

RK611/RK06

DRIVE COMPATIBILITY PROG
MD-11-DZR6Q-A

EP-DZR6Q-A-DL-A
COPYRIGHT © 1977
FICHE 1 OF 2

JUN 1977
digital
MADE IN USA

The microfiche card displays a grid of 144 frames, arranged in 12 rows and 12 columns. Each frame contains a small, high-contrast image of a document page, likely a technical manual or specification sheet. The text in the frames is too small to be legible but appears to be organized into tables and lists. The overall layout is consistent across all frames, suggesting a continuous sequence of pages from a single document.

RK611/RK06

DRIVE COMPATIBILITY PROG
MD-11-DZR6Q-A

EP-DZR6Q-A-DL-A
COPYRIGHT © 1977
FICHE 2 OF 2

JUN 1977
digital
MADE IN USA



B01

EOF10ZLAEBSEQ

00010000

770526

PDP10 411

N-HDR10ZR60ASEQ

00010000

770526

.REM 3

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

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZR60-A-D
PRODUCT NAME:	UNIBUS RK06 DRIVE COMPATIBILITY PROGRAM
DATE:	JANUARY 1977
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	DAVE HOFFMAN

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

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

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

COPYRIGHT (C) 1977 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

- 1.0 INTRODUCTION
- 2.0 HARDWARE REQUIREMENTS
- 3.0 PRELIMINARY PROGRAM REQUIREMENTS
- 4.0 GENERAL PROGRAM CONSIDERATIONS
 - 4.1 SYSMAC
 - 4.2 XXDP
 - 4.3 ACT
 - 4.4 APT
 - 4.5 DUAL-ACCESS
 - 4.6 MEMORY MANAGEMENT
 - 4.7 MEMORY PARITY OPTION
 - 4.8 BAD SECTORS
 - 4.9 EXECUTION TIME
- 5.0 PROGRAM LOAD MEDIA
- 6.0 PROGRAM OPTIONS
 - 6.1 STARTING ADDRESSES
 - 6.2 SWITCH REGISTER OPTIONS USED
- 7.0 RUNNING THE PROGRAM
- 8.0 OPERATIONAL DIALOGUE
 - 8.1 DIALOGUE FOR ADDRESS 200 START
 - 8.2 DIALOGUE FOR ADDRESS 204 START
 - 8.3 DIALOGUE FOR ADDRESS 220 START
 - 8.4 PASS 1 DIALOGUE
 - 8.5 PASS 2 DIALOGUE
- 9.0 DESCRIPTION OF TESTS
 - 9.1 DESCRIPTION OF PASS 1 TESTS
 - 9.2 DESCRIPTION OF PASS 2 TESTS
- 10.0 PRINTOUT OF TEST RESULTS
 - 10.1 OVERWRITE AND DRIVE COMPATIBILITY DATA TEST RESULTS
- 11.0 ERROR REPORTING
 - 11.1 COMMON ERRORS
 - 11.2 ERROR HANDLING
 - 11.3 ERROR PRINTOUT EXAMPLE
- TABLE A - BASIC READ/WRITE TEST SECTORS
- TABLE B - WORSE CASE DATA PATTERN

47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

E01

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 4

SEQ 0003

103
104
105
106
107
108
109
110
111
112
113
114

TABLE C - CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE
TABLE D - BASIC CYLINDER BLOCK LAYOUT EXAMPLE
TABLE E - OVERWRITE CYLINDERS
TABLE F - SELF-TEST CYLINDERS
TABLE G - PSEUDO-RANDOM DATA PATTERN

1.0 INTRODUCTION

THE PURPOSE OF THIS PROGRAM IS TO VERIFY THE COMPATIBILITY OF UP TO 16 RK06 DRIVES WHICH MAY RESIDE ON ONE OR MORE RK611/RK06 SUBSYSTEMS (INCLUDING SYSTEMS WITH MULTIPLE SUBSYSTEMS). COMPATIBILITY IS DEFINED HERE AS THE ABILITY OF A DRIVE TO WRITE DATA WHICH CAN BE READ SUCCESSFULLY BY ALL OTHER DRIVES, AND ADDITIONALLY THE ABILITY OF A DRIVE TO COMPLETELY OVER-WRITE DATA WRITTEN BY ALL OTHER DRIVES.

THE PROGRAM IS DESIGNED TO DETECT THE FOLLOWING CONDITIONS WHICH MOST COMMONLY CAUSE INCOMPATIBILITY BETWEEN DRIVES:

1. HEAD MIS-ALIGNMENT
2. POSITIONER LATERAL MISALIGNMENT
3. SPINDLE-CARTRIDGE INTERFACE RUNOUT
4. IMPROPER LEVELS OF WRITE CURRENT
5. INCORRECT ADDRESSING OF READ/WRITE HEADS

THE TESTING IS DONE IN TWO PASSES. IN PASS 1, COMPATIBILITY DATA PATTERNS ARE WRITTEN BY ALL THE DRIVES UPON THE SAME DISK CARTRIDGE, AND THE BASIC READ/WRITE CAPABILITY OF EACH DRIVE IS DEMONSTRATED. IN PASS 2, THE COMPATIBILITY DATA FROM ALL DRIVES IS READ BY EACH DRIVE, WITH INCREASING HEAD OFFSET, AND THIS IS COMPARED WITH EACH DRIVE'S ABILITY TO READ ITS OWN DATA. IN ADDITION, EACH DRIVE'S CAPABILITY TO OVERWRITE DATA WRITTEN BY ALL OTHER DRIVES IS TESTED ON THE SECOND PASS. (FOR THE REMAINDER OF THIS SPECIFICATION, THE ABOVE DEFINITIONS OF THE FIRST AND SECOND PASS SHALL APPLY).

IN BOTH PASSES, THE PROGRAM DIRECTS THE OPERATOR IN THE LOADING AND UNLOADING OF DRIVES AND THE MOVEMENT OF THE CARTRIDGE FROM DRIVE TO DRIVE, THROUGH MESSAGES AT THE CONSOLE TERMINAL. AT THE COMPLETION OF TESTING ON EACH DRIVE DURING THE SECOND PASS A SUMMARY IS PRINTED OF COMPATIBILITY TEST RESULTS FOR THAT DRIVE.

WITHIN THE VARIOUS TESTS OF BOTH PASSES, THE CAPABILITY IS PROVIDED TO LOOP ON CURRENT OPERATIONS, AND SWITCH REGISTER OPTIONS ARE PROVIDED, FOR A VARIETY OF LOOPING, RUNNING, AND REPORTING MODES (SEE SECTION 6.2).

UNEXPECTED ERRORS WILL BE REPORTED AS THEY OCCUR. THE REPORT WILL INCLUDE A TEST NUMBER AND DESCRIPTION, CURRENT AND PREVIOUS OPERATIONS GOOD AND BAD TEST DATA, AND APPLICABLE DEVICE REGISTER CONTENTS.

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

165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE RK06 DRIVE COMPATIBILITY PROGRAM, FOR EACH SUBSYSTEM WHICH MAY BE PRESENT:

PDP-11/04, (05,10 MFG. ONLY), 20,30,34,35,40,45,50,70, OR PDQ
16K MEMORY
CONSOLE TERMINAL
RK06 CONTROLLER
1 TO 8 RK06 DISK DRIVES PER CONTROLLER

IN ADDITION, A SINGLE RK06 DISK CARTRIDGE IS REQUIRED WHICH MAY BE FORMATTED IN EITHER 20 OR 22 SECTOR FORMAT, ON A RELIABLE WELL-ALIGNED RK06 DRIVE. THIS CARTRIDGE WILL BE MOVED FROM DRIVE TO DRIVE, (ON UP TO 16 DRIVES) ON EACH OF TWO PASSES.

3.0 PRELIMINARY PROGRAM REQUIREMENTS

BEFORE RUNNING THE RK06 DRIVE COMPATIBILITY PROGRAM, THE SUBSYSTEM(S) UNDER TEST SHOULD BE CAPABLE OF PASSING THE CONTROLLER DIAGNOSTICS DZR6A-DZR6E AND DZR6K, THE DRIVE DIAGNOSTICS DZR6H-DZR6J, AND THE SUBSYSTEM VERIFICATION PROGRAMS DZR6M-DZR6N. IN ADDITION, THE CARTRIDGE MUST BE FORMATTED IN 20 OR 22 SECTOR FORMAT USING THE PACK FORMATTER DZR6L, OR EQUIVALENT (PERFORMED ON KNOWN GOOD, ALIGNED DRIVE).

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 SYSMAC

THIS PROGRAM USES PORTIONS OF THE SYSMAC DIAGNOSTIC SYSTEM MACRO PACKAGE.

4.2 XXDP

THIS PROGRAM MAY BE LOADED UNDER XXDP, AND MAY BE RUN IN DUMP MODE ONLY. DUE TO MANUAL INTERVENTION AND LACK OF END-OF-PASS HOOKS, THE PROGRAM IS NOT XXDP CHAINABLE.

214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267

4.3 ACT

THIS PROGRAM MAY BE LOADED UNDER ACT AND MAY BE RUN IN DUMP MODE ONLY. IT IS NOT CHAINABLE UNDER ACT.

4.4 APT

THIS PROGRAM MAY BE LOADED BY THE APT SYSTEM, BUT MAY BE RUN IN PROGRAM (DUMP) MODE ONLY. IT CANNOT BE RUN IN APT SCRIPT MODE.

4.5 DUAL-ACCESS

THIS PROGRAM DOES NOT UTILIZE THE DUAL-ACCESS OPTION IN ANY WAY, AND ALL DRIVES UNDER TEST SHOULD BE DE-SELECTED THROUGH THE PORT WHICH IS NOT IN USE.

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM. IF IT IS INSTALLED, IT IS DISABLED BY THE PROGRAM.

4.7 MEMORY PARITY OPTION

IF PARITY MEMORY IS INSTALLED, MEMORY PARITY TRAPS ARE DISABLED BY THE PROGRAM.

4.8 BAD SECTORS

THE LIST OF BAD SECTORS ON THE CARTRIDGE IS OBTAINED FROM THE FIRST DRIVE TO BE TESTED ON THE CURRENT SUBSYSTEM. ACCORDING TO A SWITCH REGISTER OPTION (SEE SECTION 6.2) THIS LIST MAY BE TYPED AT THE CONSOLE AT THE START OF THE FIRST PASS. ALL DATA ERRORS OCCURING IN THE PROGRAM ARE IGNORED IF THEY OCCUR IN SECTORS DESIGNATED AS BAD.

4.9 EXECUTION TIME

THE TOTAL TIME REQUIRED TO RUN THE DRIVE COMPATIBILITY PROGRAM IS DIRECTLY PROPORTIONAL TO THE NUMBER OF DRIVES TO BE TESTED AND REQUIRES ABOUT 8-10 MINUTES PER DRIVE.

268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

5.0 PROGRAM LOAD MEDIA

THIS PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR FROM THE ACT OR APT SYSTEMS OR FROM ANY MEDIA SUPPORTED BY XXDP.

6.0 PROGRAM OPTIONS

6.1 STARTING ADDRESSES

200 - THIS IS THE STARTING ADDRESS FOR DEFAULT PARAMETERS AND RUNNING OF PASS 1 AND PASS 2 ON A SINGLE SUBSYSTEM. THE PROGRAM WILL USE DEFAULT RK611 BASE ADDRESS, INTERRUPT VECTOR AND PRIORITY, AND WILL AUTOMATICALLY DETERMINE WHICH DRIVES TO TEST. DRIVES MUST BE ON-LINE, POWERED UP, AND EITHER LOADED OR UNLOADED. THE PROGRAM WILL ASSUME ALL DRIVES TO BE TESTED RESIDE ON ONE RK611/RK06 SUBSYSTEM ONLY.

204 - THIS IS THE STARTING ADDRESS TO RUN PASS 1 ON ALL RK611/RK06 SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RK611 BASE ADDRESS, INTERRUPT VECTOR, AND PRIORITY FOR EACH SUBSYSTEM ON THIS SYSTEM, AND IT ASKS FOR THE LETTER NAMES (A THRU P) ASSIGNED TO ALL OTHER SUBSYSTEMS, AND THE DRIVE(S) WHICH WILL BE TESTED ON EACH.

220 - THIS IS THE STARTING ADDRESS TO RUN PASS 2 ON ALL RK611/RK06 SUBSYSTEMS WHICH RESIDE ON THIS PDP-11 SYSTEM. THE PROGRAM WILL ASK FOR THE RK611 BASE ADDRESS, INTERRUPT VECTOR, AND PRIORITY FOR EACH SUBSYSTEM ON THIS SYSTEM, AND IT ASKS FOR THE LETTER NAMES (A THRU P) ASSIGNED TO ALL OTHER SUBSYSTEMS, AND THE DRIVE(S) WHICH WILL BE TESTED ON EACH.

6.2 SWITCH REGISTER OPTIONS USED

THIS PROGRAM IS DESIGNED TO ALLOW THE USE OF THE HARDWARE SWITCH REGISTER IF PRESENT, OR THE SYSMAC-SUPPORTED SOFTWARE SWITCH REGISTER (IF HARDWARE SWR IS NOT PRESENT, OR IS SET TO ALL ONES). IN EITHER CASE, THE FOLLOWING OPTIONS ARE IMPLEMENTED WHEN THE APPROPRIATE BITS ARE SET TO 1:

BIT	OPTION
---	-----
15	HALT ON ERROR
14	LOOP ON CURRENT TEST

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378

13	INHIBIT ERROR REPORTS
12	REPORT DESCRIPTION ONLY, ON ERRORS
11	UNUSED
10	BELL ON ERROR
09	LOOP ON ERROR
08	APPLY RANDOM STALL BETWEEN OPERATIONS
07	TYPE BAD SECTOR FILES (BSF'S) AT START
06-00	UNUSED

7.0 RUNNING THE PROGRAM

ONCE THE PROGRAM HAS BEEN LOADED INTO CORE (IN A GIVEN SYSTEM, IF THERE ARE MULTIPLE SYSTEMS) THE FOLLOWING STEPS MUST BE TAKEN TO RUN THE PROGRAM:

1. INSURE THAT ALL DRIVES TO BE TESTED ARE POWERED UP AND SINGLE PORT SELECTED.
2. LOAD THE DESIRED START ADDRESS.
3. SET ANY DESIRED BITS IN THE HARDWARE SWITCH REGISTER (IF PRESENT).
4. START THE PROGRAM.
5. FOLLOW ALL INSTRUCTIONS TYPED BY THE PROGRAM PERTAINING TO THE MANUAL INTERVENTION REQUIRED, AND THE ALTERNATE USE OF MULTIPLE SYSTEMS (IF THERE ARE ANY).

8.0 OPERATIONAL DIALOGUE

THIS SECTION DESCRIBES THE CONSOLE TERMINAL DIALOGUE THROUGH WHICH THE PROGRAM DIRECTS THE OPERATOR, IN THE SELECTION OF OPTIONS AND THE LOADING AND UNLOADING OF DRIVES, AND THE MOVEMENT OF THE TEST CARTRIDGE. THE EXACT DIALOGUE WHICH IS USED DEPENDS UPON THE STARTING ADDRESS WHICH WAS CHOSEN (SEE SECTION 6.1).

IN THE FOLLOWING DISCUSSION AND IN THE PRINTOUT OF TEST RESULTS, DRIVES TO BE TESTED WILL BE REFERRED TO BY A LETTER AND A NUMBER. THE LETTER IS THE SUBSYSTEM LETTER NAME (OPERATOR ASSIGNED), AND THE NUMBER IS THE DRIVE NUMBER ON THAT SUBSYSTEM. FOR EXAMPLE, DRIVE C6 REFERS TO DRIVE 6 ON SUBSYSTEM C.

379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433

8.1 DIALOGUE FOR ADDRESS 200 START

THIS STARTING ADDRESS MAY BE USED FOR AUTOMATIC DRIVE SIZING AND DEFAULTING OF PARAMETERS, WHEN THE DRIVES RESIDE ON ONE SUBSYSTEM ONLY. THE PROGRAM FIRST IDENTIFIES ITSELF AS FOLLOWS:

MD-11-DZR60 - RK06 DRIVE COMPATIBILITY PROGRAM

THEN, THE DRIVES ON THE SYSTEM ARE AUTOMATICALLY SIZED, AND ALL EXISTING DRIVES WILL BE MARKED FOR TESTING. THE PROGRAM TYPES THE DRIVE LIST, AS IN THE FOLLOWING EXAMPLE:

DRIVES = 2,5,7

THE PROGRAM NOW PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4.

PLEASE NOTE THAT THERE IS ONLY ONE SUBSYSTEM ON AN ADR. 200 START, AND IT IS NAMED SUBSYSTEM A. THE DRIVES IN THE ABOVE EXAMPLE WOULD BE REFERRED TO AS A2, A5, A7 IN THE TEST RESULTS PRINTOUT AT THE END OF PASS 2.

8.2 DIALOGUE FOR ADDRESS 204 START

THIS STARTING ADDRESS MUST BE USED ON EACH SYSTEM, WHEN THERE IS MORE THAN ONE SUBSYSTEM, BUT IT MAY ALSO BE USED WHEN THERE IS JUST ONE SUBSYSTEM (TOTAL), TO SPECIFY DRIVES TO TEST AND NON-DEFAULT PARAMETER VALUES, FOR PASS 1.

THE PROGRAM IDENTIFIES ITSELF AS FOLLOWS:

MD-11-DZR60 - RK06 DRIVE COMPATIBILITY PROGRAM

THEN, THE PROGRAM ASKS THE OPERATOR FOR THE DRIVES TO BE TESTED ON EACH OF THE POSSIBLE SUBSYSTEMS (STARTING WITH A - THE NAMES RANGE FROM SUBSYS A TO SUBSYS P. THERE COULD BE UP TO 16 SUBSYSTEMS, WITH A DRIVE ON EACH) :

SUBSYS A DRIVE(S) =

THE OPERATOR THEN TYPES THE DESIRED DRIVE NUMBERS, AS IN THE FOLLOWING EXAMPLE:

SUBSYS A DRIVE(S) = 2,5,7

THE PROGRAM THEN VERIFIES THE DRIVE NOS. BY TYPING :

WILL TEST DRIVE(S) 2,5,7 ON SUBSYS A.

NEXT, THE PROGRAM ASKS THE FOLLOWING QUESTION:

434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489

IS THERE ANOTHER SUBSYS (Y OR N <CR>)?

THE OPERATOR TYPES Y<CR> OR N<CR>. (IF JUST <CR> IS TYPED, THE PROGRAM ASSUMES THAT N<CR> WAS TYPED). IF THE OPERATOR TYPED N, THE PROGRAM PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4. IF Y WAS TYPED, THE PROGRAM ASKS FOR THE NUMBERS OF THE DRIVES TO BE TESTED ON THE NEXT SUBSYSTEM (SUBSYS B) AS FOLLOWS:

SUBSYS B DRIVE(S) =

THE OPERATOR TYPES THE DRIVE NUMBERS, AS IN THE FOLLOWING EXAMPLE:

SUBSYS B DRIVE(S) = 2,3

THE PROGRAM THEN VERIFIES THE DRIVE NOS. BY TYPING :

WILL TEST DRIVE(S) 2,3 ON SUBSYS B.

NEXT, THE PROGRAM WILL ASK :

IS THERE ANOTHER SUBSYS (Y OR N <CR>)?

AND IN THE SAME MANNER, THE OPERATOR SPECIFIES THE DRIVES ON EACH OF THE REMAINING SUBSYSTEMS, UNTIL ALL HAVE BEEN SPECIFIED.

NEXT, THE PROGRAM ASKS FOR THE LETTER NAME ASSIGNED TO THE SUBSYSTEM TO BE TESTED NEXT :

TYPE NAME OF NEXT SUBSYS TO TEST (A-P) :

THE OPERATOR RESPONDS, AS IN THE FOLLOWING EXAMPLE :

TYPE NAME OF NEXT SUBSYS TO TEST (A-P) : A

ALL SUBSYSTEMS MUST BE TESTED IN THE ORDER IN WHICH THE LETTERS ARE ASSIGNED (A THRU P). NEXT, THE PROGRAM ALLOWS THE OPERATOR TO ALTER THE RK611 BUS ADDRESS, VECTOR ADDRESS, AND INTERRUPT PRIORITY FOR THIS SUBSYSTEM. FOR EACH PARAMETER THE CURRENT VALUE IS TYPED, AND THE OPERATOR IS GIVEN THE OPPORTUNITY TO TYPE IN A NEW VALUE, PLUS <CR>. IF JUST <CR> IS TYPED, THE PARAMETER IS NOT CHANGED. WHEN THE PROGRAM IS FIRST LOADED, THE FOLLOWING DEFAULT VALUES ARE ASSIGNED: RK611 BUS ADDRESS = 177440, VECTOR ADDRESS = 210, AND PRIORITY = 5. THE FOLLOWING EXAMPLE SHOWS A PRINTOUT IN WHICH ONLY THE VECTOR WAS CHANGED:

RK611 BUS ADR	=	177440	NEW =
RK611 VEC ADR	=	210	NEW = 240
RK611 PRIORITY	=	5	NEW =

THEN THE PROGRAM PROCEEDS WITH PASS 1, AND DIRECTS THE OPERATOR IN THE MOUNTING OF THE PACK, AS DESCRIBED IN SECTION 8.4. AT THE COMPLETION OF PASS 1 ON THE SUBSYSTEM, THE PROGRAM WILL INFORM THE OPERATOR HOW

490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
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

TO PERFORM PASS 1 ON THE NEXT SUBSYSTEM.

8.3 DIALOGUE FOR ADDRESS 220 START

THIS STARTING ADDRESS MUST BE USED ON EACH SYSTEM WHEN THERE IS MORE THAN 1 SUBSYSTEM, BUT IT MAY ALSO BE USED WHEN THERE IS JUST ONE SUBSYSTEM (TOTAL) TO SPECIFY DRIVES TO TEST AND NON-DEFAULT PARAMETER VALUES, FOR PASS 2. THE PROGRAM IDENTIFIES ITSELF, AS FOLLOWS:

MD-11-DZR6Q - RK06 DRIVE COMPATIBILITY PROGRAM

THE DIALOGUE FOR 220 START IS IDENTICAL TO THE DIALOGUE FOR THE 204 START DESCRIBED ABOVE (SECTION 8.2), FOR THE SELECTION OF SUBSYSTEM PARAMETERS AND THE SPECIFICATION OF ALL DRIVES TO BE TESTED ON THE VARIOUS SUBSYSTEMS. HOWEVER, AFTER THIS DIALOGUE IS COMPLETED, THE PROGRAM PROCEEDS WITH PASS 2, AND DIRECTS THE OPERATOR IN THE MOVEMENT OF THE PACK, AS DESCRIBED IN SECTION 8.5.

NOTE THAT SINCE THE APPROPRIATE PROCESSOR MUST BE STARTED AT THE STARTING ADDRESS FOR EACH SUBSYSTEM TO BE TESTED, THE COMPATIBILITY TEST MAY BE PERFORMED IN STEPS, AT VARIOUS TIMES AND BETWEEN VARIOUS DISTANT LOCATIONS, BY MOVING THE TEST PACK AND SAVING THE PRINTOUT FROM EACH PASS ON EACH POP-11 SYSTEM INVOLVED.

8.4 PASS 1 DIALOGUE

AFTER THE SELECTION OF PARAMETERS AND DRIVES HAS BEEN COMPLETED ON THE CURRENT SUBSYSTEM (SECTIONS 8.1-8.2), THE PROGRAM INDICATES THE START OF PASS 1 AS FOLLOWS:

** STARTING PASS 1 **

NEXT, THE PROGRAM SELECTS THE FIRST DRIVE TO BE TESTED ON THIS SUBSYSTEM, AND INSTRUCTS THE OPERATOR TO MOUNT THE TEST CARTRIDGE AND LOAD THE HEADS ON THAT DRIVE, AS IN THE FOLLOWING EXAMPLE:

MOUNT PACK ON DRIVE A2 AND LOAD.
TYPE R<CR> WHEN DRIVE READY:

THE OPERATOR PERFORMS THIS TASK AND TYPES R<CR> WHEN THE DRIVE READY LIGHT IS ON. THE PROGRAM PERFORMS PASS 1 FUNCTIONS ON THIS DRIVE (SEE SECTION 9.1) AND THEN INSTRUCTS THE OPERATOR TO UNLOAD THE DRIVE AND REMOVE THE PACK AS FOLLOWS:

UNLOAD DRIVE A2 AND REMOVE PACK.
TYPE R<CR> WHEN DONE:

546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601

THE OPERATOR PERFORMS THESE FUNCTIONS AND TYPES R<CR> AFTER THE PACK HAS BEEN REMOVED.

IN THE SAME MANNER, THE PROGRAM INSTRUCTS THE OPERATOR IN THE MOVEMENT OF THE PACK THROUGHOUT THE REST OF THE DRIVES ON THE CURRENT SUBSYSTEM. WHEN THIS HAS BEEN COMPLETED, THE PROGRAM DOES ONE OF THREE THINGS: (1) IF THERE IS ONLY ONE SUBSYSTEM (FROM ADR 200 START) THE PROGRAM BEGINS PASS 2 (SEE SECTION 8.5). (2) IF THERE IS ANOTHER SUBSYSTEM, THE PROGRAM DIRECTS THE OPERATOR TO PERFORM PASS 1 ON THE NEXT SUBSYS AS FOLLOWS (AND THEN HALTS) :

START AT ADR 204 FOR PASS 1 ON SUBSYS B

(3) IF THERE ARE NO MORE DRIVES TO TEST IN PASS 1 ON ANY SUBSYS, THE PROGRAM DIRECTS THE OPERATOR TO BEGIN PASS 2 ON THE FIRST SUBSYS (SEE SECT. 8.5) AS FOLLOWS (AND THEN HALTS) :

START AT ADR 220 FOR PASS 2 ON SUBSYS A

8.5 PASS 2 DIALOGUE

THE OPERATOR RETURNS TO THE FIRST SUBSYSTEM TO PERFORM PASS 2 EITHER THROUGH THE DIALOGUE OF THE ADR 200 START, OR AFTER THE SELECTION OF PARAMETERS AND DRIVES HAS BEEN COMPLETED IN ACCORDANCE WITH THE DIALOGUE OF THE ADR 220 START (SEE SECTION 8.3). IN EITHER CASE, THE PROGRAM INDICATES THE START OF PASS 2 BY TYPING :

** STARTING PASS 2 **

THE PROGRAM THEN DIRECTS THE OPERATOR IN THE UNLOADING, PACK MOVEMENT, AND LOADING OF ALL DRIVES ON THE FIRST SUBSYSTEM, IN THE SAME MANNER AS DESCRIBED FOR PASS 1 (SEE SECTION 8.4).

