
AUSWR	5-0	5-0												
AVECT1	5-0	5-0												
AVECT2	5-0	5-0												
BAI	4-106#													
BASFLG	6-0#	11-69*	13-337	13-354	14-464*	14-492*	14-516*							
BIT0	4-80#													
BIT00	4-80	4-80#	6-0	6-0	10-140	10-221	14-243							
BIT01	4-80	4-80#	6-0	6-0	14-541	14-569	15-404	15-659						
BIT02	4-80	4-80#	6-0	6-0	14-541	14-572								
BIT03	4-80	4-80#	6-0	6-0	14-578	15-886	15-:85							
BIT04	4-80	4-80#	6-0	6-0	14-575	15-921								
BIT05	4-80	4-80#	6-0	6-0	14-578	15-901								
BIT06	4-80	4-80#	6-0	6-0	14-581	15-337	15-395	15-469	15-782	15-<45				
BIT07	4-80	4-80#	6-0	6-0	15-337	15-604	15-759	15-886	15-901	15-921	15-954	15-<17		
BIT08	4-80	4-80#	6-0	14-569	15-337									
BIT09	4-80	4-80#	6-0	14-5	14-111	14-584	15-:60							
BIT1	4-80#													
BIT10	4-80#	6-0	14-5	14-541	14-572	14-@27	15-661							
BIT11	4-80#	6-0	14-111	14-255	14-572	15-290	15-431	15-625	15-995					
BIT12	4-80#	6-0	14-541	14-575	15-284	15-318	15-337	15-406	15-439	15-619	15-642	15-657	15-671	15-911
	15-913	15-:25	15-:80	15-<46										
BIT13	4-80#	6-0	14-5	14-541	14-569	15-389	15-860	15-:28						
BIT14	4-80#	6-0	14-111	14-473	14-541	14-575	15-401	15-436	15-847	15-990	15-:90	15-:35	15-:39	15-<43
BIT15	4-80#	6-0	14-353	14-383	14-421	14-462	14-490	14-514	14-590	14-707	14-708	15-389	15-401	15-404
	15-406	15-436	15-439	15-625	15-659	15-661	15-782	15-886	15-901	15-911	15-921	15-990	15-:60	15-:90
	15-:97													
BIT2	4-80#	15-:97												
BIT3	4-80#													
BIT4	4-80#													
BIT5	4-80#													
BIT6	4-80#													
BIT7	4-80#													
BIT8	4-80#													
BIT9	4-80#													
BITS	6-0#	11-44	11-67	11-97	11-129	11-158	11-227	11-262	11-304	11-380	11-527	12-29	12-154	12-228
	12-305	13-20	13-77	13-156	13-219	13-288	13-373	14-=62	14-=96	14->13	14->70	14-@05		
BLNKS1	14-?29	18-57#												
BLNKS2	10-115	14-58	14-84	14-91	14-98	14-814	14-824	14-839	14->96	14-?25	18-56#			
BLNKS3	18-55#													
BLNKS4	10-89	18-54#												
BPTVEC	4-80#													
BUFFER	11-391	13-33	13-45	13-56	13-90	13-100	13-115	13-191	13-298	13-325	13-334	13-344	13-351	13-361
	14-637*	14-638	14-640	14-642	14-643	14-644	14-706	14-743	14-910	14-929	14-930	14-954	14-:14	14-:33
	14-:47	14-:88	14-:05	14-:06	14-:23	17-8	17-32	17-56	17-80	23-104#				
BUSADR	6-0#	10-3*	10-6*	10-10*	10-13*	16-13	16-15*							
BYPASS	6-0#	10-219*	11-58*	11-80*	11-105*	11-137*	11-170*	11-235*	11-270*	11-313*	11-438*	11-539*	12-60*	12-175*
	12-249*	12-326*	13-26*	13-83*	13-219*	13-293*	14-523	14-973	14-:75	14-:40				