HOWEVER, AFTER PASS 2 TESTING (SEE SECTION 9.2) IS COMPLETED ON A GIVEN DRIVE, THE ENTIRE TEST RESULTS FOR THAT DRIVE ARE TYPED. THE DETAILS OF THIS PRINTOUT ARE DESCRIBED IN SECTION 10, AFTER THE DETAILS OF THE TESTING ARE DESCRIBED.

WHEN PASS 2 HAS BEEN COMPLETED FOR ALL DRIVES ON THE FIRST SUBSYSTEM, THE PROGRAM DOES ONE OF TWO THINGS: (1) IF THERE IS ONLY ONE SUBSYSTEM (FROM ADR 200 START) OR IF ALL DRIVES ON ALL SUBSYSTEMS HAVE BEEN TESTED IN PASS 2 (FROM ADR 220 START), THE ENTIRE TESTING AND REPORTING HAVE BEEN COMPLETED, AND THE PROGRAM TYPES:

** END OF TESTING **

602
603
604
605
606
607
608
609

(2) IF THERE IS ANOTHER SUBSYSTEM, HOWEVER, THE PROGRAM DIRECTS THE OPERATOR TO PERFORM PASS 2 ON THE NEXT SUBSYSTEM AS FOLLOWS (AND THEN HALTS):

START AT ADR 220 FOR PASS 2 ON SUBSYS B

610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665

9.0 DESCRIPTION OF TESTS

THE MAIN FUNCTIONAL BLOCKS OF CODE IN THE PROGRAM ARE ASSIGNED TEST NUMBERS, FOR THE PURPOSE OF IDENTIFICATION IN ERROR PRINTOUTS. TEST 0 REFERS TO THE OPERATOR INPUT DIALOGUE ROUTINES DESCRIBED IN SECTIONS 8.1-8.3. THE OTHER TEST NUMBERS ARE ASSIGNED BELOW, IN THE DESCRIPTION OF PASS 1 AND PASS 2 TESTING.

IN THE FOLLOWING SECTIONS, TABLES A-G ARE REFERRED TO. IN THESE TABLES, DRIVES ARE NAMED FROM 0-17 FOR ILLUSTRATIVE PURPOSES, ALTHOUGH THE DRIVES ARE NAMED THE FOLLOWING WAY IN AN ACTUAL SITUATION : A0,A1, A2,...B0,B1,B2,...C0,C1,C2,... ETC. (SEE SECTION 8.0).

9.1 DESCRIPTION OF PASS 1 TESTS

IN PASS 1, THE BASIC READ/WRITE CAPABILITY OF EACH DRIVE IS DEMONSTRATED, AND COMPATIBILITY DATA PATTERNS ARE WRITTEN BY ALL DRIVES UPON THE SAME TEST CARTRIDGE.

THE SEQUENCE OF OPERATIONS PERFORMED ON EACH DRIVE IS AS FOLLOWS:

1. TEST 1 - MOUNTING OF TEST CARTRIDGE FOR PASS 1 - THE OPERATOR MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.4).
2. TEST 2 - BASIC READ/WRITE DATA TEST - THE PROGRAM PERFORMS A WRITE AND WRITE CHECK OPERATION USING A "WORST CASE" DATA PATTERN, AT THE APPROPRIATE SECTOR FOR THIS DRIVE (SEE TABLE A) ON ALL SURFACES. THE PURPOSE OF THIS OPERATION IS TO VERIFY THE BASIC READ/WRITE CAPABILITY OF THE DRIVE ON PASS 1. THE ENTIRE SECTOR IS WRITTEN WITH THE REPETITION OF THE DATA PATTERN SHOWN IN TABLE B. THIS PATTERN CONSISTS OF A MIXTURE OF HIGH AND LOW FREQUENCY ELEMENTS AND BIT STRINGS REQUIRING VARIOUS DEGREES OF PRECOMPENSATION, WHEN ENCODED.
3. TEST 3 - WRITE OVERWRITE AND COMPATIBILITY CYLINDER BLOCKS - NEXT, THE PROGRAM WRITES ALL SECTORS FOR THIS DRIVE WITHIN THE CYLINDER BLOCKS SHOWN IN TABLE C ON ALL SURFACES USING A SINGLE REPEATED WORD OF THE PATTERN IN TABLE G. LOGICAL DRIVE 0 USES WORD 0, LOGICAL DRIVE 1 USES WORD 1, LOGICAL DRIVE 10 USES WORD 10, ETC. THUS, THE DATA FROM EACH DRIVE IS UNIQUE. TABLE C HAS BEEN DETERMINED AS FOLLOWS:

IN EACH OF THE SEVEN WRITE CURRENT ZONES ON EACH SURFACE SECTORS ARE WRITTEN WITHIN TWO DISTINCT CYLINDER BLOCKS. IN THE FIRST 16 CYLINDERS OF EACH WRITE CURRENT ZONE DATA IS WRITTEN TO BE LATER OVERWRITTEN IN PASS 2, DURING THE OVERWRITE TEST. IN THE LAST 16 CYLINDERS OF EACH CURRENT

666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705

ZONE (EXCEPT THE INNERMOST ZONE) COMPATIBILITY DATA IS WRITTEN TO BE READ WITH OFFSET IN PASS 2. WITHIN EACH CURRENT ZONE BOTH THE OVERWRITE AND COMPATIBILITY CYLINDER BLOCKS ARE IDENTICALLY WRITTEN BUT FROM ZONE TO ZONE THE DATA WRITTEN BY EACH DRIVE IS ROTATED SEVERAL SECTORS SO THAT OVER ALL THE ZONES THE DATA APPEARS AT VARIOUS ANGULAR POSITIONS ON THE PACK.

WITHIN EACH CYLINDER BLOCK, UP TO 16 SECTORS ARE WRITTEN (DEPENDING ON THE NUMBER OF DRIVES BEING TESTED) ON EACH CYLINDER. THESE SECTORS ARE ALWAYS SECTORS 0, 1, 2, 3, 5, 6, 7, 10, 12, 13, 14, 15, 17, 20, 21, 22 (OCTAL). OF THE REMAINING SECTORS 4, 11, 16, AND 23 ARE USED FOR DRIVE SELF-TESTING IN PASS 2 (SEE SECTION 9.2).

THE BASIC LAYOUT OF A TYPICAL CYLINDER BLOCK IS SHOWN IN TABLE D, WHERE THE BLOCK SHOWN IS THE COMPATIBILITY BLOCK FOR ZONE 1, WHICH STARTS ON CYLINDER 60, AND HAS THE ROTATING STARTING SECTOR = SECTOR 0. EACH NUMBER INSIDE THE BLOCK IS THE NUMBER OF THE DRIVE WHICH WRITES THAT SECTOR. TABLE D SHOWS THE BLOCKS WRITTEN BY EACH OF 16 DRIVES. IF ANY OF THE DRIVES SHOWN ARE NOT PRESENT, HOWEVER, THE BLOCKS RESERVED FOR THE MISSING DRIVES ARE SIMPLY NOT WRITTEN.

THE ABOVE PATTERN OF SECTOR WRITES INSURES THAT DATA FROM EACH DRIVE IS WRITTEN ON ADJACENT CYLINDERS TO DATA FROM EVERY OTHER DRIVE, IN BOTH DIRECTIONS. IN ADDITION, THE ROTATION OF THE ABOVE SECTORS FROM CURRENT ZONE TO CURRENT ZONE INSURES THAT OVERWRITE AND DATA COMPATIBILITY TESTING IS DONE AT SEVERAL DIFFERENT ANGULAR POSITIONS WITH RESPECT TO THE CARTRIDGE.

- 4. TEST 4 - DISMOUNTING OF TEST CARTRIDGE IN PASS 1 - THE OPERATOR UNLOADS THE DRIVE AND DISMOUNTS THE PACK, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.4), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761

9.2 DESCRIPTION OF PHASE 2 TESTS

IN PASS 2 THE ABILITY OF EACH DRIVE TO COMPLETELY OVERWRITE DATA WRITTEN BY ALL OTHER DRIVES AND TO READ DATA WRITTEN BY ALL OTHER DRIVES, IS TESTED.

THE SEQUENCE OF OPERATIONS PERFORMED BY EACH DRIVE IS AS FOLLOWS:

1. TEST 5 - MOUNTING OF TEST CARTRIDGE FOR PASS 2 - THE OPERATOR MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS, AS DIRECTED BY THE PROGRAM (SEE SECTION 8.5).
2. TEST 6 - OVERWRITE TEST - NEXT, THE PROGRAM PROCEEDS TO TEST THIS DRIVE'S OVERWRITE CAPABILITY. FIRST, THE APPROPRIATE CYLINDERS IN TABLE E FOR THIS DRIVE ARE OVERWRITTEN ON EACH SURFACE. THE DATA USED IS A REPETITION OF A SINGLE WORD OF THE PATTERN IN TABLE G. LOGICAL DRIVE 0 USES WORD 0, LOGICAL DRIVE 1 USES WORD 1, LOGICAL DRIVE 10 USES WORD 10, ETC.

THEN, EACH CYLINDER OVERWRITTEN IS READ BACK BY THIS DRIVE WITH THE FOLLOWING RANGE OF OFFSET VALUES: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200 MICRO-INCHES, IN EACH OFFSET DIRECTION (+ AND -). THE PROGRAM SCANS FOR READ ERRORS (DCK HVRC, ETC.) DURING THIS READ, AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA HAS NOT BEEN CORRECTLY OVERWRITTEN, AND A SCORE FOR THAT DRIVE IS DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE ABOVE OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT, WHILE THE CURRENT DRIVE IS PERFORMING THE OVERWRITES. FOR EACH TRACK (0, 1, 2), SCORES ARE AVERAGED OVER ALL CYLS TESTED, IN EACH OFFSET DIRECTION. AT THE COMPLETION OF THE OVERWRITE TEST ON THIS DRIVE, THE SCORES OF ALL THE DRIVES ARE CONVERTED AND STORED, FOR PRINTING AT THE END OF PASS 2 (AS DESCRIBED IN SECTION 10.2). EACH SCORE IS PROPORTIONAL TO THE DEGREE OF OFFSET WHICH COULD BE APPLIED IN A GIVEN OFFSET DIRECTION BY THE CURRENT DRIVE WHILE SUCCESSFULLY READING THE DATA IT WROTE OVER ONE OF THE OTHER DRIVE'S DATA. THUS, THE PRINTOUT REVEALS WHICH DRIVES ARE INVOLVED, IN A SITUATION IN WHICH A DRIVE CANNOT OVERWRITE ONE OR SEVERAL OTHER DRIVE'S DATA.

3. TEST 7 - DRIVE SELF-TEST - THE PROGRAM NEXT EVALUATES THE DRIVE'S ABILITY TO WRITE AND READ ITS OWN DATA, AT VARIOUS POSITIONS ON THE PACK. FIRST, SECTORS 4, 11, 16, AND 23 OF THE APPROPRIATE CYLINDERS SHOWN IN TABLE F FOR THIS DRIVE ARE WRITTEN WITH THE DATA PATTERN SHOWN IN TABLE B, FOR ALL SURFACES. THEN, THE SECTORS ARE READ WITH THE FOLLOWING OFFSETS APPLIED: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200 MICRO-INCHES, IN EACH OFFSET DIRECTION. THE PROGRAM SCANS FOR READ ERRORS DURING EACH READ, AND IT COMPUTES A SCORE WHICH IS PROPORTIONAL TO THE FAILING OFFSET

762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803

MAGNITUDE. THEN, THE SCORES FOR ALL 4 SECTORS READ IN THIS CYLINDER BLOCK ARE AVERAGED, TO COME UP WITH A DRIVE SELF-TEST SCORE FOR EACH SURFACE FOR EACH OFFSET DIRECTION. THIS SCORE IS SAVED FOR LATER USE, TO BECOME THE STANDARD FOR THE COMPATIBILITY DATA READS WHICH ARE TO FOLLOW.

4. TEST 10 - COMPATIBILITY DATA READ TEST - HAVING ESTABLISHED A SELF-TEST SCORE FOR THIS DRIVE, THE PROGRAM PROCEEDS TO PERFORM THE COMPATIBILITY DATA READS OF THE PATTERNS WRITTEN BY ALL THE DRIVES IN EACH CYLINDER BLOCK (ON EACH SURFACE). EACH COMPATIBILITY CYLINDER BLOCK SHOWN IN TABLE C IS READ, A CYLINDER AT A TIME, FOR THE FOLLOWING OFFSET VALUES: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200 MICRO-INCHES, IN EACH OFFSET DIRECTION. THE PROGRAM SCANS FOR READ ERRORS DURING EACH READ AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA WAS BEING READ AT THAT INSTANT AND A SCORE FOR THAT DRIVE IS DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE ABOVE OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT, WHILE THE CURRENT DRIVE IS READING THE COMPATIBILITY DATA. THEN, EACH SCORE IS APPROPRIATELY ADJUSTED TO REFLECT THE SELF-TEST SCORE FOR THE CURRENT DRIVE AT THAT PARTICULAR CYLINDER BLOCK. THE SCORES ARE THEN AVERAGED OVER ALL CYLINDER BLOCKS. EACH SCORE IS PROPORTIONAL TO THE CAPABILITY OF THE CURRENT DRIVE TO SUCCESSFULLY READ THE DATA WRITTEN BY ONE OF THE OTHER DRIVES, AND SCORES ARE COMPUTED SEPARATELY FOR EACH SURFACE (TRACK), FOR EACH OFFSET DIRECTION. THUS, THE PRINTOUT REVEALS WHICH DRIVES ARE INVOLVED IN A SITUATION IN WHICH A PARTICULAR DRIVE HAS DIFFICULTY IN READING THE DATA OF ONE OR SEVERAL OTHER DRIVES.
5. TEST 11 - TYPE TEST SCORES AND DISMOUNT PACK IN PASS 2 - THE OVERWRITE AND COMPATIBILITY DATA READ TEST SCORES FOR THIS DRIVE ARE CONVERTED AND TYPED. THEN, THE OPERATOR UNLOADS THE DRIVE AND DISMOUNTS THE PACK AS DIRECTED BY THE PROGRAM (SEE SECTION 8.5), TO PROCEED WITH THE ABOVE STEPS ON THE NEXT DRIVE.

804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859

10.0 PRINTOUT OF TEST RESULTS

THE TEST RESULTS ARE PRINTED AT THE END OF PASS 2 ON EACH DRIVE BEING TESTED. THESE RESULTS PERTAIN TO THE OVERWRITE TEST AND THE COMPATIBILITY DATA READ TEST.

10.1 OVERWRITE AND DRIVE COMPATIBILITY DATA TEST RESULTS

THE RESULTS OF BOTH THE OVERWRITE AND THE COMPATIBILITY DATA READ TESTS ARE ALWAYS PRINTED, REGARDLESS OF DEGREE OF SUCCESS. COMPLETE RESULTS ARE PRINTED, SEPARATELY FOR EACH DRIVE, AT THE END OF PASS 2 ON THAT DRIVE. THE TEST RESULTS ARE TABULAR IN FORM, AND CONSIST OF READ SCORES FOR THE CURRENT DRIVE, USING DATA WRITTEN BY ALL OTHER DRIVES. THE SCORES ARE CONVERTED ON THE BASIS OF 12(DEC), (WITH A PERFECT SCORE = 12). ALSO, THE SCORES ARE LISTED SEPARATELY FOR EACH TRACK (SURFACE), FOR EACH OFFSET DIRECTION, AND AN OVERALL AVERAGE FOR ALL THREE TRACKS, ON THE CURRENT DRIVE, IS PRINTED.

IN THE FOLLOWING EXAMPLE, THERE ARE 2 SYSTEMS, AND THE DRIVES BEING TESTED ARE A0,A1,A2,B0, AND B5. THE TEST RESULTS FOR DRIVE A1 ARE SHOWN BELOW:

SCORES FOR DRIVE A1 (0-12 DEC.) :

TRACK NO.	DRIVE READ	OVRWRT SCORE OFST-	OVRWRT SCORE OFST+	READ SCORE OFST-	READ SCORE OFST+
0	A0	12	12	12	12
0	SELF	12	12	12	12
0	A2	* 7	* 6	12	12
0	B0	11	12	11	11
0	B5	9	12	11	11
1	A0	12	11	12	11
1	SELF	11	10	12	12
1	A2	12	11	11	12
1	B0	12	12	9	9
1	B5	12	12	11	12
2	A0	12	12	12	12
2	SELF	12	12	12	12
2	A2	11	9	11	11
2	B0	12	12	12	12
2	B5	12	12	12	12

DRIVE A1 OVRWRT AVG = 12
 DRIVE A1 READ AVG = 10

THE ABOVE EXAMPLE REVEALS A POSSIBLE COMPATIBILITY PROBLEM BETWEEN DRIVES A1 AND A2. NOTICE THAT ON TRACK 0, THAT THE OVERWRITE SCORES

H02

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 20

SEQ 0019

860
861
862
863
864

WERE UNACCEPTABLY LOW (7 OR LESS), AND THE PROGRAM NOTED THESE BAD
SCORES WITH AN ASTERISK (*).

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
913
914
915
916
917
918

11.0 ERROR REPORTING

11.1 COMMON ERRORS

THE FOLLOWING IS A LIST OF COMMON ERROR MESSAGES WHICH ACCOMPANY ERROR TYPEOUTS FROM THE RK06 DRIVE COMPATIBILITY PROGRAM. THE ERRORS ARE SELF-EXPLANATORY.

1. UNIBUS PARITY ERROR
2. NON-EXISTANT MEMORY ERROR
3. NON-EXISTANT DRIVE ERROR
4. UNIT FIELD ERROR
5. SUBSYSTEM TIMEOUT
6. SERCON PARITY ERROR
7. DRIVE DETECTED PARITY ERROR
8. AC LOW
9. SPEED LOSS
10. ILLEGAL FUNCTION ERROR
11. PROGRAMMING ERROR
12. NON-EXECUTABLE FUNCTION ERROR
13. DRIVE TYPE ERROR
14. FORMAT ERROR
15. WRITE LOCK ERROR
16. DRIVE UNSAFE ERROR
17. SEEK INCOMPLETE ERROR
18. CYLINDER OVERFLOW ERROR
19. ILLEGAL CYLINDER ADDRESS ERROR
20. DRIVE OFF TRACK
21. DRIVE TIMING ERROR

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
970
971
972
973

- 22. DATA LATE ERROR
- 23. CONTROLLER TIMEOUT ERROR
- 24. OPERATION INCOMPLETE ERROR
- 25. HEADER VRC ERROR
- 26. DATA CHECK ERROR
- 27. WRITE CHECK ERROR
- 28. DATA MISCOMPARE
- 29. NO DRIVE RESPONSE - UFE AND NXD
- 30. DRIVE ERROR WILL NOT CLEAR
- 31. DRIVE STATUS CHANGE WILL NOT CLEAR
- 32. ATTENTION BUT NO STATUS CHANGE OR FAULT
- 33. ATTENTION BUT DRIVE NOT AVAILABLE
- 34. ERROR WHILE GATHERING DRIVE STATUS
- 35. MULTIPLE DRIVE SELECT
- 36. HEADER COMPARE ERROR
- 37. ERROR IN RECALIBRATE FOR RECOVERY
- 38. CLEAR CONTROLLER DID NOT CLEAR ERROR
- 39. NO ATTENTION IN ATTENTION SUMMARY REGISTER
- 40. UNSOLICITED ATTENTION
- 41. UNEXPECTED DATA TYPE ERROR
- 42. ATTENTION DID NOT RESET WITH CLEAR
- 43. SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION
- 44. DATA LATE WHEN UNLOADING HEADER
- 45. CONTROLLER ERROR WHEN DRIVER SERVICING
- 46. RETRY UNSUCCESSFUL
- 47. BAD SECTOR ERROR ON SECTOR NOT LISTED BAD

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

11.2 ERROR HANDLING

ERRORS REPORTED BY THE PROGRAM CONSIST OF COMMON FAILURES RESULTING FROM ATTEMPTED SUBSYSTEM FUNCTIONS, AS WELL AS CERTAIN ERRORS UNIQUE TO PARTICULAR TESTS. EACH ERROR PRINTOUT CONSISTS OF AN ERROR DESCRIPTION AND TEST NUMBER, POSSIBLY FOLLOWED BY HEADER LINES, COLUMN HEADINGS, AND COLUMNS OF REGISTER CONTENTS IN OCTAL. AS MUCH MEANINGFUL REGISTER DATA AS POSSIBLE (FOR EXAMPLE, RK611 REGISTERS) ARE REPORTED IN A GIVEN ERROR. OTHER ERROR REPORTS MAY CONSIST OF A SINGLE DESCRIPTIVE LINE.

11.3 ERROR PRINTOUT EXAMPLE

** WRITE CHECK ERROR

PREVIOUS COMMAND:

DRIVE	CMND	CYLNR	TRACK	SECTOR	WD CNT
00000	000121	000016	000001	000000	175000
HI BA	LO BA				
000000	061566				

CURRENT COMMAND:

ERR PC	DRIVE	CMND	CYLNR	TRACK	SECTOR	WD CNT
041154	000000	000131	000016	000001	000006	175000
HI BA	LO BA					
000000	061566					

PACK ADDRESS OF ERROR(S):

CYLNR	TRACK	SECTOR
000016	000001	000006

TABLE A

BASIC READ/WRITE TEST SECTORS

ADDRESS OF SECTOR ON EACH SURFACE (IN OCTAL)

1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046

DRIVE
NO.

CYLINDER

SECTOR

0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17

620
620
620
620
620
620
620
620
622
622
622
622
622
622
622
622

0
2
4
6
10
12
14
16
20
22
24
26
30
32
34
16

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

TABLE B

WORST CASE DATA PATTERN (REPEATS EVERY 16 WORDS)

<u>WORD NO.</u>	<u>DATA (OCTAL)</u>
0	072307
1	135143
2	156461
3	167230
4	073514
5	035646
6	016723
7	107351
10	143564
11	061672
12	030735
13	114356
14	046167
15	123073
16	151453
17	164616

1082
1083
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

TABLE C

CYLINDER BLOCK ASSIGNMENT FOR A GIVEN SURFACE

CURRENT ZONE - RANGE	OVERWRITE CYL BLK RANGE (OCT)	COMPATIBILITY CYL BLK RANGE (OCT)	ROTATING STARTING SECTOR
1 - CYL 0-77	CYL 0-17	CYL 60-77	SECT 0
2 - 100-177	100-117	160-177	3
3 - 200-277	200-217	260-277	7
4 - 300-377	300-317	360-377	13
5 - 400-477	400-417	460-477	17
6 - 500-577	500-517	560-577	22
7 - 600-632	600-617	---	0

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
1141
1142
1143
1144
1145
1146
1147
1148

TABLE D

BASIC CYLINDER BLOCK LAYOUT EXAMPLE

CYLINDER NUMBERS (OCTAL)	SECTOR NUMBERS (OCTAL)															
	0	1	2	3	5	6	7	10	12	13	14	15	17	20	21	22
60	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
61	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	0
62	2	3	4	5	6	7	10	11	12	13	14	15	16	17	0	1
63	3	4	5	6	7	10	11	12	13	14	15	16	17	0	1	2
64	4	5	6	7	10	11	12	13	14	15	16	17	0	1	2	3
65	5	6	7	10	11	12	13	14	15	16	17	0	1	2	3	4
66	6	7	10	11	12	13	14	15	16	17	0	1	2	3	4	5
67	7	10	11	12	13	14	15	16	17	0	1	2	3	4	5	6
70	10	11	12	13	14	15	16	17	0	1	2	3	4	5	6	7
71	11	12	13	14	15	16	17	0	1	2	3	4	5	6	7	10
72	12	13	14	15	16	17	0	1	2	3	4	5	6	7	10	11
73	13	14	15	16	17	0	1	2	3	4	5	6	7	10	11	12
74	14	15	16	17	0	1	2	3	4	5	6	7	10	11	12	13
75	15	16	17	0	1	2	3	4	5	6	7	10	11	12	13	14
76	16	17	0	1	2	3	4	5	6	7	10	11	12	13	14	15
77	17	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16

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

TABLE E

OVERWRITE CYLINDERS

DRIVE #	CYLINDERS OVERWRITTEN (OCTAL)
0	0, 100, 200, 300, 400, 500, 600
1	1, 101, 201, 301, 401, 501, 601
2	2, 102, 202, 302, 402, 502, 602
3	3, 103, 203, 303, 403, 503, 603
4	4, 104, 204, 304, 404, 504, 604
5	5, 105, 205, 305, 405, 505, 605
6	6, 106, 206, 306, 406, 506, 606
7	7, 107, 207, 307, 407, 507, 607
10	10, 110, 210, 310, 410, 510, 610
11	11, 111, 211, 311, 411, 511, 611
12	12, 112, 212, 312, 412, 512, 612
13	13, 113, 213, 313, 413, 513, 613
14	14, 114, 214, 314, 414, 514, 614
15	15, 115, 215, 315, 415, 515, 615
16	16, 116, 216, 316, 416, 516, 616
17	17, 117, 217, 317, 417, 517, 617

1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218

TABLE F

SELF-TEST CYLINDERS

DRIVE #	CYLINDERS (WHERE SECTORS 4,11,16,23 ARE TESTED)
0	60, 160, 260, 360, 460, 560
1	61, 161, 261, 361, 461, 561
2	62, 162, 262, 362, 462, 562
3	63, 163, 263, 363, 463, 563
4	64, 164, 264, 364, 464, 564
5	65, 165, 265, 365, 465, 565
6	66, 166, 266, 366, 466, 566
7	67, 167, 267, 367, 467, 567
8	68, 168, 268, 368, 468, 568
9	69, 169, 269, 369, 469, 569
10	70, 170, 270, 370, 470, 570
11	71, 171, 271, 371, 471, 571
12	72, 172, 272, 372, 472, 572
13	73, 173, 273, 373, 473, 573
14	74, 174, 274, 374, 474, 574
15	75, 175, 275, 375, 475, 575
16	76, 176, 276, 376, 476, 576
17	77, 177, 277, 377, 477, 577

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

TABLE G

PSEUDO-RANDOM DATA PATTERN

WORD #	DATA (OCTAL)
-----	-----
0	040135
1	177070
2	070414
3	064531
4	174473
5	062422
6	114352
7	036620
10	010031
11	052336
12	017310
13	011347
14	102367
15	152567
16	001246
17	160073

2

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

000001

160000

163000

001100

000011

000012

000015

.NLIST MC,MD,CND
.LIST ME
.ENABL ABS,AMA

```
*****  
SBTTL STARTING ADDRESSES  
*  
*      200  DEFAULT PARAMETERS AND RUN PASS 1 AND PASS 2  
*      204  SELECT PARAMETERS AND RUN PASS 1  
*      220  SELECT PARAMETERS AND RUN PASS 2  
*      224  MEMORY DUMP ROUTINE  
*****
```

```
$TN=1  
.TITLE MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM  
*COPYRIGHT (C) 1976  
*DIGITAL EQUIPMENT CORP.  
*MAYNARD, MASS. 01754  
*  
*PROGRAM BY DAVE HOFFMAN  
*  
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
*  
$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
```

```
    $SWR=163000  
.SBTTL OPERATIONAL SWITCH SETTINGS  
*  
*      SWITCH          USE  
*      -----  
*      15             HALT ON ERROR  
*      14             LOOP ON TEST  
*      13             INHIBIT ERROR TYPEOUTS  
*      12             REPORT DESCRIPTION ONLY, ON ERRORS  
*      10             BELL ON ERROR  
*      9              LOOP ON ERROR  
*      8              APPLY RANDOM STALL BETWEEN OPERATIONS  
*      7              TYPE BAD SECTOR FILES (BSF'S) AT START
```

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

```
*MISCELLANEOUS DEFINITIONS  
HT= 11 ;;CODE FOR HORIZONTAL TAB  
LF= 12 ;;CODE FOR LINE FEED  
CR= 15 ;;CODE FOR CARRIAGE RETURN
```

BASIC DEFINITIONS

```

1311      000200      CRLF= 200          ;; CODE FOR CARRIAGE RETURN-LINE FEED
1312      177776      PS= 177776          ;; PROCESSOR STATUS WORD
1313      .EQUIV PS,PSW
1314      177774      STKLMT= 177774      ;; STACK LIMIT REGISTER
1315      177772      PIRQ= 177772       ;; PROGRAM INTERRUPT REQUEST REGISTER
1316      177570      DSWR= 177570       ;; HARDWARE SWITCH REGISTER
1317      177570      DDISP= 177570      ;; HARDWARE DISPLAY REGISTER
1318
1319      . *GENERAL PURPOSE REGISTER DEFINITIONS
1320      000000      R0= %0              ;; GENERAL REGISTER
1321      000001      R1= %1              ;; GENERAL REGISTER
1322      000002      R2= %2              ;; GENERAL REGISTER
1323      000003      R3= %3              ;; GENERAL REGISTER
1324      000004      R4= %4              ;; GENERAL REGISTER
1325      000005      R5= %5              ;; GENERAL REGISTER
1326      000006      R6= %6              ;; GENERAL REGISTER
1327      000007      R7= %7              ;; GENERAL REGISTER
1328      000006      SP= %6              ;; STACK POINTER
1329      000007      PC= %7              ;; PROGRAM COUNTER
1330
1331      . *PRIORITY LEVEL DEFINITIONS
1332      000000      PR0= 0                ;; PRIORITY LEVEL 0
1333      000040      PR1= 40              ;; PRIORITY LEVEL 1
1334      000100      PR2= 100            ;; PRIORITY LEVEL 2
1335      000140      PR3= 140            ;; PRIORITY LEVEL 3
1336      000200      PR4= 200            ;; PRIORITY LEVEL 4
1337      000240      PR5= 240            ;; PRIORITY LEVEL 5
1338      000300      PR6= 300            ;; PRIORITY LEVEL 6
1339      000340      PR7= 340            ;; PRIORITY LEVEL 7
1340
1341      . *"SWITCH REGISTER" SWITCH DEFINITIONS
1342      100000      SW15= 100000
1343      040000      SW14= 40000
1344      020000      SW13= 20000
1345      010000      SW12= 10000
1346      004000      SW11= 4000
1347      002000      SW10= 2000
1348      001000      SW09= 1000
1349      000400      SW08= 400
1350      000200      SW07= 200
1351      000100      SW06= 100
1352      000040      SW05= 40
1353      000020      SW04= 20
1354      000010      SW03= 10
1355      000004      SW02= 4
1356      000002      SW01= 2
1357      000001      SW00= 1
1358      .EQUIV SW09,SW9
1359      .EQUIV SW08,SW8
1360      .EQUIV SW07,SW7
1361      .EQUIV SW06,SW6
1362      .EQUIV SW05,SW5
1363      .EQUIV SW04,SW4
1364      .EQUIV SW03,SW3
1365      .EQUIV SW02,SW2
1366      .EQUIV SW01,SW1

```

BASIC DEFINITIONS

1367
1368
1369
1370 100000
1371 040000
1372 020000
1373 010000
1374 004000
1375 002000
1376 001000
1377 000400
1378 000200
1379 000100
1380 000040
1381 000020
1382 000010
1383 000004
1384 000002
1385 000001
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398 000004
1399 000010
1400 000014
1401 000014
1402 000014
1403 000020
1404 000024
1405 000030
1406 000034
1407 000060
1408 000064
1409 000240
1410
1411
1412
1413 000000
1414
1415
1416
1417 000174
1418 000174 000000
1419 000176 000000
1420
1421 000200 000137 011746
1422 000204

```

.EQUIV SW00,SW0
; *DATA BIT DEFINITIONS (BIT00 TO BIT15)
BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7
.EQUIV BIT06,BIT6
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

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

.SBTTL TRAP CATCHER
.=0
; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0 ; ; SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ; ; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#DFSTR ; ; JUMP TO STARTING ADDRESS OF PROGRAM
.=204

```

1423 000204 000137 011754
 1424 000220 000137 011772
 1425 000224 000700
 1426 000226 001100
 1427
 1428
 1429
 1430
 1431
 1432
 1433
 1434 000230
 1435 000046
 1436 000046 016712
 1437 000052
 1438 000052 140000
 1439 000230
 1440 001000
 1441
 1442
 1443
 1444
 1445
 1446 001000
 1447 000024
 1448 000024 000200
 1449 000044
 1450 000044 001000
 1451 001000
 1452
 1453
 1454
 1455
 1456 001000
 1457 001000 000000
 1458 001002 001330
 1459 001004 001320
 1460 001006 001546
 1461 001010 001546
 1462 001012 000030
 1463 120210
 1464 177440
 1465 000377

```

JMP @#P1STRT
      =220
JMP @#P2STRT

..LOW: .WORD 700
..HIGH: .WORD 1100

.SBTTL ACT11 HOOKS

;*****
;HOOKS REQUIRED BY ACT11
      $SVPC=      ;SAVE PC
      =46
      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
      =52
      .WORD 140000 ;;2)SET LOC.52 TO 140000
      .= $SVPC    ;; RESTORE PC
      =1000

.SBTTL APT PARAMETER BLOCK

;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
      .SX=      ;;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.

$APTHD:
$SHIBTS: .WORD 0 ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBAADR: .WORD $MAIL ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$STSM: .WORD 1320 ;; RUN TIM OF LONGEST TEST
$SPASTM: .WORD 1546 ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$SUNITH: .WORD 1546 ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD $ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

AVECT1=120210 ;RKVEC=210, RKPRI=5
ABASE=177440 ;RYBAS ADRS
AOEVM=000377 ;SET DEVICES 0-7 IN MAP

```

COMMON TAGS

.SBTTL COMMON TAGS

1466
1467
1468
1469
1470
1471
1472
1473 001100
1474 001100 000000
1475 001102 000
1476 001103 000
1477 001104 000000
1478 001106 000000
1479 001110 000000
1480 001112 000000
1481 001114 000
1482 001115 001
1483 001116 000000
1484 001120 000000
1485 001122 000000
1486 001124 000000
1487 001126 000000
1488 001130 000000
1489 001132 000000
1490 001134 000
1491 001135 000
1492 001136 000000
1493 001140 177570
1494 001142 177570
1495 001144 177560
1496 001146 177562
1497 001150 177564
1498 001152 177566
1499 001154 000
1500 001155 002
1501 001156 012
1502 001157 000
1503 001160 000000
1504
1505 001162 000000
1506 001164 000000
1507 001166 000000
1508 001170 000000
1509 001172 000000
1510 001174 000000
1511 001176 000000
1512 001200 000000
1513 001202 000000
1514 001204 000000
1515 001206 000000
1516 001210 000000
1517 001212 000000
1518 001214 000000
1519 001216 000000
1520 001220 000000
1521 001222 000000

```

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

```

.=1100

```

$CMTAG:      .WORD      0
              .BYTE      0
$STNM:       .BYTE      0
$ERFLG:      .BYTE      0
$ICNT:       .WORD      0
$LPADR:      .WORD      0
$LPERR:      .WORD      0
$ERTTL:      .WORD      0
$ITEMB:      .BYTE      0
$ERMAX:      .BYTE      1
$ERRPC:      .WORD      0
$GDADR:      .WORD      0
$BDADR:      .WORD      0
$GD0AT:      .WORD      0
$BD0AT:      .WORD      0
              .WORD      0
              .WORD      0
$AUTOB:      .BYTE      0
$INTAG:      .BYTE      0
              .WORD      0
$SWR:        .WORD      DSWR
DISPLAY:     .WORD      DDISP
$TKS:        177560
$TKB:        177562
$TPS:        177564
$TPB:        177566
$NULL:       .BYTE      0
$FILLS:      .BYTE      2
$FILLC:      .BYTE      12
$TPFLG:      .BYTE      0
$REGAD:      .WORD      0
              .WORD      0
$REG0:       .WORD      7
$REG1:       .WORD      0
$REG2:       .WORD      0
$REG3:       .WORD      0
$REG4:       .WORD      0
$REG5:       .WORD      0
$REG6:       .WORD      0
$REG7:       .WORD      0
$REG10:      .WORD      0
$REG11:      .WORD      0
$REG12:      .WORD      0
$REG13:      .WORD      0
$REG14:      .WORD      0
$REG15:      .WORD      0
$REG16:      .WORD      0
$REG17:      .WORD      0
$REG20:      .WORD      0

```

```

;; START OF COMMON TAGS
;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL BYTE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED

;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR

;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
;; CONTAINS THE ADDRESS FROM WHICH ($REG0) WAS OBTAINED
;; CONTAINS (( $REGAD)+0)
;; CONTAINS (( $REGAD)+2)
;; CONTAINS (( $REGAD)+4)
;; CONTAINS (( $REGAD)+6)
;; CONTAINS (( $REGAD)+10)
;; CONTAINS (( $REGAD)+12)
;; CONTAINS (( $REGAD)+14)
;; CONTAINS (( $REGAD)+16)
;; CONTAINS (( $REGAD)+20)
;; CONTAINS (( $REGAD)+22)
;; CONTAINS (( $REGAD)+24)
;; CONTAINS (( $REGAD)+26)
;; CONTAINS (( $REGAD)+30)
;; CONTAINS (( $REGAD)+32)
;; CONTAINS (( $REGAD)+34)
;; CONTAINS (( $REGAD)+36)
;; CONTAINS (( $REGAD)+40)

```

COMMON TAGS

1522 001224 000000
 1523 001226 000000
 1524 001230 000000
 1525 001232 000000
 1526 001234 000000
 1527 001236 000000
 1528 001240 000000
 1529 001242 000000
 1530 001244 000000
 1531 001246 000000
 1532 001250 000000
 1533 001252 000000
 1534 001254 000000
 1535 001256 000000
 1536 001260 000000
 1537 001262 000000
 1538 001264 000000
 1539 001266 000000
 1540 001270 000000
 1541 001272 000000
 1542 001274 000000
 1543 001276 000000
 1544 001300 000000
 1545 001302 000000
 1546 001304 000000
 1547 001306 000000
 1548 001310 000000
 1549 001312 000000
 1550 001314 000000
 1551 001316 000000
 1552 001320 177607
 1553 001324 077
 1554 001328 015
 1555 001326 000012
 1556
 1557
 1558
 1559
 1560
 1561 001330
 1562 001330 000000
 1563 001332 000000
 1564 001334 000000
 1565 001336 000000
 1566 001340 000000
 1567 001342 000000
 1568 001344 000000
 1569 001346 000000
 1570 001350
 1571 001350 000
 1572 001351 000
 1573 001352 000000
 1574 001354 000000
 1575 001356 000000
 1576
 1577

000377

\$REG2: .WORD 0
 \$REG3: .WORD 0
 \$REG23: .WORD 0
 \$REG24: .WORD 0
 \$REG25: .WORD 0
 \$REG26: .WORD 0
 \$REG27: .WORD 0
 \$REG30: .WORD 0
 \$REG31: .WORD 0
 \$REG32: .WORD 0
 \$REG33: .WORD 0
 \$REG34: .WORD 0
 \$REG35: .WORD 0
 \$REG36: .WORD 0
 \$REG37: .WORD 0
 \$THP0: .WORD 0
 \$THP1: .WORD 0
 \$THP2: .WORD 0
 \$THP3: .WORD 0
 \$THP4: .WORD 0
 \$THP5: .WORD 0
 \$THP6: .WORD 0
 \$THP7: .WORD 0
 \$THP10: .WORD 0
 \$THP11: .WORD 0
 \$THP12: .WORD 0
 \$THP13: .WORD 0
 \$THP14: .WORD 0
 \$THP15: .WORD 0
 \$ESCAPE: 0
 \$BELL: .ASCIZ <20><377><377>
 \$QUES: .ASCII /?
 \$CARL: .ASCII <15>
 \$LF: .ASCIZ <12>

 \$BTTL APT MAILBOX-ETABLE

 .EVEN
 \$MAIL: .WORD AMSGTY : APT MAILBOX
 \$MSGTY: .WORD AMSGTY : MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL : FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN : TEST NUMBER
 \$PASS: .WORD APASS : PASS COUNT
 \$DEVCT: .WORD ADEVCT : DEVICE COUNT
 \$UNIT: .WORD AUNIT : I/O UNIT NUMBER
 \$MSGAD: .WORD AMSGAD : MESSAGE ADDRESS
 \$MSGLG: .WORD AMSGLG : MESSAGE LENGTH
 \$ETABLE: .WORD APT ENVIRONMENT TABLE
 \$ENV: .BYTE AENV : ENVIRONMENT BYTE
 \$ENVM: .BYTE AENVM : ENVIRONMENT MODE BITS
 \$SWREG: .WORD ASWREG : APT SWITCH REGISTER
 \$USWR: .WORD AUSWR : USER SWITCHES
 \$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

```

1578      *
1579      *
1580      *
1581      *
1582 001360      000      $MAMS1: .BYTE      AMAMS1
1583 001361      000      $MAMP1: .BYTE      AMAMP1
1584      *
1585      *
1586      *
1587      *
1588 001362      000000    $MAADR1: .WORD      AMAADR1
1589      *
1590 001364      000      $MAMS2: .BYTE      AMAMS2
1591 001365      000      $MAMP2: .BYTE      AMAMP2
1592 001366      000000    $MAADR2: .WORD      AMAADR2
1593 001370      000      $MAMS3: .BYTE      AMAMS3
1594 001371      000      $MAMP3: .BYTE      AMAMP3
1595 001372      000000    $MAADR3: .WORD      AMAADR3
1596 001374      000      $MAMS4: .BYTE      AMAMS4
1597 001375      000      $MAMP4: .BYTE      AMAMP4
1598 001376      000000    $MAADR4: .WORD      AMAADR4
1599 001400      120210    $VECT1: .WORD      AVECT1
1600 001402      000000    $VECT2: .WORD      AVECT2
1601 001404      177440    $BASE: .WORD      ABASE
1602 001406      000377    $DEVN: .WORD      ADEVN
1603 001410
1604      *

```

11/70=06, P00=07, Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT
;; HIGH ADDRESS, M.S. BYTE
;; MEM. TYPE, BLK#1
MEM. TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
300 NSEC BIPOLAR=002
500 NSEC MOS=003
;; HIGH ADDRESS, BLK#1
MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
;; HIGH ADDRESS, M.S. BYTE
;; MEM. TYPE, BLK#2
MEM. LAST ADDRESS, BLK#2
;; HIGH ADDRESS, M.S. BYTE
;; MEM. TYPE, BLK#3
MEM. LAST ADDRESS, BLK#3
;; HIGH ADDRESS, M.S. BYTE
;; MEM. TYPE, BLK#4
MEM. LAST ADDRESS, BLK#4
;; INTERRUPT VECTOR#1, BUS PRIORITY#1
;; INTERRUPT VECTOR#2, BUS PRIORITY#2
;; BASE ADDRESS OF EQUIPMENT UNDER TEST
;; DEVICE MAP

ERROR POINTER TABLE

.SBTTL ERROR POINTER TABLE

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

```

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

SERRTB:

1605					
1606					
1607					
1608					
1609					
1610					
1611					
1612					
1613					
1614					
1615					
1616					
1617					
1618					
1619	001410				
1620					
1621					
1622	001410	044553			
1623	001412	047646			
1624	001414	051432			
1625	001416	051542			
1626					
1627					
1628	001420	044577			
1629	001422	047646			
1630	001424	051432			
1631	001426	051542			
1632					
1633					
1634	001430	044623			
1635	001432	047646			
1636	001434	051432			
1637	001436	051542			
1638					
1639					
1640	001440	044646			
1641	001442	047646			
1642	001444	051432			
1643	001446	051542			
1644					
1645					
1646	001450	044667			
1647	001452	047646			
1648	001454	051432			
1649	001456	051572			
1650					
1651					
1652	001460	044706			
1653	001462	047646			
1654	001464	051432			
1655	001466	051626			
1656					
1657					
1658	001470	044732			
1659	001472	047646			
1660	001474	051432			