C.SWR 6-0# 10-139 10-142 10-163 10-221 11-170 12-64^{K 16} 12-124 13-300 14-192 14-202 14-212 14-243 14-325
SEQ 0205

DF21 7-190 23-49#

B 1

SEQ 0207

DRVCLR 4-285# 11-459

D 1

SEQ 0209

DTE 4-163#

F 1

SEQ 0211

F1

4-128#

H 1

SEQ 0213

F2	4-129#													
F3	4-130#													
F4	4-131#													
F5	4-132#													
FC	8-7#	11-74	11-106	11-118	11-138	11-171	11-180	11-212	11-236	11-249	11-271	11-315	11-316	11-323
	11-386	11-387	11-388	11-512	11-534	11-535	11-546	12-60	12-60	12-175	12-177	12-202	12-249	12-268
	12-326	12-345	13-29	13-87	13-219	13-256	13-385	14-:87	14-<05					
FER	4-155#													
FILBUF	13-28	13-85	14-908#											
FILRAN	13-302	14-:88#												
FMT16	4-255#													
FS	8-13#	11-47	11-70	11-100	11-132	11-161	11-230	11-265	11-530	11-531	12-60	13-88	13-165	13-182
	13-190													
FT	8-10#	11-48	11-72	11-101	11-133	11-162	11-231	11-266	11-306	11-355	11-385	11-532	11-533	12-60
	13-30	13-219												
FWD	18-42#													
GETADR	10-69	16-13#												
GETNUM	14-<28	14-<50#	16-19	16-26										
GETREG	4-298#													
GETREQ	15-426	15-654	15-774	15-841	15-875	15-939	15-989	15-:59	15-:87	15-:94	15-=74#			
GETSWR	10-75	14-<23#												
GNS	4-308	4-308	10-22	10-43	10-54	10-60	10-61	14-1	14-1	14-1	14-1	14-47	14-61	14-115
	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-115
	14-115	14-115	14-115	14-115	14-115	14-115	14-115	14-<25	14-<90	14-=06	14->72	14-?51		
GO	4-127#													
GT.PR1	14-<89#	14-<93	14-<93	14-<93	14-<93	14-<93	14-<95	14-<95						
GT.PR2	14-=03#	14-=14	14->37	14->46	14-@04									
GT.PRM	10-144	14-<88#												
GTSWR	10-22	10-66	14-115#											
GTTST1	14-=01	14-=06#	14-=54	14-=67	14-=81	14-=84	14->10	14->17	14->26	14->32	14->41	14->61		
GTTST2	14-=19#	14->21												
GTTST3	14-=22	14-=26	14-=30	14-=34	14-=38	14->03#								
GTTST4	14-=40	14-=64	14->01	14->04#										
GTTST5	14->05	14->27#												
GTTST6	14->29	14->38#												
HCE	4-158#													
HCI	4-253#													
HCRC	4-159#													
HT	4-80#	14-103	14-103											
IAE	4-161#													
IC	8-9#	11-110	11-117	11-141	11-190	11-211	11-243	11-244	11-291	11-389	11-476	11-512	11-512	11-543
	11-547	14-882												
IE	4-86#													
ILF	4-151#													
ILR	4-152#													
INCCYL	13-219	14-880#												
INCSK	11-108#	11-112												
INCTRK	13-258	14-863#												
IOTVEC	4-80#	10-21*	10-21*											
IR	4-109#													
ISR	11-438	12-123	12-213	12-290	12-367	14-671	15-239	15-738#						
IT	8-12#	14-865												
ITEM41	7-247#	14-33												
LA	15-456	15-699#												
LACNT	15-113#	15-707*	15-708	15-729*										
LC	8-8#	11-49	11-75	11-111	11-113	11-146	11-191	11-193	11-201	11-237	11-245	11-272	11-316	11-321