```

;ERROR 1 ;UNIBUS PARITY ERROR
EM1
DH100
DT100
DF01

;ERROR 2 ;NON-EXISTANT MEMORY
EM2
DH100
DT100
DF01

;ERROR 3 ;NON-EXISTANT DRIVE
EM3
DH100
DT100
DF01

;ERROR 4 ;UNIT FIELD ERROR
EM4
DH100
DT100
DF01

;ERROR 5 ;SUBSYSTEM TIMEOUT
EM5
DH100
DT100
DF02

;ERROR 6 ;D TO C PARITY ERROR
EM6
DH100
DT100
DF03

;ERROR 7 ;DRIVE DETECTED PARITY ERROR
EM7
DH100
DT100
  
```


ERROR POINTER TABLE

1661	001476	051626	DF03	
1662				
1663			:ERROR 10	
1664	001500	044766	EM10	;AC LOW
1665	001502	047646	DH100	
1666	001504	051432	DT100	
1667	001506	051572	DF02	
1668				
1669			:ERROR 11	
1670	001510	044775	EM11	;SPEED LOSS
1671	001512	047646	DH100	
1672	001514	051432	DT100	
1673	001516	051572	DF02	
1674				
1675			:ERROR 12	
1676	001520	045010	EM12	;ILLEGAL FUNCTION
1677	001522	047646	DH100	
1678	001524	051432	DT100	
1679	001526	051572	DF02	
1680				
1681			:ERROR 13	
1682	001530	045031	EM13	;PROGRAMMING ERROR
1683	001532	047646	DH100	
1684	001534	051432	DT100	
1685	001536	051542	DF01	
1686				
1687			:ERROR 14	;NON-EXISTANT FUNCTION
1688	001540	045053	EM14	
1689	001542	047646	DH100	
1690	001544	051432	DT100	
1691	001546	051572	DF02	
1692				
1693			:ERROR 15	
1694	001550	045101	EM15	;DRIVE TYPE ERROR
1695	001552	047646	DH100	
1696	001554	051432	DT100	
1697	001556	051572	DF02	
1698				
1699			:ERROR 16	
1700	001560	045122	EM16	;FORMAT ERROR
1701	001562	047646	DH100	
1702	001564	051432	DT100	
1703	001566	051572	DF02	
1704				
1705			:ERROR 17	
1706	001570	045137	EM17	;WRITE LOCK ERROR
1707	001572	047646	DH100	
1708	001574	051432	DT100	
1709	001576	051572	DF02	
1710				
1711			:ERROR 20	
1712	001600	045160	EM20	;DRIVE UNSAFE
1713	001602	047646	DH100	
1714	001604	051432	DT100	
1715	001606	051572	DF02	
1716				

ERROR POINTER TABLE

1717			.ERROR 21	
1718	001610	045175	EM21	;SEEK INCOMPLETE
1719	001612	047646	DH100	
1720	001614	051432	DT100	
1721	001616	051572	DF02	
1722				
1723			.ERROR 22	
1724	001620	045215	EM22	;CYLINDER OVERFLOW
1725	001622	047646	DH100	
1726	001624	051432	DT100	
1727	001626	051572	DF02	
1728				
1729			.ERROR 23	
1730	001630	045237	EM23	;ILLEGAL CYLINDER
1731	001632	047646	DH100	
1732	001634	051432	DT100	
1733	001636	051572	DF02	
1734				
1735			.ERROR 24	
1736	001640	045270	EM24	;DRIVE OFF TRACK
1737	001642	047646	DH100	
1738	001644	051432	DT100	
1739	001646	051572	DF02	
1740				
1741			.ERROR 25	
1742	001650	045310	EM25	;DRIVE TIMING ERROR
1743	001652	047646	DH100	
1744	001654	051432	DT100	
1745	001656	051572	DF02	
1746				
1747			.ERROR 26	
1748	001660	045333	EM26	;DATA LATE
1749	001662	047646	DH100	
1750	001664	051432	DT100	
1751	001666	051572	DF02	
1752				
1753			.ERROR 27	
1754	001670	045345	EM27	;CONTROLLER TIMEOUT
1755	001672	047646	DH100	
1756	001674	051432	DT100	
1757	001676	051572	DF02	
1758				
1759			.ERROR 30	
1760	001700	045370	EM30	;OPERATION INCOMPLETE
1761	001702	047646	DH100	
1762	001704	051432	DT100	
1763	001706	051702	DF05	
1764				
1765			.ERROR 31	
1766	001710	045415	EM31	;HEADER VRC ERROR
1767	001712	047646	DH100	
1768	001714	051432	DT100	
1769	001716	051702	DF05	
1770				
1771			.ERROR 32	
1772	001720	045436	EM32	;DATA CHECK ERROR

ERROR POINTER TABLE

1773	001722	047646	DH100	
1774	001724	051432	DT100	
1775	001726	051736	DF07	
1776				
1777			:ERROR 33	
1778	001730	045457	EM33	;WRITE CHECK ERROR
1779	001732	047646	DH100	
1780	001734	051432	DT100	
1781	001736	051772	DF10	
1782				
1783			:ERROR 34	
1784	001740	045501	EM34	;DATA MISCOMPARE(S)
1785	001742	047646	DH100	
1786	001744	051432	DT100	
1787	001746	051666	DF04	
1788				
1789			:ERROR 35	
1790	001750	045521	EM35	;NO DRIVE RESPONSE-UFE & NXD
1791	001752	047646	DH100	
1792	001754	051432	DT100	
1793	001756	051542	DF01	
1794				
1795			:ERROR 36	
1796	001760	045555	EM36	;DRIVE ERROR WILL NOT CLEAR
1797	001762	000000	0	
1798	001764	000000	0	
1799	001766	000000	0	
1800				
1801			:ERROR 37	;DRIVE STATUS CHANGE WILL NOT CLEAR
1802	001770	045610	EM37	
1803	001772	000000	0	
1804	001774	000000	0	
1805	001776	000000	0	
1806				
1807			:ERROR 40	
1808	002000	045653	EM40	;ATTENTION BUT NO STATUS CHANGE OR FAULT
1809	002002	047646	DH100	
1810	002004	051432	DT100	
1811	002006	051572	DF02	
1812				
1813			:ERROR 41	
1814	002010	045717	EM41	;ATTENTION BUT DRIVE NOT AVAILABLE
1815	002012	047646	DH100	
1816	002014	051432	DT100	
1817	002016	051572	DF02	
1818				
1819			:ERROR 42	
1820	002020	045755	EM42	;ATTENTION WHEN NOT EXPECTED
1821	002022	047646	DH100	
1822	002024	051432	DT100	
1823	002026	051572	DF02	
1824				
1825			:ERROR 43	;ERROR WHILE GATHERING DRIVE STATUS
1826	002030	046005	EM43	
1827	002032	047646	DH100	
1828	002034	051432	DT100	

ERROR POINTER TABLE

1829	002036	052036	DF12	
1830				
1831			.ERROR 44	
1832	002040	046262	EM63	;CLEAR CONTROLLER DID NOT CLEAR ERROR
1833	002042	047646	DH100	
1834	002044	051432	DT100	
1835	002046	052036	DF12	
1836				
1837			.ERROR 45	
1838	002050	046327	EM64	;NO ATTENTION IN ATTENTION SUMMARY REG
1839	002052	047646	DH100	
1840	002054	051432	DT100	
1841	002056	052036	DF12	
1842				
1843			.ERROR 46	
1844	002060	046372	EM65	;UNSOLICITED ATTENTION
1845	002062	047646	DH100	
1846	002064	051432	DT100	
1847	002066	052036	DF12	
1848				
1849			.ERROR 47	
1850	002070	046420	EM66	;UNEXPECTED DATA TYPE ERROR
1851	002072	047646	DH100	
1852	002074	051432	DT100	
1853	002076	052036	DF12	
1854				
1855			.ERROR 50	
1856	002100	046453	EM67	;ATTENTION DID NOT RESET WITH CLEAR
1857	002102	047646	DH100	
1858	002104	051432	DT100	
1859	002106	052036	DF12	
1860				
1861			.ERROR 51	
1862	002110	046512	EM70	;SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION
1863	002112	047646	DH100	
1864	002114	051432	DT100	
1865	002116	052036	DF12	
1866				
1867			.ERROR 52	
1868	002120	046042	EM52	;MULTIPLE DRIVE SELECT
1869	002122	047646	DH100	
1870	002124	051432	DT100	
1871	002126	052036	DF12	
1872				
1873			.ERROR 53	
1874	002130	046070	EM53	;ABREVIATED HCE ERROR
1875	002132	047646	DH100	
1876	002134	051432	DT100	
1877	002136	052066	DF13	
1878				
1879			.ERROR 54	
1880	002140	045370	EM30	;OPERATION INCOMPLETE ERROR
1881	002142	047646	DH100	
1882	002144	051432	DT100	
1883	002146	052116	DF14	
1884				

ERROR POINTER TABLE

1885			.ERROR 55	
1886	002150	045415	EM31	;ABREVIATED HVRC ERROR
1887	002152	047646	DH100	
1888	002154	051432	DT100	
1889	002156	052066	DF13	
1890				
1891			.ERROR 56	
1892	002160	046115	EM56	;2 TIMEOUT ERROR
1893	002162	047646	DH100	
1894	002164	051432	DT100	
1895	002166	052146	DF15	
1896				
1897			.ERROR 57	;2ND LEVEL IN SUBSYSTEM TIMEOUT
1898	002170	046115	EM56	
1899	002172	047646	DH100	
1900	002174	051432	DT100	
1901	002176	052212	DF16	
1902				
1903			.ERROR 60	
1904	002200	046134	EM60	;ERROR IN RECAL FOR RECOVERY
1905	002202	000000	0	
1906	002204	000000	0	
1907	002206	000000	0	
1908				
1909			.ERROR 61	
1910	002210	046170	EM61	;ABORT MESSAGE
1911	002212	000000	0	
1912	002214	000000	0	
1913	002216	000000	0	
1914				
1915			.ERROR 62	
1916	002220	046236	EM62	;CYLINDER MISCOMPARE
1917	002222	047646	DH100	
1918	002224	051432	DT100	
1919	002226	052256	DF17	
1920				
1921			.ERROR 63	;DATA ERROR WORDS
1922	002230	000000	0	
1923	002232	000000	0	
1924	002234	051524	DT602	
1925	002236	052366	DF25	
1926				
1927			.ERROR 64	
1928	002240	046262	EM63	;CLEAR CONTROLLER DID NOT CLEAR ERROR
1929	002242	047646	DH100	
1930	002244	051432	DT100	
1931	002246	051572	DF02	
1932				
1933			.ERROR 65	
1934	002250	046327	EM64	;NO ATTENTION IN ATTENTION SUMMARY REG
1935	002252	047646	DH100	
1936	002254	051432	DT100	
1937	002256	051572	DF02	
1938				
1939			.ERROR 66	
1940	002260	046372	EM65	;UNSOLICITED ATTENTION

ERROR POINTER TABLE

1941	002262	047646	DH100	
1942	002264	051432	DT100	
1943	002266	051572	DF02	
1944				
1945			.ERROR 67	
1946	002270	046420	EM66	;UNEXPECTED DATA TYPE ERROR
1947	002272	047646	DH100	
1948	002274	051432	DT100	
1949	002276	051572	DF02	
1950				
1951			.ERROR 70	
1952	002300	046453	EM67	;ATTENTION DID NOT RESET WITH CLEAR
1953	002302	047646	DH100	
1954	002304	051432	DT100	
1955	002306	051572	DF02	
1956				
1957			.ERROR 71	
1958	002310	046512	EM70	;SUBSYSTEM CLEAR DID NOT CLEAR ATT
1959	002312	047646	DH100	
1960	002314	051432	DT100	
1961	002316	051572	DF02	
1962				
1963			.ERROR 72	
1964	002320	046561	EM71	;DATA LATE WHEN UNLOADING HEADER
1965	002322	047646	DH100	
1966	002324	051432	DT100	
1967	002326	051572	DF02	
1968				
1969			.ERROR 73	
1970	002330	046621	EM72	;CONTROLLER ERROR DURING DRIVER SERVICE
1971	002332	047646	DH100	
1972	002334	051432	DT100	
1973	002336	051572	DF02	
1974				
1975			.ERROR 74	
1976	002340	046670	EM73	;DRIVE DETECTED PARITY ERROR
1977	002342	047646	DH100	
1978	002344	051432	DT100	
1979	002346	051572	DF02	
1980				
1981			.ERROR 75	
1982	002350	046724	EM74	;UNDEFINED ERROR
1983	002352	047646	DH100	
1984	002354	051432	DT100	
1985	002356	051572	DF02	
1986				
1987			.ERROR 76	
1988	002360	046744	EM75	;MARKING SECTOR BAD MESSAGE
1989	002362	000000	0	
1990	002364	000000	0	
1991	002366	000000	0	
1992				
1993			.ERROR 77	
1994	002370	046774	EM76	;BAD DATA VERIFICATION WITH READ
1995	002372	050633	DH605	
1996	002374	051516	DT601	

ERROR POINTER TABLE

1997	002376	052312	DF21	
1998				
1999			:ERROR 100	
2000	002400	047056	EM77	;RETRY SUCCESSFUL MESSAGE
2001	002402	000000	0	
2002	002404	051516	DT601	
2003	002406	052352	DF23	
2004				
2005			:ERROR 101	
2006	002410	047056	EM77	;ANOTHER RETRY SUCCESSFUL MESSAGE
2007	002412	000000	0	
2008	002414	051516	DT601	
2009	002416	052352	DF23	
2010				
2011			:ERROR 102	
2012	002420	047077	EM100	;RETRY UNSUCCESSFUL MESSAGE
2013	002422	000000	0	
2014	002424	051516	DT601	
2015	002426	052352	DF23	
2016				
2017			:ERROR 103	
2018	002430	047122	EM101	;NO VALID HEADERS IN TRACK JUST READ
2019	002432	050615	DH6042	
2020	002434	051516	DT601	
2021	002436	052362	DF24	
2022				
2023			:ERROR 104	
2024	002440	047200	EM102	;BSE ERROR ON SECTOR NOT LISTED AS BAD
2025	002442	047646	DH100	
2026	002444	051432	DT100	
2027	002446	051572	DF02	
2028				
2029			:ERROR 105	
2030	002450	047252	EM103	;TIMED-OUT ON READ HEADER
2031	002452	047646	DH100	
2032	002454	051432	DT100	
2033	002456	051626	DF03	
2034				
2035			:ERROR 106	
2036	002460	047300	EM104	;TIMED-OUT ON SEEK
2037	002462	047646	DH100	
2038	002464	051432	DT100	
2039	002466	051626	DF03	
2040				
2041			:ERROR 107	
2042	002470	047322	EM105	;DRIVE SIEZED BY OTHER PORT
2043	002472	047646	DH100	
2044	002474	051432	DT100	
2045	002476	051572	DF02	
2046				
2047			:ERROR 110	
2048	002500	047355	EM106	; "DATA MISCMPR WHILE BAI SET"
2049	002502	047646	DH100	
2050	002504	051432	DT100	
2051	002506	052402	DF27	
2052				

ERROR POINTER TABLE

2053			:ERROR 111	
2054	002510	047410	EM107	; "NO MEM WHEN EXPECTED"
2055	002512	000000	0	
2056	002514	000000	0	
2057	002516	000000	0	
2058			:ERROR 112	
2059			EM110	; "INTRPT WHEN CNTRLR NOT READY"
2060	002520	047455	DH100	
2061	002522	047646	DT100	
2062	002524	051432	DF01	
2063	002526	051542		
2064			:ERROR 113	
2065			EM111	; "NO ATT'N ON SEEK"
2066	002530	047510	DH100	
2067	002532	047646	DT100	
2068	002534	051432	DF02	
2069	002536	051572		
2070			:ERROR 114	
2071			EM112	; DRIVE'S CYLINDER INCORRECT
2072	002540	047531	DH702	
2073	002542	051042	DT202	
2074	002544	051472	DF26	
2075	002546	052372		
2076			:ERROR 115	
2077			0	; TYPE ADRS OF DATA MISCOMPARE(S)
2078	002550	000000	0	
2079	002552	000000	DT601	
2080	002554	051516	DF11	
2081	002556	052016		
2082			:ERROR 116	
2083			EM34	; DATA MISCOMPARE (11/70)
2084	002560	045501	DH103	
2085	002562	050157	0	
2086	002564	000000	DF20	
2087	002566	052302		
2088			:ERROR 117	
2089			0	; PART OF DATA MISCOMPARE
2090	002570	000000	0	
2091	002572	000000	DT100	
2092	002574	051432	DF22	
2093	002576	052322		
2094			:ERROR 120	
2095			EM113	; ABORT- CAN'T READ BSF
2096	002600	047564	DH100	
2097	002602	047646	DT100	
2098	002604	051432	DF01	
2099	002606	051542		
2100			:ERROR 121	
2101			EM114	; KT11 FAILURE
2102	002610	047612	DH100	
2103	002612	047646	DT100	
2104	002614	051432	DF30	
2105	002616	052426		
2106			:ERROR 122	
2107			EM115	; MEM PARITY ERROR
2108	002620	047627		

ERROR POINTER TABLE

2109	002622	047646	OH100
2110	002624	051432	OT100
2111	002626	052452	OF31
2112			
2113			
2114			
2115			
2116			
2117			
2118			
2119			
2120			
2121			
2122			
2123			
2124			
2125			
2126			
2127			
2128			
2129			
2130			
2131			

.SBTTL BIT ASSIGNMENTS IN THE RK611 REGISTERS

2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185

RKCS1							
15	14	13	12	11	10	9	8
CERR	DI	DCPAR	CFMT	CTO	CDT	BA17	BA16
CCLR							
R/W	RO	RO	R/W	RO	R/W	R/W	R/W
7	6	5	4	3	2	1	0
RDY	IE	SPARE	F4	F3	F2	F1	GO
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RKCS2							
15	14	13	12	11	10	9	8
DLT	WCE	UPE	NED	NEM	PGE	MDS	UFE
RO	RO	RO	RO	RO	RO	RO	RO
7	6	5	4	3	2	1	0
OR	IR	CLR	BAI	DESL	DS2	DS1	DS0
RO	RO	WO	R/W	R/W	R/W	R/W	R/W
RKDC (DESIRED CYLINDER)							
15	14	13	12	11	10	9	8
NU	NU	NU	NU	NU	NU	DC9	DC8
READ ONLY							
7	6	5	4	3	2	1	0
DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
READ ONLY							
RKDA (DESIRED TRACK AND SECTOR)							
15	14	13	12	11	10	9	8
NU	UN	UN	UN	UN	TA2	TA1	TA0
READ ONLY							
7	6	5	4	3	2	1	0
UN	UN	UN	SA4	SA3	SA2	SA1	SA0
READ ONLY							

K04

2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236

RKDB (DATA BUFFER)							
15	14	13	12	11	10	9	8
DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8
READ/WRITE							
7	6	5	4	3	2	1	0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
READ/WRITE							
RKASOF (ATTENTION SUMMARY AND OFFSET)							
15	14	13	12	11	10	9	8
ATN7	ATN6	ATN5	ATN4	ATN3	ATN2	ATN1	ATN0
READ ONLY							
7	6	5	4	3	2	1	0
UN	OF6	OF5	OF4	OF3	OF2	OF1	OF0
READ/WRITE							
RKWC (WORD COUNT)							
15	14	13	12	11	10	9	8
WC15	WC14	WC13	WC12	WC11	WC10	WC9	WC8
READ/WRITE							
7	6	5	4	3	2	1	0
WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0
READ/WRITE							
RKBA (BUS ADDRESS)							
15	14	13	12	11	10	9	8
BA15	BA14	BA13	BA12	BA11	BA10	BA9	BA8
READ/WRITE							
7	6	5	4	3	2	1	0
BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
READ/WRITE							
							ALWYSO

2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275

RKER (ERROR REGISTER)							
15	14	13	12	11	10	9	8
DCK	UNS	OPI	DTE	WLE	IDAE	COE	HVRC
READ ONLY							
7	6	5	4	3	2	1	0
BSE	ECH	DTYPE	FMT	DRPAR	ILF	SKI	ILC
READ ONLY							
RKDS (DRIVE STATUS)							
15	14	13	12	11	10	9	8
SVAL	DSC	PIP	SPON	WRL	UN	UN	DTP
READ ONLY							
7	6	5	4	3	2	1	0
DRDY	VV	DROT	SPLS	ACLO	OFFSET	UN	DRA
READ ONLY							
RKMR1 (MAINTENANCE REG 1)							
15	14	13	12	11	10	9	8
RD	WRT	ECCW	PCD	PCA	MEMD	MERD	MCLK
GATE	GATE	READ ONLY				READ/WRITE	
7	6	5	4	3	2	1	0
MIND	MSP	DMD	PAT	MS3	MS2	MS1	MS0
READ/WRITE							

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

;RKECC/PAT
15 14 13 12 11 10 9 8

UN ! UN ! UN ! UN ! UN ! EPA10! EPA9 ! EPA8 !
READ ONLY

7 6 5 4 3 2 1 0

EPA7 ! EPA6 ! EPA5 ! EPA4 ! EPA3 ! EPA2 ! EPA1 ! EPA0 !
READ ONLY

;RKECC/POS
15 14 13 12 11 10 9 8

UN ! UN ! UN ! EP012! EP011! EP010! EP09 ! EP08 !
READ ONLY

7 6 5 4 3 2 1 0

EP07 ! EP06 ! EP05 ! EP04 ! EP03 ! EP02 ! EP01 ! EP00 !
READ ONLY

;DRIVE STATUS INFORMATION

;LINE A MESSAGE 00
15 14 13 12 11 10 9 8

PAR ! DSC ! PIP ! SPON ! WALK ! OFFON! FMT ! DRTP !

7 6 5 4 3 2 1 0

DRDY ! VV ! DRA ! NU ! NU ! DRIVE SELECT CODE !

2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351

LINE B MESSAGE 00							
15	14	13	12	11	10	9	8
PAR	UNS	DROT	DCLO	MLE	SKI	PERR	ILF

7	6	5	4	3	2	1	0
FLT	ACLO	IVDA	UN	UN	UN	0	0

LINE A MESSAGE 01							
15	14	13	12	11	10	9	8
PAR	UNLDG	RTZ	LDG	REV	FWD	SPEED	CART
	HDS		HDS			OK	PRES

7	6	5	4	3	2	1	0
DOOR	BRUSH	HEADS	TRK	UN	DRIVE SELECT CODE		
LTCH	HOME	HOME	FOLOW				
			OK				

LINE B MESSAGE 01							
15	14	13	12	11	10	9	8
PAR	SERVO	LIMIT	SEEK	PLO	DIBIT	INDEX	MULTI
	BRAKE	ON SK	NO MO	ERR	ERR	ERR	HD SEL

7	6	5	4	3	2	1	0
HEAD	WR GTE	WR CUR	SEC	NU	NU	0	1
FAULT	AND NO	AND NO	ERR				
	XISTON	WR GTE					

2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407

```

:LINE A MESSAGE 10
 15  14  13  12  11  10  9  8
-----
PAR !  NU !  NU !  CYLINDER DIFF/OFFSET VALUE  !
-----
 7  6  5  4  3  2  1  0
-----
CYLINDER DIFF/OFFSET VALUE!  NU !  DRIVE SELECT CODE !
-----
:LINE B MESSAGE 10
 15  14  13  12  11  10  9  8
-----
PAR !  ALIGN!  NU !  CYLINDER ADDRESS  !
  !  SIGN !
-----
 7  6  5  4  3  2  1  0
-----
CYLINDER ADDRESS  !  UN !  UN !  1 !  0 !
-----
:LINE A MESSAGE 11
 15  14  13  12  11  10  9  8
-----
PAR !  DRIVE SERIAL NUMBER  !
-----
 7  6  5  4  3  2  1  0
-----
DRIVE SERIAL NUMBER  !  DRIVE SELECT CODE !
-----
:LINE B MESSAGE 11
 15  14  13  12  11  10  9  8
-----
PAR !  NU !  NU !  DECODED HEAD !  SECTOR!
  !  !  !  ADDRESS !  COUNT !
-----
 7  6  5  4  3  2  1  0
-----
SECTOR COUNT  !  UN !  UN !  1 !  1 !
-----

```

.SBTTL RK06 CONTROLLER REGISTER DEFINITION

000000
000002
000004
000006
000010

RKCS1= 0
RKWC= 2
RKBA= 4
RKDA= 6
RKCS2= 10

:CONTROL AND STATUS REGISTER 1
:WORD COUNT REGISTER
:BUS ADDRESS REGISTER
:DESIRED TRACK SECTOR REGISTER
:CONTROL AND STATUS REGISTER 2

000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063

000012
000014
000016
000020
000020
000024
000026
000034
000036
000030
000030
000032
000032

000101
000103
000105
000107
000111
000113
000115
000117
000121
000123
000125
000127
000131

000140
000141
000164
000176
000177
000300

000001
000005

000001
000100
000200
000400
001000
002000
004000
010000

RKDS= 12
RKER= 14
RKASOF= 16
RK C= 20
RK CYL= 20
RKDB= 24
RKMR1= 26
RKMR2= 34
RKMR3= 36
RKPOS= 30
RKECPS= 30
RKPAT= 32
RKECPT= 32

.SBTTL DRIVE COMMANDS

SELDRV= 101
PACK= 103
CLEAR= 105
UNLOAD= 107
SATSPL= 111
RECAL= 113
OFFSET= 115
SEEK= 117
RDATA= 121
WDATA= 123
RHEAD= 125
WHEAD= 127
WRTCHK= 131

;
;

RELEAS= 140
RDSTAT= 141
RDALHD= 164
CONCLR= 176
SUBCLR= 177
INTR= 300

;
DR.SEL= 001
DR.CLR= 005

.SBTTL CONTROL AND STATUS REGISTER 1 BITS

GO= BIT0
IE= BIT6
RDY= BIT7
BA16= BIT8
BA17= BIT9
CDT= BIT10
CTO= BIT11

CFMT= BIT12

;DRIVE STATUS REGISTER
;ERROR REGISTER
;ATTENTION SUMMARY AND OFFSET REGISTER
;DESIRED CYLINDER REGISTER
;DESIRED CYLINDER REGISTER
;DATA BUFFER
;MAINTENANCE REGISTER 1
;MAINTENANCE REGISTER 2
;MAINTENANCE REGISTER 3
;ECC POSITION INFORMATION
;ECC POSITION INFORMATION
;ECC PATTERN INFORMATION
;ECC PATTERN INFORMATION

;SELECT DRIVE
;PACK ACKNOWLEDGE
;DRIVE CLEAR
;UNLOAD
;START SPINDLE
;RECALIBRATE
;OFFSET
;SEEK
;READ DATA
;WRITE DATA
;READ HEADER
;WRITE HEADER AND DATA
;WRITE CHECK

; THE FOLLOWING ARE NOT DRIVE COMMANDS BUT ARE USED BY THE DRIVER
; TO SIMULATE A SPECIFIC DESIRED OPERATION

;RELEASE DRIVE
;GET ALL STATUS FROM DRIVE
;READ ALL HEADERS
;CONTROLLER CLEAR (BIT 15 OF CS1)
;SUBSYSTEM CLEAR (BIT 5 OF CS2)
;GENERATE INTERRUPT TO CPU

;DRIVER ISSUED SERVICE COMMANDS
;DRIVE SELECT
;DRIVE CLEAR

;GO BIT
;INTERRUPT ENABLE
;CONTROLLER READY
;BUS ADDRESS BIT 16
;BUS ADDRESS BIT 17
;CONTROLLER DRIVE TYPE (0=RK06)
;CONTROLLER TIMED OUT WAITING FOR
;DRIVE RESPONSE
;CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)

CONTROL AND STATUS REGISTER 1 BITS

2564	020000	SPAR=	BIT13	; DRIVE BUS PARITY ERROR DETECTED BY CONTROLLER
2565	040000	DI=	BIT14	; DRIVE INTERRUPT
2566	100000	CERR=	BIT15	; CONTROLLER ERROR
2567	100000	CCLR=	BIT15	; CONTROLLER CLEAR
2568				
2569				
2570				
2571				
2572	000001	B.BA16=	BIT0	; BUS ADDRESS BIT 16
2573	000002	B.BA17=	BIT1	; BUS ADDRESS BIT 17
2574	000004	B.CDT=	BIT2	; CONTROLLER DRIVE TYPE (0=RK06)
2575	000020	B.CFMT=	BIT4	; CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)

; THESE BIT DEFINITIONS ARE USED FOR ADDRESS
 ; THE HIGH BYTE OF RKCS1

.SBTTL CONTROL AND STATUS REGISTER 2 BITS

2576				
2577				
2578				
2579	000007	DRVMSK=	7	; MASK FOR DRIVE SELECTION CODE
2580	000010	DESL=	BIT3	; DESELECT OR RELEASE DRIVE IN BITS 0-2
2581	000010	RLS=	BIT3	; DESELECT OR RELEASE DRIVE IN BITS 0-2
2582	000020	BAI=	BIT4	; BUS ADDRESS INCREMENT INHIBIT
2583	000040	CLR=	BIT5	; CLEAR CONTROLLER AND ALL DRIVES
2584	000040	SCLR=	BIT5	; CLEAR CONTROLLER AND ALL DRIVES
2585	000100	IR=	BIT6	; INPUT READY
2586	000200	OR=	BIT7	; OUTPUT READY
2587	000400	UFE=	BIT8	; UNIT FIELD ERROR
2588	001000	MDS=	BIT9	; MULTIPLE DRIVE SELECT
2589	002000	PGE=	BIT10	; PROGRAMMING ERROR
2590	004000	NEM=	BIT11	; NON-EXISTENT MEMORY
2591	010000	NED=	BIT12	; NON-EXISTENT DRIVE
2592	020000	UPE=	BIT13	; UNIBUS PARITY ERROR
2593	040000	WCE=	BIT14	; WRITE CHECK ERROR
2594	100000	DLT=	BIT15	; DATA LATE ERROR

.SBTTL ERROR REGISTER BIT DEFINITION

2595				
2596				
2597				
2598	000001	ILC=	BIT0	; ILLEGAL FUNCTION CODE
2599		*ILF=	BIT0	; ILLEGAL FUNCTION CODE
2500				
2501	000002	SKI=	BIT1	; SEEK INCOMPLETE
2502	000004	ILF=	BIT2	; ILLEGAL DRIVE FUNCTION
2503	000004	NXF=	BIT2	; ILLEGAL DRIVE FUNCTION
2504	000010	DRP=	BIT3	; DRIVE DETECTED DRIVE BUS PARITY ERROR
2505	000020	FMT=	BIT4	; FORMAT ERROR
2506	000040	DYE=	BIT5	; DRIVE TYPE ERROR
2507	000100	ECH=	BIT6	; ECC HARD
2508	000200	BSE=	BIT7	; BAD SECTOR ERROR
2509	000400	HCRC=	BIT8	; HEADER CRC ERROR
2510	000400	HVRC=	BIT8	; HEADER VRC ERROR
2511	001000	COE=	BIT9	; CYLINDER ADDRESS OVERFLOW ERROR
2512	002000	IDA=	BIT10	; INVALID DISK ADDRESS ERROR
2513	004000	WLE=	BIT11	; WRITE LOCK ERROR
2514	010000	DTE=	BIT12	; DRIVE TIMING ERROR
2515	020000	OPI=	BIT13	; OPERATION (SEARCH) INCOMPLETE
2516	040000	UNS=	BIT14	; DRIVE UNSAFE
2517	100000	DCK=	BIT15	; DATA CHECK

.SBTTL STATUS REGISTER BIT DEFINITION

2518
 2519

```

2520      000001      DRA=   BIT0      ;DRIVE AVAILABLE (CONTROLLER IS SET IF
2521                                     ; THIS BIT IS RESET)
2522      000004      OFST=  BIT2      ;DRIVE OFFSET
2523      000010      ACLO=  BIT3      ;AC LOW
2524      000020      SPDLS= BIT4      ;SPEED LOSS
2525      000020      DCLO=  BIT4      ;DC LOW
2526      000040      DROT=  BIT5      ;DRIVE OFF TRACK
2527      000100      VV=    BIT6      ;VOLUME VALID
2528      000200      DRY=   BIT7      ;DRIVE READY
2529      000200      DRDY=  BIT7      ;DRIVE READY
2530      000400      DOT=   BIT8      ;DRIVE TYPE (0=RK06)
2531      004000      WRL=   BIT11     ;WRITE LOCK
2532      020000      PIP=   BIT13     ;POSITIONING IN PROGRESS
2533      040000      DSC=   BIT14     ;DRIVE STATUS CHANGE
2534      100000      SVAL=  BIT15     ;STATUS VALID
2535
2536      .SBTTL  MAINTENANCE REGISTER 1 BIT DEFINITION
2537
2538      000017      MESMSK= 17      ;MESSAGE MASK
2539
2540      000020      PAT=   BIT4      ;FORCE EVEN PARITY ON DRIVE BUS MESSAGE LINES
2541      000040      DMD=   BIT5      ;DIAGNOSTIC MODE
2542      000100      MSP=   BIT6      ;MAINTENANCE SECTOR PULSE
2543      000200      MIND=  BIT7      ;MAINTENANCE INDEX
2544      000400      MCLK=  BIT8      ;MAINTENANCE CLOCK
2545      001000      MERD=  BIT9      ;MAINTENANCE ENCODED READ DATA
2546      002000      MEWD=  BIT10     ;MAINTENANCE ENCODED WRITE DATA
2547      004000      PCA=   BIT11     ;PRECOMPENSATION ADVANCE
2548      010000      PCD=   BIT12     ;PRECOMPENSATION DELAY
2549      020000      ECCW=  BIT13     ;ECC WORD IS BEING READ OR WRITTEN
2550      040000      WRTGAT= BIT14     ;WRITE GATE
2551      100000      RDGATE= BIT15     ;READ GATE
2552
2553      .SBTTL  DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE A
2554
2555      000040      S.DRA=  BIT5      ;DRIVE AVAILIABLE
2556      000100      S.VV=   BIT6      ;VOLUME VALID
2557      000200      S.DRY=  BIT7      ;DRIVE READY
2558      000400      S.TYPE= BIT8      ;DRIVE TYPE
2559      001000      S.FORM= BIT9      ;DRIVE FORMAT
2560      002000      S.OFF=  BIT10     ;OFFSET
2561      004000      S.WRL=  BIT11     ;WRITE LOCK
2562      010000      S.SPIN= BIT12     ;SPINDLE ON
2563      020000      S.PIP=  BIT13     ;POSITIONING IN PROGRESS
2564      040000      S.DSC=  BIT14     ;DRIVE STATUS CHANGE
2565
2566      .SBTTL  DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE B
2567
2568      000040      S.ICYL= BIT5      ;ILLEGAL CYLINDER ADDRESS
2569      000100      S.ACLO= BIT6      ;AC LOW
2570      000200      S.FLT=  BIT7      ;DRIVE FAULT
2571      000400      S.ILF=  BIT8      ;ILLEGAL FUNCTION
2572      001000      S.PAR=  BIT9      ;DRIVE DETECTED DRIVE BUS PARITY ERROR
2573      002000      S.SKI=  BIT10     ;SEEK INCOMPLETE
2574      004000      S.WLE=  BIT11     ;WRITE LOCK ERROR
2575      010000      S.SPLS= BIT12     ;SPEED LOSS
  
```

2576 010000
 2577 020000
 2578 040000
 2579
 2580
 2581
 2582 000020
 2583 000040
 2584 000100
 2585 000200
 2586 000400
 2587 001000
 2588 002000
 2589 004000
 2590 010000
 2591 020000
 2592 040000
 2593
 2594
 2595
 2596 000020
 2597 000040
 2598 000100
 2599 000200
 2600 000400
 2601 001000
 2602 002000
 2603 004000
 2604 010000
 2605 020000
 2606 040000
 2607
 2608
 2609
 2610 000007
 2611 100000
 2612 000003
 2613 017760
 2614 017760
 2615 077770
 2616 000760
 2617 007000

S.DCLO= BIT12 ;DC LOW
 S.DROT= BIT13 ;DRIVE OFF TRACK
 S.UNS= BIT14 ;DRIVE UNSAFE

.SBTTL DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE A

S.XDOK= BIT4 ;TRANSDUCER OK
 S.HOHM= BIT5 ;HEADS HOME
 S.BRHM= BIT6 ;BRUSHES HOME
 S.DOOR= BIT7 ;DOOR INTERLOCKED
 S.CART= BIT8 ;CARTRIDGE INTERLOCK
 S.SPOK= BIT9 ;SPEED OK
 S.FWD= BIT10 ;FORWARD
 S.REV= BIT11 ;REVERSE
 S.LOAD= BIT12 ;HEADS LOADING
 S.RTZ= BIT13 ;RETURN TO ZERO
 S.UNLD= BIT14 ;HEADS UNLOADING

.SBTTL DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE B

S.SECT= BIT4 ;SECTOR ERROR
 S.WCLK= BIT5 ;WRITE CLOCK AND NO WRITE GATE
 S.WGAT= BIT6 ;WRITE GATE AND NO TRANSISTIONS
 S.HOFL= BIT7 ;HEAD FAULT
 S.MHD= BIT8 ;MULTIPLE HEAD SELECT
 S.XERR= BIT9 ;INDEX ERROR
 S.DIB= BIT10 ;DIBIT ERROR
 S.PLO= BIT11 ;PLO ERROR
 S.NMOV= BIT12 ;SEEK AND NO MOTION
 S.LIMD= BIT13 ;LIMIT DETECT ON SEEK
 S.BRKE= BIT14 ;SERVO-BRAKE

.SBTTL COMMON MASKS

M.DRV= 7 ;DRIVE CODE
 M.PAR= BIT15 ;PARITY
 M.ID= 3 ;BYTE ID
 M.COIF= 17760 ;CYLINDER DIFFERENCE/OFFSET
 M.CADD= 17760 ;CYLINDER ADDRESS
 M.SER= 77770 ;DRIVE SERIAL NUMBER
 M.SECT= 760 ;SECTOR COUNT
 M.HEAD= 7000 ;HEAD DECODE

2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673

.SBTTL PARAMETER BLOCK ALLOCATION

```

*****
* 1 : COMMAND ! DRIVE NO.
* 3 : CYLINDER ADDRESS
* 5 : TRACK ! SECTOR
* 7 : BA16-17, FORMAT, DRV TYPE! OFFSET
* 11 : BUS ADDRESS (LOW 16 BITS)
* 13 : WORD COUNT (2'S COMPLEMENT)
* 15 : PROGRAM DRIVE STATUS INFORMATION
* 17 : COMMAND AND STATUS REGISTER 1
* 21 : COMMAND AND STATUS REGISTER 2
* 23 : WORD COUNT REGISTER
* 25 : BUS ADDRESS REGISTER
* 27 : DESIRED TRACK AND SECTOR
* 31 : DESIRED CYLINDER
* 33 : ATTENTION SUMMARY AND DRIVE OFFSET
* 35 : ERROR REGISTER
* 37 : STATUS REGISTER
* 41 : MESSAGE LINE A STATUS BYTE 00
* 43 : MESSAGE LINE B STATUS BYTE 00
* 45 : MESSAGE LINE A STATUS BYTE 01
* 47 : MESSAGE LINE B STATUS BYTE 01
* 51 : MESSAGE LINE A STATUS BYTE 10
* 53 : MESSAGE LINE B STATUS BYTE 10
* 55 : MESSAGE LINE A STATUS BYTE 11
* 57 : MESSAGE LINE B STATUS BYTE 11
* 61 : ECC POSITION INFORMATION
* 63 : ECC PATTERN INFORMATION
*****

```

0
4
6
10
12
14
16
20
22
24
26
28
30
32
34
36
40
42
44
46
50
52
54
56
60
62

.SBTTL PARAMETERS PASSED TO THE DRIVER

THE FOLLOWING DEFINITIONS ARE USED TO PASS PARAMETERS TO THE RK06 DRIVER

```

000000 P.DRVN= 0 ;DRIVE NUMBER
000001 P.CMND= 1 ;COMMAND
000002 P.CYLN= 2 ;CYLINDER ADDRESS
000004 P.SECT= 4 ;SECTOR
000005 P.TRCK= 5 ;TRACK
000006 P.OFST= 6 ;OFFSET
000007 P.CSIH= 7 ;RKCSI BITS 8-15
000007 P.BA16= 7 ;BUS ADDRESS (BITS 16 AND 17)
000010 P.BA10= 10 ;BUS ADDRESS (BITS 0-15)
000012 P.WC= 12 ;WORD COUNT (2'S COMPLEMENT)
000014 P.PRST= 14 ;PROGRAM DRIVE STATUS INFORMATION

```

.SBTTL PROGRAM DEVICE STATUS REGISTER DEFINITION

```

000001 DRVUSE= BIT0 ;DRIVE IN USE
000002 DRVPOS= BIT1 ;DRIVE POSITIONING
000004 DRVPOT= BIT2 ;DRIVE POSITIONED FOR DATA TRANSFER
000010 UEXATT= BIT3 ;UNEXPECTED ATTENTION
000020 DRVHRD= BIT4 ;DRIVE HAS HARD ERROR
000040 DRVDSC= BITS ;DRIVE STATUS CHANGE DID NOT CLEAR

```

```

2674      000100      CMDTO= BIT6      : NO TERMINATION TO COMMAND FOR AT
2675      :           : LEAST 1 SECOND
2676      000200      W.WCK= BIT7      : WRITE FOR WRITE WRITE CHECK
2677      000400      NOCHK= BIT8      : NO CHECK, DO NOT SET INTERRUPT ENABLE
2678      001000      PBSVAL= BIT9      : PARAMETER STATUS WORDS VALID
2679      :           : (SET WHEN ERROR TERMINATION OR
2680      :           : READ STATUS COMMAND)
2681      002000      DRPDRV= BIT10      : DROP DRIVE FROM TEST SEQUENCE
2682      004000      MODSC= BIT11      : ATTENTION SET BUT DCS AND FAULT RESET
2683      010000      DRVSZD= BIT12      : DRIVE SEIZED BY OTHER PORT
2684      020000      E.UNLD= BIT13      : DRIVE UNLOADED DUE TO ERROR
2685      040000      Q.INIT= BIT14      : PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
2686      100000      DTBAII= BIT15      : INHIBIT BUS ADDRESS INCREMENT
2687
2688      .SBTTL  PARAMETERS PASSED FROM DRIVER TO PROGRAM
2689
2690      ;           THE FOLLOWING DEFINITIONS ARE USED FOR REGISTER RETURNS
2691      ;           FROM THE DRIVER TO THE CALLING PROGRAM
2692
2693      000016      P.CS1= 16      : COMMAND AND STATUS REGISTER 1
2694      000020      P.CS2= 20      : COMMAND AND STATUS REGISTER 2
2695      000022      P.WCR= 22      : WORD COUNT REGISTER
2696      000024      P.BAR= 24      : BUS ADDRESS REGISTER
2697      000026      P.DTS= 26      : DESIRED TRACK SECTOR REGISTER
2698      000030      P.DCYL= 30      : DESIRED CYLINDER REGISTER
2699      000032      P.ASOF= 32      : ATTENTION SUMMARY/OFFSET REGISTER
2700      000034      P.ER= 34      : ERROR REGISTER
2701      000036      P.OS= 36      : STATUS REGISTER
2702      000040      P.A00= 40      : MESSAGE A STATUS BYTE 00
2703      000042      P.B00= 42      : MESSAGE B STATUS BYTE 00
2704      000044      P.A01= 44      : MESSAGE A STATUS BYTE 01
2705      000046      P.B01= 46      : MESSAGE B STATUS BYTE 01
2706      000050      P.A10= 50      : MESSAGE A STATUS BYTE 10
2707      000052      P.B10= 52      : MESSAGE B STATUS BYTE 10
2708      000054      P.A11= 54      : MESSAGE A STATUS BYTE 11
2709      000056      P.B11= 56      : MESSAGE B STATUS BYTE 11
2710      000060      P.EPOS= 60      : ECC POSITION INFORMATION
2711      000062      P.EPAT= 62      : ECC PATTERN INFORMATION
2712
2713      .SBTTL  PARAMETER BLOCK 0 FOR DRIVE
2714
2715      002630      000      PARMO: .BYTE 0      : DRIVE NUMBER
2716      002631      000      : .BYTE 0      : COMMAND
2717      002632      000000      : .WORD 0      : CYLINDER ADDRESS
2718      002634      000      : .BYTE 0      : SECTOR ADDRESS
2719      002635      000      : .BYTE 0      : TRACK ADDRESS
2720      002636      000      : .BYTE 0      : OFFSET VALUE
2721      002637      000      : .BYTE 0      : BUS ADDRESS (BITS 16 AND 17)
2722      002640      000000      : .WORD 0      : BUS ADDRESS (BITS 0 - 15)
2723      002642      000000      : .WORD 0      : WORD COUNT (2'S COMPLEMENT)
2724      002644      000000      : .WORD 0      : PROGRAM DRIVE STATUS INFORMATION
2725      002646      000000      : .WORD 0      : COMMAND AND STATUS REGISTER 1
2726      002650      000000      : .WORD 0      : COMMAND AND STATUS REGISTER 2
2727      002652      000000      : .WORD 0      : WORD COUNT REGISTER
2728      002654      000000      : .WORD 0      : BUS ADDRESS REGISTER
2729      002656      000000      : .WORD 0      : DESIRED TRACK AND SECTOR REGISTER

```

2730	002660	000000	.WORD	0	: DESIRED CYLINDER REGISTER
2731	002662	000000	.WORD	0	: ATTENTION SUMMARY/OFFSET REGISTER
2732	002664	000000	.WORD	0	: ERROR REGISTER
2733	002666	000000	.WORD	0	: STATUS REGISTER
2734	002670	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 00
2735	002672	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 00
2736	002674	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 01
2737	002676	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 01
2738	002700	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 10
2739	002702	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 10
2740	002704	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 11
2741	002706	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 11
2742	002710	000000	.WORD	0	: ECC POSITION INFORMATION
2743	002712	000000	.WORD	0	: ECC PATTERN INFORMATION

.SBTTL PARAMETER BLOCK 1 FOR DRIVE

2744					
2745					
2746					
2747	002714	000	PARM1: .BYTE	0	: DRIVE NUMBER
2748	002715	000	.BYTE	0	: COMMAND
2749	002716	000000	.WORD	0	: CYLINDER ADDRESS
2750	002720	000	.BYTE	0	: SECTOR ADDRESS
2751	002721	000	.BYTE	0	: TRACK ADDRESS
2752	002722	000	.BYTE	0	: OFFSET VALUE
2753	002723	000	.BYTE	0	: BUS ADDRESS (BITS 16 AND 17)
2754	002724	000000	.WORD	0	: BUS ADDRESS (BITS 0 - 15)
2755	002726	000000	.WORD	0	: WORD COUNT (2'S COMPLEMENT)
2756	002730	000000	.WORD	0	: PROGRAM DRIVE STATUS INFORMATION
2757	002732	000000	.WORD	0	: COMMAND AND STATUS REGISTER 1
2758	002734	000000	.WORD	0	: COMMAND AND STATUS REGISTER 2
2759	002736	000000	.WORD	0	: WORD COUNT REGISTER
2760	002740	000000	.WORD	0	: BUS ADDRESS REGISTER
2761	002742	000000	.WORD	0	: DESIRED TRACK AND SECTOR REGISTER
2762	002744	000000	.WORD	0	: DESIRED CYLINDER REGISTER
2763	002746	000000	.WORD	0	: ATTENTION SUMMARY/OFFSET REGISTER
2764	002750	000000	.WORD	0	: ERROR REGISTER
2765	002752	000000	.WORD	0	: STATUS REGISTER
2766	002754	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 00
2767	002756	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 00
2768	002760	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 01
2769	002762	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 01
2770	002764	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 10
2771	002766	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 10
2772	002770	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 11
2773	002772	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 11
2774	002774	000000	.WORD	0	: ECC POSITION INFORMATION
2775	002776	000000	.WORD	0	: ECC PATTERN INFORMATION

.SBTTL TEMPORARY CONTROLLER REGISTER STORAGE

2776					
2777					
2778					
2779	003000	000000	T.CS1: .WORD	0	: TEMPORARY STORAGE FOR COMMAND AND STATUS REGISTER 1
2780					
2781	003002	000000	T.CS2: .WORD	0	: TEMPORARY STORAGE FOR COMMAND AND STATUS REGISTER 2
2782					
2783	003004	000000	T.WCR: .WORD	0	: TEMPORARY STORAGE FOR WORD COUNT REGISTER
2784	003006	000000	T.BA: .WORD	0	: TEMPORARY STORAGE FOR BUS ADDRESS REGISTER
2785	003010	000000	T.DA: .WORD	0	: TEMPORARY STORAGE FOR DISK TRACK AND SECTOR

TEMPORARY CONTROLLER REGISTER STORAGE

2786	003012	000000	T.DC:	.WORD	0	TEMPORARY STORAGE FOR DRIVE CYLINDER
2787	003014	000000	T.ASOF:	.WORD	0	TEMPORARY STORAGE FOR ATTENTION SUMMARY AND OFFSET
2788						
2789	003016	000000	T.ER:	.WORD	0	TEMPORARY STORAGE FOR ERROR REGISTER
2790	003020	000000	T.DS:	.WORD	0	TEMPORARY STORAGE FOR DRIVE STATUS REGISTER
2791	003022	000000	T.MR1:	.WORD	0	TEMPORARY STORAGE FOR MAINTENANCE REGISTER 1
2792	003024	000000	T.MR2:	.WORD	0	TEMPORARY STORAGE FOR MAINTENANCE REGISTER 2
2793	003026	000000	T.MR3:	.WORD	0	TEMPORARY STORAGE FOR MAINTENANCE REGISTER 3
2794	003030	000000	T.POS:	.WORD	0	TEMPORARY STORAGE FOR ECC POSITION
2795	003032	000000	T.PAT:	.WORD	0	TEMPORARY STORAGE FOR ECC PATTERN
2796	003034	000000	T.DB:	.WORD	0	TEMPORARY STORAGE FOR DATA BUFFER REGISTER
2797						
2798			.SBTTL	DRIVER	PARAMETERS	
2799						
2800	003036	177440	RKBAS:	.WORD	177440	ADDRESS OF RK611 UNIBUS ADDRESS BLOCK
2801	003040	000210	RKVEC:	.WORD	210	ADDRESS OF R611 VECTOR
2802	003042	000240	RKPRI:	.WORD	PR5	RK611 INTERRUPT PRIORITY
2803	003044	027426	A.NORM:	ERRFRE		ADDRESS OF NORMAL RETURN FROM DRIVER
2804	003046	030640	A.ABNL:	ERRHDL		ADDRESS OF ABNORMAL RETURN FROM DRIVER
2805	003050	030134	A.CONT:	CONERR		ADDRESS OF CONTROLLER ERROR RETURN
2806	003052	000000	E.CONT:	.WORD	0	CONTROLLER ERROR STATUS
2807						THIS LOCATION IS CLEARED WHEN EVERY COMMAND IS INITIATED. IF A CONTROLLED ERROR OCCURS THE FOLLOWING BIT ASSIGNMENT IS USED:
2808						
2809						
2810						
2811						
2812		000001	E.CCLR=	BIT0		CLEAR CONTROLLER DID NOT CLEAR ERROR
2813		000002	E.NOAT=	BIT1		NO ATTENTION IN ATTENTION SUMMARY REG
2814		000004	E.UATT=	BIT2		UNSOLICITED ATTENTION (SEQUENTIAL ONLY)
2815		000010	E.UDAT=	BIT3		UNEXPECTED DATA TYPE ERROR
2816		000020	E.CLAT=	BIT4		ATTENTION DID NOT RESET WITH CLEAR
2817		000040	E.SCLR=	BIT5		SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION
2818						
2819		000100	E.ILLD=	BIT6		ILLEGAL DRIVER COMMAND
2820		000400	E.DLT=	BIT8		DATA LATE WHEN UNLOADING HEADER
2821		001000	E.CERR=	BIT9		CONTROLLER ERROR DURING DRIVER SERVICING
2822		002000	E.DPAR=	BIT10		DRIVE DETECTED PARITY ERROR
2823		040000	E.CMTO=	BIT14		CONTROLLER COMMAND TIME OUT (QUEUED ONLY)
2824		100000	E.MDS=	BIT15		MULTIPLE DRIVE SELECT
2825						
2826	003054	000000	O.WAIT:	.WORD	0	PARAMETER BLOCK OF THE DRIVE
2827						WAITING FOR COMMAND COMPLETION
2828	003056	000400	W.MTIM:	.WORD	400	LOOP COUNTER FOR MILLISECOND SCAN OF DRIVE
2829	003060	000400	W.MILI:	.WORD	400	16 MILLISECOND TIME FOR PROGRAM
2830						
2831						
2832						
2833						
2834						
2835						
2836						
2837						
2838						
2839						
2840						
2841						

CPU	VALUE
---	----
11/05	100
11/10	
11/20	
11/34	
11/40	
11/45	400
11/50	
11/70	

DRIVER PARAMENTERS