11-477 11-512 11-512 11-544 11-548 12-197 12-256^J 1¹ 12-333 13-223 13-310 13-380 14-880 14-883 14-885
SEQ 0215

MSG12X 6-0 18-62#

L 1

SEQ 0217

OPN.A1 14-?19 14-?41 14-?97#

N 1

SEQ 0219

PRM13 8-26 8-192#

C 2

SEQ 0221

RECAL 4-284# 10-227 10-227 11-46 11-46 14-651 14-676^E 2

SEQ 0223

SAVEFG 10-79* 15-124# 15-605 15-766 15-955

G 2

SEQ 0225

SAVREG	12-100	12-109	12-119	12-191	12-209	12-263	12-275	12-286	12-340	12-352	12-363	14-18	14-115#	14-119
	14-701	14-908	14-927	14-951	14-:13	14-:32	14-<88	14->50	14-?45	14-@11	15-221	15-362	15-422	15-641
	15-739	15-:12	15-<07	15-=13										
SC	15-745	15-778	15-781	15-787	15-809#									
SCO	4-239#													
SC1	4-240#													
SC11	15-877	15-885	15-946#											
SC12	15-843	15-854	15-965#											
SC13	15-833	15-836	15-975#											
SC2	4-241#													
SC3	15-824#	15-828												
SC4	15-826#	15-857	15-869	15-871	15-920	15-936	15-945	15-964	15-:02					
SC5	15-825	15-831#												
SC6	15-853	15-872#												
SC6A	15-851	15-897#												
SC7	15-887	15-900	15-902	15-912	15-922#									
SC8	15-866	15-880	15-890	15-896	15-905	15-935	15-937#	15-974						
SCOPE	4-80#	11-58	11-84	11-120	11-148	11-214	11-251	11-293	11-356	11-514	11-550	12-143	12-217	12-294
	12-371	13-63	13-134	13-261	13-364	13-390								
SCTRW	4-304#	13-49	13-89	13-314										
SEARCH	4-291#	12-95	12-103											
SEC.DS	6-0#	14-349*	14-376*	14-414*	14-458*	14-486*	14-510*	14-645*	14-685*	14-:62*	22-10	22-11	22-12	22-23
	22-26													
SEC.RD	6-0#	14-644*	22-11											
SEEK	4-283#	11-307	11-307	11-383	11-383	11-438	12-60	12-60	12-175	12-175	12-185	12-249	12-249	12-257
	12-269	12-326	12-326	12-334	12-346	13-381	13-381	13-386	13-386	14-330	14-331			
SEEKFG	10-162*	10-165*	15-132#	15-226	15-454									
SEKNT	6-0#	11-473	11-475*	11-512*										
SEKMR	6-0#	11-438*	11-512*											
SELDRV	4-300#													
SERIAL	10-232	18-35#												
SET.IE	15-255	15-286	15-397	15-471	15-685	15-818	15-<39#							
SETBUF	11-393	13-232	14-:13#											
SETFOR	4-299#	10-224	10-227	11-46	11-307	11-383	12-60	12-175	12-249	12-326	13-381	13-386		
SETVEC	10-78#													
SKI	4-270#													
SLASH	14-<54	14-?02	18-19#											
SP10	6-0#	12-216												
SP11	6-0#	12-293												
SP12	6-0#	12-370												
SP7A	6-0#	12-82	12-138											
SP7A1	6-0#	12-76												
SP7B	6-0#	12-73	12-142											
SP7B1	6-0#	12-68												
SPTYP	12-138	12-141	12-215	12-292	12-369	14-758#								
SRCHOO	12-37	12-162	12-236	12-313	14-669#									
SRCHWT	15-86#	15-424*	15-530*	15-950	15-959*									
SRTDRV	10-83	10-136#												
SRTINT	10-68	10-71#												
SRVCLK	14-204	14-214	14-219#											
ST.CLK	10-76	12-122	12-212	12-289	12-366	14-171#								
ST.LCL	14-186	14-211#												
ST.PCL	14-180	14-201#												
STACK	4-80#	10-21	10-202	11-58	11-81	11-107	11-114	11-139	11-142	11-173	11-194	11-238	11-274	11-314
	11-328	11-330	11-438	11-439	11-540	12-88	12-176	12-250	12-327	13-27	13-34	13-38	13-40	13-47
	13-53	13-84	13-92	13-102	13-120	13-126	13-219	13-235	13-243	13-252	13-304	13-320	13-329	13-335

13-346 13-352 13-362 13-379

1 2

SEQ 0227

11-263 11-305 11-381 11-528 12-30 12-155 12-229^K 2 12-306 13-21 13-78 13-157 13-228 13-233 13-241
SEQ 0229

TSTO 11-44#

M 2

SEQ 0231

WCF

4-156#

8 3

SEQ 0233