```

2842 003062 000300      W.SEC: .WORD 300      ; SECOND COUNT COUNT FOR ALL COMMANDS
2843                                     ; EXCEPT START SPINDLE
2844 003064 003000      W.BSEC: .WORD 3000    ; 8 SECOND FOR DRIVE CYCLE DOWN
2845 003066 030000      W.MIN:  .WORD 30000   ; MINUTE TIME FOR START SPINDLE
2846 003070 000000      HDR.AD:  .WORD 0      ; ADDRESS USED FOR READ ALL HEADERS
2847 003072 000000      HDR.CT:  .WORD 0      ; NUMBER OF HEADERS LEFT TO READ FOR READ
2848                                     ; ALL HEADERS
2849 003074 000         I.ISRL:  .BYTE 0      ; INTERRUPT OR RELEASED COMMAND ISSUED
2850 003075 002         H.HEAD:  .BYTE 2,4,10 ; HEAD DECODES
2851 003100 000         W.TIME:  .BYTE 0      ; DRIVES BEING WATCH-DOG TIMED
2852                                     ;
2853                                     .SBTTL INTERRUPT MASKS
2854                                     ;
2855 003101 000         INTMSK: .BYTE 0      ; INTERRUPT MASKS FOR DRIVE IN PARAMETER BLOCK
2856                                     ;
2857                                     ; INTERRUPT MASK TABLE
2858                                     ;
2859 003102 001         I.DRV:  .BYTE 1      ; INTERRUPT MASK FOR DRIVE 0
2860 003103 002         .BYTE 2      ; INTERRUPT MASK FOR DRIVE 1
2861 003104 004         .BYTE 4      ; INTERRUPT MASK FOR DRIVE 2
2862 003105 010         .BYTE 10     ; INTERRUPT MASK FOR DRIVE 3
2863 003106 020         .BYTE 20     ; INTERRUPT MASK FOR DRIVE 4
2864 003107 040         .BYTE 40     ; INTERRUPT MASK FOR DRIVE 5
2865 003110 100         .BYTE 100    ; INTERRUPT MASK FOR DRIVE 6
2866 003111 200         .BYTE 200    ; INTERRUPT MASK FOR DRIVE 7
2867                                     ;
2868                                     .SBTTL PARAMETER BLOCK TABLE
2869                                     ;
2870 003112 002630      PBLKT:  PARM0     ; ADDRESS OF PARAMETER BLOCK GIVEN WITH
2871                                     ; DRIVE CALL. MUST BE LOADED INTO PBLKT
2872                                     ;
2873                                     ;
2874                                     .SBTTL TIME FOR WATCH-DOG TIMER
2875                                     ;
2876 003114 000000      W.DRV:  .WORD 0      ; TIME FOR INSTRUCTION IN PARAMETER BLOCK
2877                                     .SBTTL PROGRAM SPECIFIC RESERVED LOCATIONS
2878 003116 000         MDFLAG: .BYTE 0      ; FLAG TO INDIC. DEFLT OR PARAM MODE
2879 003117 000         TSTING: .BYTE 0      ; CURRENTLY RUNNING TESTS IF = 1
2880 003120 000         DERCNT: .BYTE 0      ; DATA ERROR COUNT
2881 003121 000         OPCOMP: .BYTE 0      ; OPERATION COMPLETE FLAG
2882 003122 000         DONE:   .BYTE 0      ; DONE SWITCH
2883 003123 000         TYPFMT: .BYTE 0      ; DRIVE TYPE & FORMAT CONTROL
2884 003124 000         FORMAT: .BYTE 0      ; DRIVE FORMAT IN BIT 4 OF BYTE
2885 003125 000         ERRCNT: .BYTE 0      ; ERROR COUNT
2886 003126 004         ERRLMT: .BYTE 4      ; ERROR LIMIT
2887 003127 000         DRVERS: .BYTE 0      ; ERROR COUNT FOR CURRENT DRIVE
2888 003130 000         OPCONT: .BYTE 0      ; OPERATION CONTROL SWITCHES
2889 003131 000         DRMAFG: .BYTE 0      ; =1 INDICATES DRIVE SIEZED BY OTHER PORT
2890 003132 000         REISSU: .BYTE 0      ; DUAL-ACC FLAG TO RE-ISSUE COMMAND
2891 003133 000         TSTTYP: .BYTE 0      ; OFFSET TEST IDENTIFIER
2892 003134 000         DCEFLG: .BYTE 0      ; DATA CHECK ERROR FLAG
2893 003135 000         WCEFLG: .BYTE 0      ; WRITE CHECK ERROR FLAG
2894 003136 000         DLTF LG: .BYTE 0      ; DATA LATE ERROR FLAG
2895 003137 000         DIF100: .BYTE 0      ; "CYL DIF = 100" FLAG
2896 003140 000         NORTRY: .BYTE 0      ; "NO-RETRY" FLAG
2897 003141 000         MEMABT: .BYTE 0      ; SET BYTE = 1 FOR NO ABORT

```


L05

Address	Hex	Dec	Label	Type	Value	Description
2898						ON MEMORY PARITY ERRORS
2899	003142	000	HLPOVL:	.BYTE	0	HELP FILE OVERLAID INDICATOR
2900	003143	000	NOTYPE:	.BYTE	0	INDICATES PROGRAM JUST LOADED IF 0
2901	003144	000	LOGDRV:	.BYTE	0	LOGICAL DRIVE NUMBER (00-17)
2902	003145	000	PASSNO:	.BYTE	0	PASS NUMBER (= 1 OR 2)
2903	003146	000	OFSDIR:	.BYTE	0	OFFSET DIRECTION FLAG (0=NEG)
2904	003147	000	OFSCNT:	.BYTE	0	OFFSET COUNT FOR ONE DIRECTION
2905		000001	WHDSW=BIT0			WRITE HEADER & DATA SWITCH
2906		000002	VHDSW=BIT1			VERIFY HEADERS SWITCH
2907		000004	WCDSW=BIT2			WRITE CHECK DATA SWITCH
2908		000010	RCDASW=BIT3			READ CHECK DATA SWITCH
2909		000020	OREQSW=BIT4			OFFSET REQUIRED SWITCH
2910						
2911						
2912		000114	MEMVEC=114			MEMORY PARITY TRAP VECTOR
2913	003150	000000	SAVWRD:	.WORD	0	SCRATCH WORD
2914	003152	000400	BSSOFT:	.BLKW	↑0256	RECORD OF BAD SECTORS FROM SOFTWARE
2915	004152	000400	BSFACT:	.BLKW	↑0256	RECORD OF B.O SECTORS FROM FACTORY
2916	005152	000000	PRVCMD:	.WORD	0,0,0,0,0,0	PREVIOUS COMMAND STORAGE
2917	005160	000000				
2918	005166	000000	COMSTR:	.WORD	0,0,0,0,0,0	CURRENT COMMAND STORAGE
2919	005174	000000				
2920	005202	000102	BUFF0:	.BLKW	↑066	OUTPUT BUFFER 1
2921	005406	000000	LOWOCT:	.WORD	0	LOW 16 BITS OF CONVERTED BINARY NUMBER
2922	005410	000000	HIGOCT:	.WORD	0	HIGH BITS OF CONVERTED BINARY NO.
2923	005412	000000	BUFPR:	.WORD	0	BUFFER POINTER
2924	005414	000000	RECODE:	.WORD	0	RECOVERY CODE WORD
2925	005416	000000	ERRCOM:	.WORD	0	ERROR COMMAND
2926	005420	000000	DRIVE:	.WORD	0	NO. OF DRIVE IN USE
2927	005422	000000	STALLS:	.WORD	0	CURRENT NO. OF UNIT STALLS TO APPLY
2928	005424	000000	CYLNDR:	.WORD	0	CURRENT CYLINDER NUMBER
2929	005426	000000	OFINUS:	.WORD	0	OFFSET IN USE
2930	005430	000000	TRACK:	.WORD	0	TRACK IN USE
2931	005432	000000	INTCHR:	.WORD	0	TTY INTERRUPT INPUT WORD
2932	005434	000000	SCRACH:	.WORD	0	ALL-PURPOSE SCRATCH WORD
2933	005436	000000	PATRN:	.WORD	0	DATA PATTERN LIST WORD
2934	005440	000000	PLOFST:	.WORD	0	(+) OFFSET VALUE
2935	005442	000000	NGOFST:	.WORD	0	(-) OFFSET VALUE
2936	005444	000005	SAVPRS:	.BLKW	5	SAVE INITIAL PARAMS FOR XFER
2937	005446	000000	WDSXFR:	.WORD	0	NO. OF WORDS ACTUALLY XFERRED
2938	005450	000000	FINCYL:	.WORD	0	FINAL CYLINDER ADRS
2939	005452	000	FINTRK:	.BYTE	0	FINAL TRACK ADRS
2940	005453	000	FINSEC:	.BYTE	0	FINAL SECTOR ADRS
2941	005454	000000	LASTWC:	.WORD	0	ACTUAL FINAL WC
2942	005466	000000	NEWON:	.WORD	0	NEW LOOK AT ONLINE DRIVES
2943	005470	000002	MA:	.BLKW	2	
2944	005474	000002	PMA:	.BLKW	2	MEM. BUFFER ADDRESS
2945	005500	000000	SUBSYS:	.WORD	0	CURRENT SUBSYSTEM NAME STOR (ASCII)
2946	005502	000021	DRVLST:	.BLKW	17.	EACH WORD CONTAINS THE NAME OF A DRIVE TO BE TESTED. HIGH BYTE IS SUBSYSTEM NAME, LOW BYTE IS DRIVE NO. (ASCII). IF WORD = 0, DRIVE IS NOT TESTED.
2947						
2948						
2949						
2950	005544	000000	DRVPTN:	.WORD	0	DRIVE LIST POINTER
2951	005546	000010	SCRLLST:	.BLKW	8.	SCRATCH DRIVE LIST
2952						
2953	005556		PAT00:			

2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009

005556 000620 000000
005556 000620 000002
005562 000620 000004
005572 000620 000006
005576 000620 000010
005602 000620 000012
005606 000620 000014
005612 000620 000016
005616 000622 000000
005622 000622 000002
005626 000622 000004
005632 000622 000006
005636 000622 000010
005642 000622 000012
005646 000622 000014
005652 000622 000016

:TABLE A - BASIC READ/WRITE TEST SECTORS

TABLE A:

.WORD 620,0 :DRIVE 0
.WORD 620,2 :DRIVE 1
.WORD 620,4 :DRIVE 2
.WORD 620,6 :DRIVE 3
.WORD 620,10 :DRIVE 4
.WORD 620,12 :DRIVE 5
.WORD 620,14 :DRIVE 6
.WORD 620,16 :DRIVE 7
.WORD 622,0 :DRIVE 10
.WORD 622,2 :DRIVE 11
.WORD 622,4 :DRIVE 12
.WORD 622,6 :DRIVE 13
.WORD 622,10 :DRIVE 14
.WORD 622,12 :DRIVE 15
.WORD 622,14 :DRIVE 16
.WORD 622,16 :DRIVE 17

:TABLE B - WORST CASE DATA PATTERN

TABLE B:

072307
135143
156461
167230
073514
035646
016723
107351
143564
061672
030735
114356
046167
123073
151453
164616

:TABLE OF STARTING CYLINDERS FOR ALL OVWRT AND COMPAT CYL BLKS

TABLE C:

.WORD 0,60 :FOR CURRENT ZONE 1
.WORD 100,150 :FOR CURRENT ZONE 2
.WORD 200,260 :FOR CURRENT ZONE 3
.WORD 300,360 :FOR CURRENT ZONE 4
.WORD 400,460 :FOR CURRENT ZONE 5
.WORD 500,560 :FOR CURRENT ZONE 6
.WORD 600 :FOR CURRENT ZONE 7

005716 000000 000060
005722 000100 000160
005726 000200 000260
005732 000300 000360
005736 000400 000460
005742 000500 000560
005746 000600

3066 006441 013 014 015
 3067 006444 017 020 021
 3068 006447 022

3069
 3070
 3071
 3072
 3073
 3074
 3075
 3076
 3077
 3078
 3079
 3080
 3081
 3082
 3083
 3084
 3085
 3086
 3087
 3088
 3089
 3090
 3091
 3092
 3093
 3094
 3095
 3096
 3097
 3098
 3099
 3100
 3101
 3102
 3103
 3104
 3105
 3106
 3107
 3108
 3109
 3110
 3111
 3112
 3113
 3114
 3115
 3116
 3117
 3118
 3119
 3120
 3121

006450 000020
 006510 000020
 006550 000020
 006610 000020
 006650 000020
 006710 000020
 006750 000020
 007010 000020
 007050 000020
 007110 000020
 007150 000020
 007210 000020
 007250 000000
 007252 000000

:TABLE OF OVERWRITE TEST SCORES FOR TRACK 0 FOR NEG OFST
 NOSCR0: .BLKW 16.
 :TABLE OF OVERWRITE TEST SCORES FOR TRACK 1 FOR NEG OFST
 NOSCR1: .BLKW 16.
 :TABLE OF OVERWRITE TEST SCORES FOR TRACK 2 FOR NEG OFST
 NOSCR2: .BLKW 16.
 :TABLE OF OVERWRITE TEST SCORES FOR TRACK 0 FOR POS OFST
 POSCR0: .BLKW 16.
 :TABLE OF OVERWRITE TEST SCORES FOR TRACK 1 FOR POS OFST
 POSCR1: .BLKW 16.
 :TABLE OF OVERWRITE TEST SCORES FOR TRACK 2 FOR POS OFST
 POSCR2: .BLKW 16.
 :TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 0 FOR NEG OFST
 NCSCR0: .BLKW 16.
 :TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 1 FOR NEG OFST
 NCSCR1: .BLKW 16.
 :TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 2 FOR NEG OFST
 NCSCR2: .BLKW 16.
 :TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 0 FOR POS OFST
 PCSCR0: .BLKW 16.
 :TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 1 FOR POS OFST
 PCSCR1: .BLKW 16.
 :TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 2 FOR POS OFST
 PCSCR2: .BLKW 16.
 :SELF-TEST SCORE FOR TRACK 0 FOR NEG OFST
 NSSCR0: .WORD 0
 :SELF-TEST SCORE FOR TRACK 1 FOR NEG OFST
 NSSCR1: .WORD 0

```

3122
3123 :SELF-TEST SCORE FOR TRACK 2 FOR NEG OFST
3124 007254 000000 NSSCR2: .WORD 0
3125
3126
3127
3128 :SELF-TEST SCORE FOR TRACK 0 FOR POS OFST
3129 007256 000000 PSSCR0: .WORD 0
3130
3131 :SELF-TEST SCORE FOR TRACK 1 FOR POS OFST
3132 007260 000000 PSSCR1: .WORD 0
3133
3134 :SELF-TEST SCORE FOR TRACK 2 FOR POS OFST
3135 007262 000000 PSSCR2: .WORD 0
3136
3137
3138
3139 :AVERAGE OF OVRWRT SCORES OVER ALL SURFACES
3140 007264 000000 OVRAVG: .WORD 0
3141
3142 :AVERAGE OF READ SCORES OVER ALL SURFACES
3143 007266 000000 COMAVG: .WORD 0
3144
3145
3146
3147 :SCRATCH TABLE OF SCORES FOR TRACK 0 FOR NEG OFST
3148 007270 000020 NSCOR0: .BLKB 16.
3149
3150 :SCRATCH TABLE OF SCORES FOR TRACK 1 FOR NEG OFST
3151 007310 000020 NSCOR1: .BLKB 16.
3152
3153 :SCRATCH TABLE OF SCORES FOR TRACK 2 FOR NEG OFST
3154 007330 000020 NSCOR2: .BLKB 16.
3155
3156
3157
3158 :SCRATCH TABLE OF SCORES FOR TRACK 0 FOR POS OFST
3159 007350 000020 PSCOR0: .BLKB 16.
3160
3161 :SCRATCH TABLE OF SCORES FOR TRACK 1 FOR POS OFST
3162 007370 000020 PSCOR1: .BLKB 16.
3163
3164 :SCRATCH TABLE OF SCORES FOR TRACK 2 FOR POS OFST
3165 007410 000020 PSCOR2: .BLKB 16.
3166
3167
3168
3169 000632 LSTCYL=632 ;LAST CYL. = 410(10)
3170 000002 LSTTRK=2 ;LAST TRACK = 2
3171 000002 BSERR=BIT1 ;BSE ERROR
3172 000004 HVR CER=BIT2 ;HVRC ERROR
3173 000010 OPIERR=BIT3 ;OPI ERROR
3174 000020 DCKERR=BIT4 ;DATA CHECK ERROR
3175 000040 ECCNC=BIT5 ;ECC NON-CORRECTABLE
3176 000100 WCERR=BIT6 ;WRITE CHECK ERROR
3177 000200 ABORT=BIT7 ;ABORT

```

3178	000400
3179	001000
3180	002000
3181	004000
3182	010000
3183	100000
3184	
3185	007430 005015 046412 026504
3186	007436 030461 042055 051132
3187	007444 050466 040455 026440
3188	007452 051040 030113 020066
3189	007460 051104 053111 020105
3190	007466 047503 050115 052101
3191	007474 041111 046111 052111
3192	007502 020131 051120 043517
3193	007510 040522 006515 005012
3194	007516 000
3195	007517 012 045522 033060
3196	007524 041040 051525 040440
3197	007532 051104 036440 000040
3198	007540 045522 033060 053040
3199	007546 041505 040440 051104
3200	007554 036440 000040
3201	007560 045522 033060 050040
3202	007566 044522 051117 052111
3203	007574 020131 000075
3204	007600 052012 050131 020105
3205	007606 040516 042515 047440
3206	007614 020106 042516 052130
3207	007622 051440 041125 054523
3208	007630 020123 047524 052040
3209	007636 051505 020124 040450
3210	007644 050055 020051 020072
3211	007652 000
3212	007653 123 041125 054523
3213	007660 020123 020040 051104
3214	007666 053111 024105 024523
3215	007674 036440 000040
3216	007700 044527 046114 052040
3217	007706 051505 020124 051104
3218	007714 053111 024105 024523
3219	007722 000040
3220	007724 047440 020116 052523
3221	007732 051502 051531 020040
3222	007740 035015 000012
3223	007744 051511 052040 042510
3224	007752 042522 040440 047516
3225	007760 044124 051105 051440
3226	007766 041125 054523 020123
3227	007774 054450 047440 020122
3228	010002 020116 041474 037122
3229	010010 020051 020077 000
3230	010015 012 047515 042522
3231	010022 052040 040510 020116
3232	010030 033061 042040 044522
3233	010036 042526 020123 047105

```

LEVEL=BIT8           ; LEVEL TWO ERROR
BADSEC=BIT9         ; BAD SECTOR FLAG
TWOLOS=BIT10        ; TWO TIME OUTS
BCLREQ=BIT11        ; RECALIBRATE REQUIRED
DRAVFL=BIT12        ; DRIVE NOT RELEASED BY OTHER PORT
ANYDER=BIT15        ; ANY ERROR DETECTED FLAG

```

DZR60: .ASCIZ <15><12><12>/MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM/<15><12><1

RKBADR: .ASCIZ <12>/RK06 BUS ADR = /

RKVADR: .ASCIZ /RK06 VEC ADR = /

RKPRTY: .ASCIZ /RK06 PRIORITY = /

TYP SYS: .ASCIZ <12>/TYPE NAME OF NEXT SUBSYS TO TEST (A-P) : /

SYS DRV: .ASCIZ /SUBSYS DRIVE(S) = /

WILTST: .ASCIZ /WILL TEST DRIVE(S) /

ON SUBS: .ASCIZ / ON SUBSYS /<15><12><12>

ANOTHR: .ASCIZ /IS THERE ANOTHER SUBSYS (Y OR N <CR>) ? /

DROVFL: .ASCIZ <12>/MORE THAN 16 DRIVES ENTERED !/<15><12><12>

3234	010044	042524	042522	020104	
3235	010052	006441	005012	000	
3236	010057	012	025052	051440	SRPASS: .ASCIZ <12>/** STARTING PASS **/<15><12>
3237	010064	040524	052122	047111	
3238	010072	020107	040520	051523	
3239	010100	020040	025040	006452	
3240	010106	000012			
3241	010110	046412	052517	052116	MNTPAK: .ASCIZ <12>/MOUNT PACK ON DRIVE AND LOAD./<15><12>
3242	010116	050040	041501	020113	
3243	010124	047117	042040	044522	
3244	010132	042526	020040	020040	
3245	010140	047101	03104	047514	
3246	010146	042101	006456	000012	
3247	010154	052412	046116	040517	DISPAK: .ASCIZ <12>/UNLOAD DRIVE AND REMOVE PACK./<15><12>
3248	010162	020104	051104	053111	
3249	010170	020105	020040	040440	
3250	010176	042116	051040	046505	
3251	010204	053117	020105	040520	
3252	010212	045503	01456	000012	
3253	010220	051412	040524	052122	STR204: .ASCIZ <12>/START AT ADR 204 FOR PASS 1 ON SUBSYS ./<15><12><12>
3254	010226	040440	020124	042101	
3255	010234	020122	030062	020064	
3256	010242	047506	020122	040520	
3257	010250	051523	030440	047440	
3258	010256	020116	052523	051502	
3259	010264	051531	020040	006456	
3260	010272	005012	000		
3261	010275	012	052123	051101	STR220: .ASCIZ <12>/START AT ADR 220 FOR PASS 2 ON SUBSYS ./<15><12><12>
3262	010302	020124	052101	040440	
3263	010310	051104	031040	030062	
3264	010316	043040	051117	050040	
3265	010324	051501	020123	020062	
3266	010332	047117	051440	041125	
3267	010340	054524	020123	027040	
3268	010346	075015	000012		
3269	010352	054524	042520	051040	TYPRWN: .ASCIZ /TYPE R<CR> WHEN /
3270	010360	041474	037122	053440	
3271	010366	042510	020116	000	
3272	010373	104	044522	042526	DRIRDY: .ASCIZ /DRIVE READY : /
3273	010400	051040	040505	054504	
3274	010406	035040	000040		
3275	010412	047504	042516	035040	DRIDON: .ASCIZ /DONE : /
3276	010420	000040			
3277	010422	047516	042040	044522	NODRLS: .ASCIZ /NO DRIVES LISTED FOR THIS SUBSYS !/<15><12>
3278	010430	042526	020123	044514	
3279	010436	052123	042105	043040	
3280	010444	051117	052040	044510	
3281	010452	020123	052523	051502	
3282	010460	051531	020440	005015	
3283	010466	000			
3284	010467	015	051412	047503	RESULTS: .ASCII <15><12>/SCORES FOR DRIVE (0-12 DEC.) :/<15><12><12>
3285	010474	042522	020123	047506	
3286	010502	020122	051104	053111	
3287	010510	020105	020040	020040	
3288	010516	030050	030455	020062	
3289	010524	042504	027103	020051	

3290	010532	006472	005012						
3291	010536	051124	041501	004513	.ASCII	/TRACK	DRIVE	OVRWRT	OVRWRT READ READ/<15><12>
3292	010544	051104	053111	004505					
3293	010552	053117	053522	052122					
3294	010550	047411	051126	051127					
3295	010556	004524	042522	042101					
3296	010574	051011	040505	006504					
3297	010602	012							
3298	010603	116	046525	051011	.ASCII	/NUM	READ	SCORE	SCORE SCORE SCORE SCORE/<15><12>
3299	010610	040505	004504	041523					
3300	010616	051117	004505	041523					
3301	010624	051117	004505	041523					
3302	010632	051117	004505	041523					
3303	010640	051117	006505	012					
3304	010645	011	047411	051506	.ASCIZ	/		OFST-	OFST+ OFST- OFST+/<15><12><12>
3305	010652	026524	047411	051506					
3306	010660	025524	047411	051506					
3307	010666	026524	047411	051506					
3308	010674	025524	005015	000012					
3309	010702	047440	051126	051127	ORVERG:	.ASCIZ	/	OVRWRT	AVG = /
3310	010710	020124	053101	020107					
3311	010716	020075	000						
3312	010721	040	042522	042101	RAVERG:	.ASCIZ	/	READ	AVG = /
3313	010726	040440	043526	020040					
3314	010734	036440	000040						
3315	010740	005015	025012	020052	ENDTST:	.ASCIZ	<15><12><12>/**	END OF TESTING	**/<15><12>
3316	010746	042440	042116	047440					
3317	010754	020106	042524	052123					
3318	010762	047111	020107	025040					
3319	010770	006452	000012						
3320	010774	020040	000011		TRKBUF:	.ASCIZ	/	/	
3321	011000	020040	004440	000	DRVBUF:	.ASCIZ	/	/	
3322	011005	123	046105	004506	SELF:	.ASCIZ	/	SELF	/
3323	011012	000							
3324	011013	011	000		TAB:	.ASCIZ	/	/	
3325	011015	123	051127	036440	SWRMSG:	.ASCIZ	/	SWR =	/
3326	011022	000040							
3327	011024	020040	042516	020127	NEWMSG:	.ASCIZ	/	NEW =	/
3328	011032	020075	000						
3329	011035	015	042012	044522	DRVSEQ:	.ASCIZ	<15><12>/	DRIVE(S) =	/
3330	011042	042526	051450	020051					
3331	011050	020075	000						
3332	011053	104	044522	042526	BADDRV:	.ASCIZ	/	DRIVE	/
3333	011060	020040	000040						
3334	011064	047516	026516	054105	NXDRIV:	.ASCIZ	/	NON-EXISTENT/<15><12>	
3335	011072	051511	042524	052116					
3336	011100	005015	000						
3337	011103	116	052117	051040	NTREDY:	.ASCIZ	/	NOT READY/<15><12>	
3338	011110	040505	054504	005015					
3339	011116	000							
3340	011117	127	044522	042524	WRTLOK:	.ASCIZ	/	WRITE-LOCKED/<15><12>	
3341	011124	046055	041517	042513					
3342	011132	006504	000012						
3343	011136	047514	042101	042105	ALNPAK:	.ASCIZ	/	LOADED WITH ALIGN PACK/<15><12>	
3344	011144	053440	052111	020110					
3345	011152	046101	043511	020116					

3346	011160	040520	045503	005015	
3347	011166	000			
3348	011167	111	020106	054130	REPLPK: .ASCIZ /IF XXDP PACK ON DRV 0, TYPE "Y <CR>", & REPLACE IT : /
3349	011174	050104	050040	041501	
3350	011202	020113	047117	042040	
3351	011210	053122	030040	020054	
3352	011216	054524	042520	021040	
3353	011224	020131	041474	037122	
3354	011232	026042	023040	020040	
3355	011240	042522	046120	041501	
3356	011246	020105	052111	035040	
3357	011254	007040			
3358	011256	000015	025012	020052	NOORTS: .ASCII <15><12><12>/** NO DRIVES TO TEST/<15><12>
3359	011264	047516	042040	044522	
3360	011272	042526	020123	047524	
3361	011300	052040	051505	006524	
3362	011306	012			
3363	011307	120	042522	051523	CNTRDY: .ASCIZ /PRESS "CONT" WHEN RDY/<15><12>
3364	011314	021040	047503	052116	
3365	011322	020042	044127	047105	
3366	011330	051040	054504	005015	
3367	011336	000			
3368	011337	110	046101	020124	HLTRQD: .ASCIZ /HALT REQUESTED/<15><12>
3369	011344	042522	052521	051505	
3370	011352	042524	006504	000012	
3371	011360	051104	053111	020105	DROXDP: .ASCIZ /DRIVE 0 IS LOAD MEDIUM/<15><12>
3372	011366	020060	051511	046040	
3373	011374	040517	020104	042515	
3374	011402	044504	046525	005015	
3375	011410	000			
3376	011411	015	025012	020052	DROPDR: .ASCIZ <15><12>/** DROPPING DRIVE - 20 ERRORS/<15><12>
3377	011416	042040	047522	050120	
3378	011424	047111	020107	051104	
3379	011432	053111	020105	020055	
3380	011440	030062	042440	051122	
3381	011446	051117	006523	000012	
3382	011454	005015	052012	051505	TSTDNR: .ASCII <15><12><12>/TESTING DRIVE /
3383	011462	044524	043516	042040	
3384	011470	044522	042526	040	
3385	011475	040	005015	000	DRVNO: .ASCIZ / /<15><12>
3386	011501	104	044522	042526	DRIV: .ASCIZ /DRIVE /
3387	011506	000040			
3388	011510	040503	052122	020056	CART: .ASCIZ /CART. /
3389	011516	000			
3390	011517	123	051105	020056	SERNM: .ASCIZ /SER. NO. /
3391	011524	047516	020056	000040	
3392	011532	005015	040506	052103	FACTBS: .ASCIZ <15><12>/FACTORY /
3393	011540	051117	020131	000	
3394	011545	012	047523	052106	SOFTBS: .ASCII <12>/SOFTWARE /
3395	011552	040527	042522	040	
3396	011557	102	042101	051440	BOSECT: .ASCIZ /BAD SECTORS :/<15><12>
3397	011564	041505	047524	051522	
3398	011572	035040	005015	000	
3399	011577	040	047040	047117	NOFALS: .ASCIZ / NONE/
3400	011604	000105			
3401					

```

3402 011606 025052 020040 040503 BAD632: .ASCIZ /** CANNOT READ BAD SECTOR TRACK!/(15)<(12)
3403 011614 047116 052117 051040
3404 011622 040505 020104 040502
3405 011630 020104 042523 052103
3406 011636 051117 052040 040522
3407 011644 045503 006441 000012
3408 011652 041536 005015 000 CNTRLC: .ASCIZ /!C/(15)<(12)
3409 011657 136 006532 000012 CNTRLZ: .ASCIZ /!Z/(15)<(12)
3410 011664 051136 005015 000 CNTRLR: .ASCIZ /!R/(15)<(12)
3411 011671 136 006525 000012 CNTRLU: .ASCIZ /!U/(15)<(12)
3412 011676 043536 005015 000 CNTRLG: .ASCIZ /!G/(15)<(12)
3413 011703 015 005012 000 CR2LF: .ASCIZ <(15)<(12)<(12)
3414 011707 134 000 BKSLSH: .ASCIZ /\
3415 011711 054 000 COMMA: .ASCIZ /,/
3416 011713 040 040 SPACE6: .ASCII //
3417 011715 040 SPACE4: .ASCII //
3418 011716 040 SPACE3: .ASCII //
3419 011717 040 SPACE2: .ASCII //
3420 011720 000040 SPACE1: .ASCIZ //
3421 .EVEN
3422 011722 020040 000075 PRMBUF: .ASCIZ / =/
3423 011726 047527 042122 000040 WORDSP: .ASCIZ /WORD /
3424 011734 036440 000040 EQUALS: .ASCIZ / = /
3425 011740 020052 000 PROMPT: .ASCIZ /* /
3426 011743 076 000040 PRMPSP: .ASCIZ /> /
3427
3428 .EVEN
3429
3430
3431
3432
3433 011746 105037 003116 DFSTRT: CLRB MDFLAG ;SET FLAG FOR 200 START
3434 011752 000403 BR PISET ;CONTINUE
3435
3436 011754 112737 000001 003116 P1STRT: MOVB #1,MDFLAG ;SET FLAG FOR 204 START
3437 011762 112737 000061 003145 PISET: MOVB #1,PASSNO ;SET PASS NO. = 1
3438 011770 000406 BR CMSTRT ;CONTINUE
3439
3440 011772 112737 000002 003116 P2STRT: MOVB #2,MDFLAG ;SET FLAG FOR 220 START
3441 012000 112737 000062 003145 MOVB #2,PASSNO ;SET PASS NO. = 2
3442
3443 012006 012737 000340 177776 CMSTRT: MOV #PR7,@#PS ;BLOCK ALL INTERRUPTS
3444 .SBTTL INITIALIZE THE COMMON TAGS
3445 ;;CLEAR THE COMMON TAGS ($CHTAG) AREA
3446 012014 012706 001100 MOV #SCHTAG,R6 ;;FIRST LOCATION TO BE CLEARED
3447 012020 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
3448 012022 022706 001140 CMP #SWR,R6 ;;DONE?
3449 012026 001374 BNE -6 ;;LOOP BACK IF NO
3450 012030 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
3451 ;;INITIALIZE A FEW VECTORS
3452 012034 012737 043742 000020 MOV #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
3453 012042 012737 000340 000022 MOV #340,@#IOTVEC+2 ;;LEVEL 7
3454 012050 012737 043236 000030 MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
3455 012056 012737 000340 000032 MOV #340,@#EMTVEC+2 ;;LEVEL 7
3456 012064 012737 044454 000034 MOV #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
3457 012072 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7

```

```

3458 012100 012737 043606 000024      MOV      #SPWRON, @#PWVEC      ;; POWER FAILURE VECTOR
3459 012106 012737 000340 000026      MOV      #340, @#PWVEC+2      ;; LEVEL 7
3460 012114 013737 016676 016670      MOV      SENDCT, SEOPCT      ;; SETUP END-OF-PROGRAM COUNTER
3461 012122 005037 001316          CLR      $ESCAPE              ;; CLEAR THE ESCAPE ON ERROR ADDRESS
3462 012126 112737 000001 001115      MOV      #1, $ERMAX          ;; ALLOW ONE ERROR PER TEST
3463 012134 012737 012134 001106      MOV      #., $LPAOR          ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
3464 012142 012737 012142 001110      MOV      #., $LPERA          ;; SETUP THE ERROR LOOP ADDRESS
3465          ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3466          ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
3467 012150 013746 000004          MOV      @#ERRVEC, -(SP)      ;; SAVE ERROR VECTOR
3468 012154 012737 012210 000004      MOV      #64$, @#ERRVEC      ;; SET UP ERROR VECTOR
3469 012162 012737 177570 001140      MOV      #0$SWR, SWR         ;; SETUP FOR A HARDWARE SWICH REGISTER
3470 012170 012737 177570 001142      MOV      #00$ISP, DISPLAY    ;; AND A HARDWARE DISPLAY REGISTER
3471 012176 022777 177777 166734      CMP      #-1, @SWR           ;; TRY TO REFERENCE HARDWARE SWR
3472 012204 001012          BNE      65$                 ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
3473          ;; AND THE HARDWARE SWR IS NOT = -1
3474 012206 000403          BR       65$                 ;; BRANCH IF NO TIMEOUT
3475 012210 012716 012216 64$:      MOV      #65$, (SP)          ;; SET UP FOR TRAP RETURN
3476 012214 000002          RTI                          ;;
3477 012216 012737 000176 001140 65$:      MOV      #SWREG, SWR         ;; POINT TO SOFTWARE SWR
3478 012224 012737 000174 001142      MOV      #DISPREG, DISPLAY    ;;
3479 012232 012637 000004 66$:      MOV      (SP)+, @#ERRVEC     ;; RESTORE ERROR VECTOR
3480
3481 012236 005037 001336          CLR      $PASS              ;; CLEAR PASS COUNT
3482 012242 132737 000200 001351      BITB    #APTSIZE, $ENVM      ;; TEST USER SIZE UNDER APT
3483 012250 001403          BEQ     67$                 ;; YES, USE NON-APT SWITCH
3484 012252 012737 001352 001140 67$:      MOV      #$$SWREG, SWR       ;; NO, USE APT SWITCH REGISTER
3485 012260
3486 012260 000005          RESET                       ;; CLEAR THE UNIBUS
3487 012262 012737 000006 000004      MOV      #6, @#ERRVEC        ;; SET TIME-OUT VECTOR
3488 012270 005037 000006          CLR      @#ERRVEC+2          ;;
3489 012274 105037 003133          CLR      TSTTYP              ;; CLEAR OFFSET TEST IDENTIFIER
3490 012300 105037 003127          CLR      DRVERS              ;; CLEAR ERROR COUNT FOR CURRENT DRIVE
3491 012304 105037 003125          CLR      ERRCNT              ;; CLEAR ERROR RETRY COUNT
3492 012310 105037 001102          CLR      $STNM              ;; SET TEST NO. = 0
3493 012314 112737 000001 003142      MOV      #1, HLPVOL          ;; SET HLP FILE OVL D INDICTR
3494 012322 104401 007430          TYPE    , DZR60             ;; TYPE PROGRAM I.D.
3495 012326 105737 003143          TSTB    #NOTYPE             ;; SEE IF OPERATOR NOTE SHOULD BE TYPED
3496 012332 001004          BNE     40$                 ;; BR IF NOT
3497 012334 104401 053476          TYPE    , NOTMSG            ;; TYPE OPERATOR NOTE
3498 012340 105237 003143          INCB    #NOTYPE             ;; SET FLAG FOR NEXT TIME
3499 012344 012737 020262 000060 40$:      MOV      #KBHDL, @#TKVEC     ;; LOAD VECTOR FOR TTY KBD
3500 012352 012737 000200 000062      MOV      #PR4, @#TKVEC+2     ;; SET KBD PRIORITY = 4
3501 012360 013701 003040          MOV      RKVEC, R1           ;; ADDR. OF RK06 VECTOR STORAGE
3502 012364 012721 034056          MOV      #I.INTR, (R1)+      ;; SET IT TO RK06 HANDLER
3503 012370 013711 003042          MOV      RKPRI, (R1)         ;; SET RK06 PRIORITY
3504 012374 105037 003117          CLR      TSTING              ;; CLEAR "RUNNING TESTS" FLAG
3505 012400 012737 000000 177776      MOV      #PRO, @#PS          ;; ALLOW ALL INTERRUPTS AGAIN
3506 012406 012701 005152          MOV      #PRVCMO, R1         ;; ZERO OUT PREVIOUS COMMAND
3507 012412 012700 000006          MOV      #6, RO              ;; SET COUNTER
3508 012416 005021 42$:      CLR      (R1)+              ;; ZERO A WORD
3509 012420 005300          DEC     RO                   ;;
3510 012422 001375          BNE     42$                 ;; BR IF NOT DONE YET
3511 012424 005037 005422          CLR      STALLS              ;; DON'T ALLOW STALLS YET
3512 012430 004737 020564          JSR     PC, GTSWRG           ;; OPEN SOFTWARE SWR FOR MODIFICATION
3513 012434 012737 176543 041504 44$:      MOV      #176543, $HINUM     ;; INIT. PSEUDO-RANDOM NOS.

```

```

3514 012442 012737 123456 041506      MOV      #123456, $LONUM
3515 012450 105737 003116      9$: TSTB   MDFLAG
3516 012454 001034      BNE     ALLSYS      ; SEE IF DEFAULT MODE
3517 012456 013700 001400      MOV     $VECT1, R0   ; BR IF NOT DEFAULT MODE
3518 012462 013737 001404 003036      MOV     $BASE, RKBAS ; GET APT RK06 VECTOR AND PRTY
3519 012470 132737 000200 001351      BITB   #BIT7, $ENVM ; GET APT RK06 BASE ADDRESS
3520 012476 001005      BNE     18$         ; SEE IF NO SIZING
3521 012500 012700 120210      MOV     #AVECT1, R0  ; BR IF NO SIZING
3522 012504 012737 177440 003036      MOV     #ABASE, RKBAS ; GET DEFAULT VECTOR AND PRIORITY
3523 012512 110037 003040      18$: MOV    R0, R0      ; GET DEFAULT BASE ADDRESS
3524 012516 105037 003041      CLR    RKBAS+1      ; STORE VECTOR
3525 012522 000300      SWAB   R0           ; CLEAR HI BYTE
3526 012524 042700 177437      BIC    #177437, R0  ; GET PRTY INTO BITS 5-7
3527 012530 010037 003042      MOV    R0, R0      ; CLEAR OTHER BITS
3528 012534 112737 000101 005500      MOV    #A, $SUBSYS ; STORE RK06 PRIORITY
3529 012542 000137 013540      JMP    DFLDR       ; SET SUBSYSTEM NAME = A
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539 012546 104401 001325      ; *****
3540 012552 105037 001102      ; SBTTL TEST 0      PARAMETER INPUT ROUTINES
3541 012556 112703 000101      ; *****
3542 012562 012701 005502      ; READ AND STORE ALL SUBSYSTEMS AND DRIVES TO TEST
3543 012566 012700 000021      ALLSYS: TYPE   $CRLF      ; TYPE (CR), (LF)
3544 012572 005021      CLR     $STNM        ; SET TEST NO. = 0
3545 012574 005300      MOV    #A, R3        ; INIT SUBSYS INDICATOR
3546 012576 001375      MOV    #DRVLST, R1   ; GET DRIVE LIST ADDRESS
3547 012600 012704 005502      2$:  MOV    #17, R0    ; CLEAR OUT THE DRIVE LIST
3548      CLR    (R1)+
3549 012604 110337 007662      DEC    R0
3550 012610 104401 007653      BNE    2$
3551 012614 064737 023006      MOV    #DRVLST, R4   ; GET STARTING ADRS OF DRIVE LIST
3552 012620 012546      ; ASK FOR DRIVES TO TEST ON THIS SUBSYSTEM
3553 012622 012546      ALLSY1: MOV    R3, $SYSDRV+7 ; SET SUBSYS NAME FOR PRINTOUT
3554 012624 012610      6$:  TYPE   $SYSDRV      ; ASK FOR DRIVES ON THIS SUBSYSTEM
3555 012626 005700      JSR    PC, $R0CHRS   ; READ INPUT STRING
3556 012630 001767      ALLSYS      ; (↑C) RETURN ADDRESS
3557 012632 022700 000017      ALLSYS      ; (↑Z) RETURN ADDRESS
3558 012636 002005      6$:  TST    R0         ; (↑U) RETURN ADDRESS
3559 012640 104401 005202      BEQ    6$           ; SEE IF NULL INPUT
3560 012644 104401 001324      CMP    #15, R0      ; BR TO ASK AGAIN
3561 012650 000757      BGE    12$         ; SEE IF TOO MANY CHARS TYPED
3562 012652 005001      10$: TYPE   , $BUFFO   ; BR IF NOT TOO MANY
3563 012654 126127 005202 000060      TYPE   , $QUES      ; ECHO BAD INPUT
3564 012662 103766      BR     6$          ; TYPE (↑) AND (CR), (LF)
3565 012664 126127 005202 000067      12$: CLR    R1       ; GO ASK AGAIN
3566 012672 101362      14$: CMP    $BUFFO(R1), #0 ; CLEAR CHAR POINTER
3567 012674 005201      BLO   10$         ; SEE IF DRIVE NO. < 0
3568 012676 020100      BHI   10$         ; BR TO ECHO BAD INPUT
3569 012700 001406      INC    R1         ; SEE IF CHAR > 7
3569      CMP    R1, R0    ; BR TO ECHO BAD INPUT
3569      BEQ    16$      ; INCR CHAR POINTER
3569      ; SEE IF MORE CHARS TO CHECK
3569      ; BR IF ALL CHARS CHECKED

```

K06

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 75
 TEST 0 PARAMETER INPUT ROUTINES

SEQ 0074

```

3570 012702 126127 005202 000054      CMPB   BUFFO(R1),',',      ;SEE IF THIS IS A COMMA
3571 012710 001353                    BNE    10$                ;BR IF NOT A COMMA
3572 012712 005201                    INC    R1                 ;INCR CHAR POINTER
3573 012714 000757                    BR     14$                ;BR TO CONTINUE CHECKING CHARS
3574                                ;PUT DRIVES FOR THIS SUBSYSTEM INTO SCRATCH LIST
3575 012716 012701 005546      16$:  MOV   #SCRATCH,R1      ;GET SCRATCH DRIVE LIST ADRS
3576 012722 005021                    CLR   (R1)+              ;CLEAR OUT SCRATCH DRIVE LIST
3577 012724 005021                    CLR   (R1)+
3578 012726 005021                    CLR   (R1)+
3579 012730 005011                    CLR   (R1)
3580 012732 005000                    CLR   R0                 ;INIT CHAR POINTER
3581 012734 116001 005202      18$:  MOVB  BUFFO(R0),R1      ;GET A DRIVE NO.
3582 012740 042701 000060                    BIC   #'D,R1             ;STRIP ASCII BITS
3583 012744 116061 005202      005546  MOVB  BUFFO(R0),SCRATCH(R1) ;SET ASCII DRV NO.
3584 012752 062700 000002                    ADD   #2,R0              ;POINT TO NEXT DRIVE NO.
3585 012756 105760 005201                    TSTB  BUFFO-1(R0)       ;SEE IF NULL CHAR YET
3586 012762 001364                    BNE   18$                ;BR IF NOT DONE YET
3587                                ;LOAD DRIVES FROM SCRATCH LIST INTO TOTAL DRIVE LIST
3588 012764 005000                    ALLSY3: CLR  R0           ;INIT SCRATCH DRV LIST POINTER
3589 012766 116014 005546      ALLSY2: MOVB  SCRATCH(R0),(R4) ;MOVE THIS DRIVE INTO ACTUAL LIST
3590 012772 001411                    BEQ   22$                ;BR IF DRIVE NOT MARKED
3591 012774 005204                    INC   R4                 ;INCR DRIVE LIST POINTER
3592 012776 110324                    MOVB  R3,(R4)+           ;PUT SUBSYS LETTER INTO HI BYTE
3593 013000 026427 005544                    CMP   R4,#DRVLST+34.    ;SEE IF DRIVE LIST OVERFLOW
3594 013004 103404                    BLO   22$                ;BR IF NO OVERFLOW YET
3595 013006 104401 010015                    TYPE  DROVFL            ;TYPE "MORE THAN 16 DRIVES ENTERED !"
3596 013012 000137 012546                    JMP   ALLSYS            ;GO START OVER, ASKING FOR DRIVES
3597 013016 005200                    22$:  INC   R0             ;INCR SCRATCH DRIVE LIST POINTER
3598 013020 020027 000010                    CMP   R0,#8.           ;SEE IF DONE WITH THIS SUBSYS
3599 013024 103760                    BLO   ALLSY2           ;BR IF NOT YET
3600                                ;ECHO DRIVES TO TEST ON THIS SUBSYSTEM
3601 013026 104401 007700                    TYPE  WILTST           ;TYPE "WILL TEST DRIVE(S) "
3602 013032 005037 005434                    CLR   SCRATCH          ;INIT CHAR OUTPUT BUF
3603 013036 005000                    CLR   R0               ;INIT SCRATCH LIST POINTER
3604 013040 005001                    CLR   R1               ;INIT COUNT OF LISTED DRIVES FOR SUBSYS
3605 013042 116037 005546      005434  23$:  MOVB  SCRATCH(R0),SCRATCH ;SEE IF THIS DRIVE LISTED
3606 013050 001407                    BEQ   26$                ;BR IF DRIVE NOT LISTED
3607 013052 005701                    TST  R1                 ;SEE IF FIRST TIME HERE
3608 013054 001402                    BEQ   24$                ;BR IF FIRST TIME
3609 013056 104401 011711                    TYPE  ',COMMA         ;TYPE A COMMA
3610 013062 104401 005434      24$:  TYPE  ',SCRATCH      ;TYPE A DRIVE NO.
3611 013066 005201                    INC   R1                 ;INCR. COUNT OF LISTED DRIVES FOR THIS SUBSYS
3612 013070 005200                    26$:  INC   R0             ;INCR LIST POINTER
3613 013072 020027 000010                    CMP   R0,#8.           ;SEE IF DONE CHECKING LIST YET
3614 013076 002761                    BLT   23$                ;BR IF NOT DONE YET
3615 013100 110337 007737                    MOVB  R3,ONSUBS+11.    ;SET SUBSYS NO. IN MSG
3616 013104 104401 007724                    TYPE  ONSUBS          ;TYPE "ON SUBSYSTEM "
3617 013110 005203                    INC   R3                 ;INCR SUBSYSTEM NAME
3618 013112 020327 000120                    CMP   R3,#'P          ;SEE IF NAME > P
3619 013116 101035                    BHI   ASKSYS           ;BR IF YES
3620 013120 105737 003116                    TSTB  MDFLAG          ;SEE IF 200 START
3621 013124 001002                    BNE   28$                ;BR IF NOT
3622 013126 000137 013500                    JMP   DAVTST          ;START PASS 1
3623                                ;ASK IF THERE IS ANOTHER SUBSYSTEM TO SPECIFY
3624 013132 104401 007744      28$:  TYPE  'ANOTHR        ;ASK FOR ANOTHER SUBSYS
3625 013136 004737 023006                    JSR   PC,ROCHRS        ;READ RESPONSE

```

L06

3626	013142	012546		ALLSYS				;(1C) RETURN ADDRESS
3627	013144	012546		ALLSYS				;(1Z) RETURN ADDRESS
3628	013146	013132		28\$;(1U) RETURN ADDRESS
3629	013150	005700		TST	RO			SEE IF NULL INPUT
3630	013152	001417		BEQ	ASKSYS			BR IF NULL INPUT, TO ASSUME (N)
3631	013154	022737	000131 005202	CMP	#'Y,BUFFO			SEE IF (Y) TYPED
3632	013162	001002		BNE	30\$			BR IF NOT (Y)
3633	013164	000137	012604	JMP	ALLSY1			GO ASK FOR DRIVES ON NEXT SUBSYS
3634	013170	022737	000116 005202 30\$:	CMP	#'N,BUFFO			SEE IF (N) TYPED
3635	013176	001405		BEQ	ASKSYS			BR IF (N)
3636	013200	104401	005202	TYPE	,BUFFO			ECHO BAD INPUT
3637	013204	104401	001324	TYPE	,SQUES			TYPE (') AND (CR), (LF)
3638	013210	000750		BR	28\$			GO ASK AGAIN
3639								
3640								
3641	013212	105037	001102	ASKSYS:	CLR8	\$TSTNM		;204 OR 220 START - GET NAME OF SUBSYSTEM TO TEST NEXT
3642	013216	104401	007E00	TYPE	TYPYS			SET TEST NO. = 0
3643	013222	004737	023006	JSR	PC,ROCHRS			ASK FOR NEXT SUBSYS TO TEST
3644	013226	012546		ALLSYS				READ RESPONSE
3645	013230	013212		ASKSYS				;(1C) RETURN ADDRESS
3646	013232	013212		ASKSYS				;(1Z) RETURN ADDRESS
3647	013234	005700		ASKSYS				;(1U) RETURN ADDRESS
3648	013236	001765		TST	RO			SEE IF NULL INPUT
3649	013240	023727	005202 000101	BEQ	ASKSYS			BR IF NULL, TO ASK AGAIN
3650	013246	103005		CMP	BUFFO,#'A			SEE IF A CHAR (A TYPED
3651	013250	104401	005202 4\$:	BHS	6\$			BR IF NOT
3652	013254	104401	001324	TYPE	,BUFFO			ECHO BAD INPUT
3653	013258	000754		TYPE	,SQUES			TYPE (') AND (CR), (LF)
3654	013262	023727	005202 000120 6\$:	BR	ASKSYS			GO ASK AGAIN
3655	013270	101367		CMP	BUFFO,#'P			SEE IF A CHAR > P TYPED
3656	013272	113737	005202 005500	BHI	4\$			BR IF YES
3657	013300	012700	005503	MOV8	BUFFO,SUBSYS			STORE CURRENT SUBSYSTEM NAME
3658	013304	122037	005500 8\$:	MOV	#DRVLST+1,RO			SET DRIVE LIST POINTER
3659	013310	001410		CMP8	(RO)+,SUBSYS			SEE IF A DRIVE LISTED FOR THIS SUBSYS
3660	013312	005200		BEQ	9\$			BR IF YES
3661	013314	020027	005543	INC	RO			INCR POINTER
3662	013320	103771		CMP	RO,#DRVLST+33.			SEE IF WHOLE LIST CHECKED YET
3663	013322	104401	010422	BLO	8\$			BR IF NOT YET
3664	013326	000137	013212	TYPE	,NOORLS			TYPE "NO DRIVES LISTED FOR SUBSYS"
3665	013332	162700	000002 9\$:	JMP	ASKSYS			GO ASK FOR A SUBSYS AGAIN
3666	013336	010037	005544	SUB	#2,RO			POINT TO LOW BYTE OF LIST ENTRY
3667				MOV	RO,DRVPTA			STORE POINTER
3668	013342	013746	003036					;OPEN RK06 UNIBUS ADDRESS FOR MODIFICATION
3669	013346	104401	007517	MOV	RKBAS,-(SP)			PUT OLD VALUE ON STACK
3670	013352	004737	020454	TYPE	RKBAOR			TYPE "RK06 BUS ADR = "
3671	013356	012637	003036	JSR	PC,GETPRM			TYPE OLD, GET NEW RKBAS VALUE
3672				MOV	(SP)+,RKBAS			STORE NEW VALUE
3673	013362	013746	003040					;OPEN RK06 VECTOR ADDRESS FOR MODIFICATION
3674	013366	104401	007540	MOV	RKVEC,-(SP)			PUT OLD VALUE ON STACK
3675	013372	004737	020454	TYPE	RKVAOR			TYPE "RK06 VEC ADR = "
3676	013376	012637	003040	JSR	PC,GETPRM			TYPE OLD, GET NEW RKVEC VALUE
3677				MOV	(SP)+,RKVEC			STORE NEW VALUE
3678	013402	013746	003042					;OPEN RK06 INTERRUPT HANDLER PRIORITY LEVEL FOR MODIFICATION
3679	013406	006316		MOV	RKPRI,-(SP)			GET OLD VALUE OF PRIORITY
3680	013410	006316		ASL	(SP)			GET IT INTO BITS 0-2
3681	013412	006316		ASL	(SP)			

M06

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 77
TEST 0 PARAMETER INPUT ROUTINES

SEQ 0076

```

3682 013414 000316 SWAB (SP)
3683 013416 104401 007560 TYPE RKPRTY ;TYPE "RK06 PRIORITY = "
3684 013422 004737 020454 JSR PC,GETPRM ;TYPE OLD, GET NEW RKPRI VALUE
3685 013426 012600 MOV (SP)+,RO
3686 013430 000027 000004 CMP RO,#4 ;SEE IF AT LEAST LEVEL 4
3687 013434 000414 BLT 12$ ;BR IF NEW VALUE TOO SMALL
3688 013436 020027 000007 CMP RO,#7 ;SEE IF LEVEL 7 OR LESS
3689 013442 000301 BGT 12$ ;BR IF NEW VALUE TOO LARGE
3690 013444 000300 SWAB RO ;GET PRIORITY INTO BITS 5-7
3691 013446 006200 ASR RO
3692 013450 006200 ASR RO
3693 013452 006200 ASR RO
3694 013454 010037 003042 MOV RO,RKPRI ;STORE NEW VALUE
3695 013460 104401 001325 TYPE $CRLF
3696 013464 000405 BR DRVTST
3697 013466 104401 005202 12$: TYPE ,BUFFO ;ECHO BAD INPUT
3698 013472 104401 001324 TYPE $QUES
3699 013476 000741 BR 11$ ;GO ASK AGAIN
3700
3701 013500 113737 003145 010101 DRVTST: MOVB PASSNO,SRPASS+18. ;SET PASS NO. FOR PRINTOUT
3702 013506 104401 010057 TYPE $CRPHSS ;TYPE "STARTING PASS X"
3703 013512 122737 000061 003145 CMPB #'1,PASSNO ;SEE IF PASS 1
3704 013520 001002 BNE 2$ ;BR IF NOT PASS 1
3705 013522 000137 014076 JMP TST1 ;START PASS 1
3706 013526 112737 000004 001102 2$: MOVB #4,$TSTNM ;SET TEST NUMBER = 4
3707 013534 000137 014574 JMP TST5 ;START PASS 2
3708
3709
3710 ;ADR 200 START - COMPILE SCRATCH LIST OF DRIVES TO SIZE
3711 013540 005000 DFLTDR: CLR RO ;CLEAR DRIVE NUMBER
3712 013542 013703 001406 MOV $DEVM,R3 ;GET APT DEVICE MAP
3713 013546 132737 000200 001351 BITB #BIT7,$ENVM ;SEE IF NO SIZING
3714 013554 001002 BNE 1$ ;BR IF NO SIZING
3715 013556 012703 000377 MOV #377,R3 ;SET UP FOR SIZING
3716 013562 012701 000001 1$: MOV #BIT0,R1 ;SET BIT POINTER
3717 013566 105060 005546 2$: CLRB SCALST(RO) ;INITIALIZE DRIVE ENTRY
3718 013572 030103 BIT R1,R3 ;SEE IF THIS DRIVE IS REQUESTED
3719 013574 001405 BEQ 4$ ;BR IF NOT
3720 013576 110060 005546 MOVB RO,SCALST(RO) ;MARK DRIVE IN SCRATCH LIST
3721 013602 152760 000060 005546 BISB #'0,SCALST(RO) ;CONVERT NO. TO ASCII
3722 013610 006301 4$: ASL R1 ;SHIFT BIT POINTER
3723 013612 005200 INC RO ;INCREMENT DRIVE NUMBER
3724 013614 022700 000010 CMP #10,RO ;SEE IF DONE MARKING ALL DRIVES
3725 013620 001362 BNE 2$ ;BR IF NOT DONE YET
3726 013622 132737 000200 001351 BITB #BIT7,$ENVM ;SEE IF PROGRAM SHOULD SIZE
3727 013630 001416 BEQ 8$ ;BR IF YES
3728 013632 012704 005502 10$: MOV #DRVLST,R4 ;GET DRIVE LIST ADDRESS
3729 013636 113703 005500 MOVB SUBSYS,R3 ;GET SUBSYS NAME IN R3
3730 013642 010401 MOV R4,R1 ;GET DRIVE LIST ADRS
3731 013644 010437 005544 MOV R4,DRVPTN ;SET POINTER TO FIRST DRIVE TO TEST
3732 013650 012700 000021 MOV #17,RO ;SET COUNTER
3733 013654 005021 11$: CLR (R1)+ ;CLEAR A DRIVE LIST WORD
3734 013656 005300 DEC RO ;DECREMENT COUNTER
3735 013660 001375 BNE 11$ ;BR IF NOT DONE YET
3736 013662 000137 012764 JMP ALLSY3 ;GO MOVE SCRATCH LIST INTO LIST
3737 ;REMOVE NON-EXISTENT DRIVES FROM SCRATCH LIST

```

N06

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 78
TEST 0 PARAMETER INPUT ROUTINES

SEQ 0077

```

3738 013666 005000      85: CLR      R0          ;CLEAR DRIVE NUMBER
3739 013670 105760 005546 125: TSTB   SCRLST(R0) ;SEE IF THIS DRIVE IS MARKED IN SCRATCH LIST
3740 013674 001467      BEQ      145         ;BR IF NOT MARKED
3741 013676 010037 005420      MOV      R0,DRIVE    ;SET DRIVE NUMBER FOR SCNDRV
3742 013702 004737 020616      JSR      PC,INITSS   ;CLEAR DRIVER AND INIT S.S.
3743 013706 042712 000100      BIC      #IE,(R2)    ;DISABLE RK06 INTERRUPT
3744 013712 113762 005420 000010      MOVVB   DRIVE,RKCS2(R2) ;SET DRIVE NO. IN RKCS2
3745 013720 012712 000001      MOV      #BIT0,(R2)  ;SELECT THE DRIVE
3746 013724 005037 005434      CLR     SCRACH       ;CLEAR STALL COUNTER
3747 013730 005237 005434      155: INC     SCRACH       ;INCREMENT STALL COUNTER
3748 013734 001375      BNE     155          ;STALL FOR SEVERAL MILLI-SEC
3749 013736 032762 001000 000010      BIT     #MDS,RKCS2(R2) ;SEE IF MDS ERROR
3750 013744 001417      BEQ     185          ;BR IF NOT
3751 013746 112765 000101 000001      MOVVB   #SELDRV,P.CMND(R5) ;SET CMND FOR ERROR REPORT
3752 013754 011265 000016      MOV     (R2),P.CS1(R5) ;GET RKCS1 FOR REPORT
3753 013760 004737 037000      JSR     PC,I.CST1    ;GET OTHER RK611 REGS FOR REPORT
3754 013764 004737 030360      JSR     PC,REPSUP    ;SET UP ERROR REPORT
3755 013770 104052      ERROR   52          ;REPORT MDS ERROR
3756 013772 052737 000200 005414      BIS     #ABORT,RECODE ;SET ABORT FLAG
3757 014000 000137 032472      JMP     ALLTRM       ;GO ABORT TESTING
3758 014004 004737 020616      185: JSR     PC,INITSS   ;INIT THE S.S.
3759 014010 112765 000141 000001      MOVVB   #ROSTAT,P.CMND(R5) ;SET READ STATUS COMMAND
3760 014016 012737 020126 003046      MOV     #NEDHDL,A.ABNL ;SET NED ABNORMAL RETURN ADR
3761 014024 012737 000377 005466      MOV     #377,NEWON   ;INIT ONLINE INDICATOR
3762 014032 004737 027302      JSR     PC,DRVCAL    ;READ DRIVE STATUS
3763 014036 012737 030640 003046      MOV     #ERRHDL,A.ABNL ;RESTORE ABNORMAL RETURN ADDRESS
3764 014044 022737 000377 005466      CMP     #377,NEWON   ;SEE IF THIS DRIVE EXISTS
3765 014052 001006      BNE     165          ;BR IF NO
3766 014054 005200      145: INC     R0          ;RETURN HERE IF DRIVE IS USEABLE
3767 014056 022700 000010      CMP     #10,R0       ;SEE IF DONE CHECKING LIST
3768 014062 001302      BNE     125          ;BR IF NOT DONE YET
3769 014064 000137 013632      JMP     105          ;GO BACK TO LIST SELECTED DRIVES
3770 014070 105060 005546      165: CLRB   SCRLST(R0)   ;REMOVE INVALID DRIVE FROM LIST
3771 014074 000767      BR     145          ;CONTINUE CHECKING THE LIST

```

```

3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789 014076 000004
3790 014100 012737 000001 001334
3791 014106 004737 021424
3792 014112 004737 020616
3793 014116 004737 020422

```

```

*****
*TEST 1 MOUNTING OF TEST CARTRIDGE FOR PASS 1
*
* THE PROGRAM TYPES "STARTING PASS 1". THEN, THE OPERATOR
* MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS,
* AS DIRECTED BY THE PROGRAM. IT THEN CHECKS THIS DRIVE
* FOR VALID STATUS, AND READS THE HEADER ON SECTOR 0, TO
* DETERMINE THE FORMAT. THE FORMAT IS STORED IN "FORMAT".
* ALSO, THE PROGRAM READS AND STORES THE BAD SECTOR FILES, AND
* IT TYPES THE DRIVE AND CARTRIDGE SERIAL NUMBERS.
*****
*ST1: SCOPE
MOV #1,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
JSR PC,SETUP ;:SET UP FOR LOOP ON ERROR
JSR PC,INITSS ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
JSR PC,PREPKB ;:PREPARE FOR POSSIBLE KBD INPUT

```


3794 014122 004737 021472

JSR PC, MTCART

3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849

014126 000004
014130 012737 000002 001334
014136 004737 021424
014142 004737 020616
014146 004737 020422
014152 112737 000001 003117
014160 004737 023640
014164 113700 003144
014170 006300
014172 006300
014174 016065 005556 000002
014202 062700 000002
014206 116065 005556 000004
014214 012765 052476 000010
014222 012765 177400 000012
014230 112765 000123 000001
014236 004737 027302
014242 112765 000131 000001
014250 004737 027302
014254 105265 000005
014260 126527 000005 000002
014266 101760

*TEST 2 BASIC READ/WRITE DATA TEST

* THE PROGRAM PERFORMS A WRITE AND WRITE CHECK OPERATION
* USING A "WORST CASE" DATA PATTERN, AT THE APPROPRIATE SECTOR
* FOR THIS DRIVE (FROM TABLE A) ON ALL SURFACES. THE PURPOSE
* OF THIS OPERATION IS TO VERIFY THE BASIC READ/WRITE
* CAPABILITY OF THE DRIVE ON PASS 1. THE ENTIRE SECTOR
* IS WRITTEN WITH THE REPETITION OF THE DATA PATTERN SHOWN
* IN TABLE B.

ST2: SCOPE
MOV #2, STESTN ; SET TEST NUMBER IN APT MAIL BOX
JSR PC, SETUP ; SET UP FOR LOOP ON ERROR
JSR PC, INITSS ; INITIALIZE DRIVER PARAMS AND SUB-SYS
JSR PC, PREPKB ; PREPARE FOR POSSIBLE KBD INPUT
MOVB #1, TSTING ; ALLOW TTY ESCAPE
; LOAD RMBUF WITH DATA PATTERN
JSR PC, LODSEC
; SET UP PARAMETERS
MOVB LOGDRV, RO ; GET LOGICAL DRIVE NUMBER
ASL RO ; CONVERT IT TO INDEX
RO
MOV TABLEA(RO), P.CYLN(R5) ; GET CYL # FOR THIS DRIVE
ADD #2, RO ; INCREMENT INDEX
MOVB TABLEA(RO), P.SECT(R5) ; GET SECTOR # FOR THIS DRIVE
MOV #R3UF, P.BALO(R5) ; SET BUS ADDRESS
MOV #-400, P.WC(R5) ; SET WORD COUNT FOR 1 SECTOR
; WRITE THE DATA
BS: MOVB #WRDATA, P.CMND(R5) ; SET WRITE COMMAND
JSR PC, DRVCAL ; WRITE THE SECTOR
; PERFORM WRITE CHECK OF SECTOR
MOVB #WRCHK, P.CMND(R5) ; SET WRITE CHECK COMMAND
JSR PC, DRVCAL ; PERFORM THE WRITE CHECK
; INCREMENT TRACK ADDRESS
INCB P.TRCK(R5) ; INCREMENT TRACK NO.
CMPB P.TRCK(R5), #2 ; SEE IF ALL SURFACES WRITTEN YET
BLOS BS ; BR IF NOT YET

*TEST 3 WRITE OVRWRT AND COMPAT CYL BLOCKS FOR CURRENT DRIVE
* ALL SECTORS ARE WRITTEN FOR THE CURRENT DRIVE WITHIN THE
* CYLINDER BLOCKS IN TABLEC ON ALL SURFACES USING A SINGLE
* REPEATED WORD OF THE PSEUDO-RANDOM DATA PATTERN IN TABLEG. LOGICAL DRIVE 0
* USES WORD 0, LOGICAL DRIVE 14 USES WORD 14, ETC. THESE CYLINDER
* BLOCKS ARE THE OVERWRITE AND COMPATIBILITY TEST DATA TO

```

3850 ;* BE USED IN PASS 2.
3851 ;*****
3852 014270 000004 TST3: SCOPE
3853 014272 012737 000003 001334 MOV #3,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
3854 014300 004737 021424 JSR PC,SETUP ;SET UP FOR LOOP ON ERROR
3855 014304 004737 020616 JSR PC,INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
3856 014310 004737 020422 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
3857 014314 004737 022224 JSR PC,INTBLD ;LOAD INITIAL VALUES INTO TABLED
3858 014320 005000 CLR RO ;INITIALIZE TABLEC INDEX TO 0
3859 014322 016065 005716 000002 4$: MOV TABLEC(RO),P.CYLN(R5) ;GET START OVRWRT CYL FROM TABLEC
3860 014330 004737 022470 JSR PC,WRTBLK ;WRITE THIS OVRWRT CYL BLK ON ALL SURF'S
3861 014334 062700 000002 ADD #2,RO ;INCREMENT TABLEC POINTER
3862 014340 016065 005716 000002 MOV TABLEC(RO),P.CYLN(R5) ;GET START COMPAT CYL FROM TABLEC
3863 014346 004737 022470 JSR PC,WRTBLK ;WRITE THIS COMPAT CYL BLK ON ALL SURF'S
3864 014352 062700 000002 ADD #2,RO ;INCREMENT TABLEC POINTER
3865 014356 020027 000030 CMP RO,#30 ;SEE IF ON LAST CURRENT ZONE YET
3866 014362 002003 BGE B$ ;BR IF YES
3867 014364 004737 022404 JSR PC,ROTBLD ;ROTATE TABLED SEVERAL SECTORS
3868 014370 000754 BR 4$ ;CONTINUE WRITING BLOCKS
3869 014372 004737 022224 8$: JSR PC,INTBLD ;RE-LOAD INITIAL VALUES INTO TABLED
3870 014376 016065 005716 000002 MOV TABLEC(RO),P.CYLN(R5) ;GET START OVRWRT CYL FROM TABLEC
3871 014404 004737 022470 JSR PC,WRTBLK ;WRITE THIS OVRWRT CYL BLK ON ALL SURF'S

```

```

3872
3873
3874
3875 ;*****
3876 ;TEST 4 DISMOUNTING OF TEST CARTRIDGE IN PASS 1
3877 ;
3878 ; THE PROGRAM DIRECTS THE OPERATOR TO UNLOAD THE DRIVE CURRENTLY
3879 ; UNDER TEST, AND TO REMOVE THE TEST PACK. THEN, THE PROGRAM
3880 ; DOES ONE OF 4 THINGS :
3881 ; (1) IF THERE IS ANOTHER DRIVE TO TEST ON THIS SUBSYSTEM, THE
3882 ; PROGRAM JUMPS TO TEST 1 FOR THE NEXT DRIVE.
3883 ; (2) IF THERE IS ONLY ONE SUBSYSTEM, (FROM 200 START), AND
3884 ; IF PASS 1 HAS BEEN COMPLETED ON ALL DRIVES, THE PROGRAM STARTS
3885 ; PASS 2 ON THE FIRST DRIVE, BY JUMPING TO TEST 5.
3886 ; (3) IF THERE IS ANOTHER SUBSYSTEM TO PERFORM PASS 1 ON, THE PROGRAM
3887 ; TELLS THE OPERATOR TO RESTART AT ADDRESS 204, AND THEN IT HALTS.
3888 ; (4) IF THERE ARE NO MORE DRIVES TO TEST IN PASS 1 ON ANY
3889 ; OTHER SUBSYSTEM, THE PROGRAM TELLS THE OPERATOR TO
3890 ; RESTART AT ADDRESS 220 FOR PASS 2 ON THE NEXT SUBSYSTEM, AND
3891 ; THEN IT HALTS.
3892 ;
3893 ;*****

```

```

3894 014410 000004 TST4: SCOPE
3895 014412 012737 000004 001334 MOV #4,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
3896 014420 004737 021424 JSR PC,SETUP ;SET UP FOR LOOP ON ERROR
3897 014424 004737 020616 JSR PC,INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
3898 014430 004737 020422 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
3899 014434 105037 003117 CLR B ;DON'T ALLOW TTY ESCAPE
3900 014440 004737 022054 JSR PC,DMCART ;DIRECT OPERATOR TO DISMOUNT PACK
3901 014444 062737 000002 005544 ADD #2,DRVPTN ;INCREMENT DRIVE LIST POINTER
3902 014452 013700 005544 MOV DRVPTN,RO
3903 014456 005710 TST (RO) ;SEE IF ALL DONE WITH PASS 1 YET
3904 014460 001420 BEQ B$ ;BR IF DONE WITH PASS 1
3905 014462 126037 000001 005500 CMPB I(RO),SUBSYS ;SEE IF DONE WITH THIS SUBSYS YET

```

```

3906 014470 001004 BNE 48 ;BR IF DONE WITH THIS SUBSYS
3907 014472 105037 001102 CLR8 $TSTNM ;CLEAR TEST NUMBER
3908 014476 000137 014076 JMP TST1 ;DO PASS 1 ON NEXT DRIVE OF SUBSYS
3909 014502 116037 000001 010267 4$: MOV8 1(R0),STR204+39. ;PUT SUBSYS NAME INTO MSG
3910 014510 104401 010220 5$: TYPE ,STR204 ;TYPE RESTART MSG FOR PASS 1 ON NEXT SUBSYS
3911 014514 000000 6$: HALT ;HALT THE PROCESSOR SO THAT
3912 014516 000137 000204 JMP 204 ;OPERATOR CAN RESTART AT PROPER ADRS
3913 014522 012737 005502 005544 8$: MOV #DRVLST,DRVPTR ;RE-INIT DRIVE LIST POINTER
3914 014530 105737 003116 TSTB MOFLAG ;SEE IF 200 START
3915 014534 001007 BNE 128 ;BR IF NOT
3916 014536 112737 000062 010101 MOV8 #'2,SRPASS+18. ;SET PASS NO. = 2 FOR PRINTOUT
3917 014544 104401 010057 TYPE SRPASS ;TYPE "STARTING PASS 2"
3918 014550 000137 014574 JMP $TST5 ;GO START PASS 2
3919 014554 112737 000101 010344 12$: MOV8 #'A,STR220+39. ;PUT SUBSYS NAME INTO MSG
3920 014562 104401 010275 TYPE ,STR220 ;TYPE RESTART MSG FOR PASS 2 ON FIRST SUBSYS
3921 014566 000000 HALT ;HALT PROCESSOR SO THAT OPERATOR CAN RESTART
3922 014570 000137 000220 JMP 220 ; AT THE PROPER ADRS

```

```

*****
*TEST 5 MOUNTING OF TEST CARTRIDGE FOR PASS 2

```

```

* THE PROGRAM TYPES "STARTING PASS 2". THEN, THE OPERATOR
* MOUNTS THE PACK ON THIS DRIVE AND MANUALLY LOADS THE HEADS,
* AS DIRECTED BY THE PROGRAM. IT THEN CHECKS THIS DRIVE
* FOR VALID STATUS, AND READS THE HEADER ON SECTOR 0, TO
* DETERMINE THE FORMAT. THE FORMAT IS STORED IN "FORMAT".
* ALSO, THE PROGRAM READS AND STORES THE BAD SECTOR FILES, AND
* IT TYPES THE DRIVE AND CARTRIDGE SERIAL NUMBERS (IF STARTED
* AT ADDRESS 204 OR 220).

```

```

*****
*TESTS: SCOPE
3939 014574 000004 MOV #5,$TSTNM ;SET TEST NUMBER IN RPT MAIL BOX
3940 014576 012737 000005 001334 JSR PC,SETUP ;SET UP FOR LOOP ON ERROR
3941 014604 004737 021424 JSR PC,INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
3942 014610 004737 020616 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
3943 014614 004737 020422 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
3944 014620 112737 000062 003145 MOV8 #'2,PASSNO ;SET PASS NO. = 2
3945 014626 004737 021472 JSR PC,ATCART

```

```

*****
*TEST 6 OVERWRITE TEST

```

```

* THE PROGRAM PROCEEDS TO TEST THIS DRIVE'S OVERWRITE CAPABILITY.
* FIRST, THE APPROPRIATE CYLINDERS IN TABLE E FOR THIS DRIVE
* ARE OVERWRITTEN, ON EACH SURFACE. THE DATA USED IS A
* REPETITION OF A SINGLE WORD OF THE PATTERN IN TABLE G.
* DRIVE 0 USES WORD 0, DRIVE 1 USES WORD 1, DRIVE 10 USES
* WORD 10, ETC.
* THEN, EACH CYLINDER OVERWRITTEN IS READ BACK BY THIS DRIVE
* WITH THE FOLLOWING RANGE OF OFFSET VALUES : 100, 200, 300,
* 400, 500, 600, 700, 800, 900, 1000, 1100, 1200 MICRO-IN., IN BOTH
* THE (+) AND (-) DIRECTIONS.

```

3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961

T6 OVERWRITE TEST

```

3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973 014632 000004
3974 014634 012737 000006 001334
3975 014642 004737 021424
3976 014646 004737 020616
3977 014652 004737 020422
3978 014656 112737 000001 003117
3979 014664 105037 003137
3980
3981 014670 112765 000117 000001
3982 014676 105065 000002
3983 014702 004737 027302
3984 014706 012703 000007
3985 014712 112765 000123 000001
3986 014720 113700 003144
3987 014724 010065 000002
3988 014730 006300
3989 014732 062700 006350
3990 014736 011037 052476
3991 014742 012765 052476 000010
3992 014750 052765 100000 000014
3993 014756 012765 137000 000012
3994 014764 105737 003124
3995 014770 001403
3996 014772 012765 142000 000012
3997 015000 004737 026634 65:
3998 015004 004737 026724
3999 015010 062765 000100 000002
4000 015016 005303
4001 015020 001367
4002
4003 015022 112737 000001 003133
4004 015030 012700 006450
4005 015034 005020
4006 015036 020027 006750
4007 015042 103774
4008 015044 012703 000007
4009 015050 113700 003144
4010 015054 010065 000002
4011 015060 004737 022224 115:
4012 015064 004737 016726 125:
4013 015070 004737 017272
4014 015074 062765 000100 000002
4015 015102 112737 000001 003137
4016 015110 005303
4017 015112 004737 022404

```

```

: * THE PROGRAM SCANS FOR DATA CHECK ERRORS DURING THIS READ, AND IF
: * ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA HAS NOT
: * BEEN CORRECTLY OVERWRITTEN, AND A SCORE FOR THAT DRIVE IS
: * DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR,
: * WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE ABOVE
: * OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT,
: * WHILE THE CURRENT DRIVE IS PERFORMING THE OVERWRITES.
: * FOR EACH TRACK (0,1,2). SCORES ARE AVERAGED FOR EACH OFFSET
: * DIRECTION, OVER ALL CYLINDERS TESTED.
: *
: *****
TST6: SCOPE
MOV #6,STESTN ; SET TEST NUMBER IN APT MAIL BOX
JSR PC,SETUP ; SET UP FOR LOOP ON FLOOR
JSR PC,INITSS ; INITIALIZE DRIVER PHONES AND SUB-SYS
JSR PC,PREPKB ; PREPARE FOR POSSIBLE KBD INPUT
MOV #1,TSTING ; ALLOW TTY ESCAPE
CLRB DIF100 ; CLEAR "CYL DIF = 100" FLAG
; OVERWRITE THE APPROPRIATE CYLS FOR THIS DRIVE
MOV #SEEK.P.COMD(RS) ; SET SEEK COMMAND
CLRB P.CYLN(RS) ; SET CYL = 0
JSR PC,DRVCAL ; SEEK TO CYL 0
MOV #7,R3 ; SET COUNTER = 7
MOV #WRDATA.P.COMD(RS) ; SET WRITE DATA COMMAND
MOV LOGDRV,RO ; GET LOGICAL DRIVE NUMBER
MOV RO,P.CYLN(RS) ; SET STARTING OVERWRITE CYL FOR THIS DRIVE
ASL RO ; GET DATA PATTERN INDEX
ADD #TABLEG,RO ; GET PATTERN WORD ADRS
MOV (RO),RBUF ; PUT DATA WORD INTO BUFFER
MOV #RBUF,P.BALO(RS) ; SET BUS ADDRESS
BIS #OTBATT,P.PRST(RS) ; SET BUS ADRS INCREMENT INHIBIT
MOV #16896.,P.WC(RS) ; SET 22-SECTOR WORD COUNT
TSTB FORMAT ; DETERMINE THE FORMAT
BEQ 65 ; BR IF 22 SECTORS
MOV #15360.,P.WC(RS) ; SET 20-SECTOR WORD COUNT
65: JSR PC,SVPRMS ; SAVE PARAMETERS OF TRANSFER
JSR PC,TRANSR ; WRITE THE DATA
ADD #100,P.CYLN(RS) ; INCR THE CYL NO.
DEC R3 ; SEE IF DONE WITH ALL CYLS YET
BNE 65 ; BR IF NOT YET
; READ OVERWRITE CYLS WITH RANGE OF OFFSETS
MOV #1,TSTTYP ; SET INDICATOR FOR OVRWRT TEST
MOV #NOSCRO,RO ; GET STARTING ADRS OF OVRWRT SCORES
105: CLR (RO)+ ; INIT SCORE TO 0
CMP RO,#NOSCRO+192. ; SEE IF ALL SCORES CLEARED YET
BLO 105 ; BR IF NOT DONE YET
MOV #7,R3 ; SET CYL CNTR = 7
MOV LOGDRV,RO ; GET LOGICAL DRIVE NO.
MOV RO,P.CYLN(RS) ; GET START OVRWRT CYL FOR THIS DRIVE
115: JSR PC,INTBLD ; INIT TABLED
125: JSR PC,REDOFS ; DO READS WITH OFFSETS ON THIS CYL
JSR PC,ADSCOR ; ADD SCRATCH SCORES TO SUM
ADD #100,P.CYLN(RS) ; INCR CYL NO.
MOV #1,DIF100 ; SET "CYL DIF = 100" FLAG
DEC R3 ; DECR CYL CNTR
JSR PC,ROTBLD ; ROTATE TABLED FOR NEXT CURRENT ZONE

```

4018	015116	020327	000001
4019	015122	003360	
4020	015124	001755	
4021	015126	004737	017452
4022	015132	105037	003133

```

CMP      R3, #1      ;SEE IF ON LAST ZONE YET
BGT      12$         ;BR IF NOT YET
BEQ      11$         ;BR IF ON LAST CYL
JSR      PC, AVSCOR ;COMPUTE AVERAGE OVERWRITE SCORES
CLRB    TSTYP       ;CLEAR OVRWRT TEST INDICATOR

```

4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044

```

*****
TEST 7      DRIVE SELF-TEST
*****
THE PROGRAM EVALUATES THE DRIVE'S ABILITY TO WRITE AND READ
ITS OWN DATA, AT VARIOUS ANGULAR POSITIONS ON THE PACK. FIRST,
SECTORS 4, 11, 16, AND 23 OF THE APPROPRIATE CYLINDERS SHOWN
IN TABLE F FOR THIS DRIVE ARE WRITTEN WITH THE DATA PATTERN
SHOWN IN TABLE F, FOR ALL SURFACES. THEN, THE SECTORS ARE READ
WITH THE FOLLOWING OFFSETS APPLIED : 100, 200, 300, 400, 500, 600,
700, 800, 900, 1000, 1100, 1200 MICRO-IN. IN (+) AND (-) DIRECTIONS.
THE PROGRAM SCANS FOR DATA CHECK ERRORS DURING EACH READ, AND IT
COMPUTES A SCORE WHICH IS PROPORTIONAL TO THE FAILING OFFSET
MAGNITUDE. THEN, THE SCORES FOR ALL SECTORS READ ARE AVERAGED,
TO COME UP WITH A DRIVE SELF-TEST SCORE FOR EACH OFFSET
DIRECTION, FOR EACH SURFACE.
THIS SCORE IS SAVED FOR LATER USE, TO BECOME THE STANDARD
FOR THE COMPATIBILITY DATA READS WHICH ARE TO FOLLOW.

```

4045	015136	000004		
4046	015140	012737	000007	001334
4047	015146	004737	021424	
4048	015152	004737	020616	
4049	015156	004737	020422	
4050	015162	005037	007250	
4051	015166	005037	007252	
4052	015172	005037	007254	
4053	015176	005037	007256	
4054	015202	005037	007260	
4055	015206	005037	007262	
4056	015212	105037	003137	
4057	015216	012703	000006	
4058	015222	113700	003144	
4059	015226	062700	000060	
4060	015232	012765	052476	000010
4061	015240	004737	023440	
4062	015244	012765	177400	000012
4063	015252	112765	000004	000004
4064	015260	105065	000005	
4065	015264	105037	007270	
4066	015270	105037	007310	
4067	015274	105037	007330	
4068	015300	105037	007350	
4069	015304	105037	007370	
4070	015310	105037	007410	
4071	015314	105037	003133	
4072	015320	005065	000002	
4073	015324	112765	000117	000001

```

*****
TST7:  SCOPE
MOV      #7, $TESTN ;SET TEST NUMBER IN APT MAIL BOX
JSR      PC, SETUP  ;SET UP FOR LOOP ON ERROR
JSR      PC, INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
JSR      PC, PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
CLRB    NSCOR0      ;INIT SELF-TEST SCORES TO 0
CLRB    NSCOR1
CLRB    NSCOR2
CLRB    PSSCOR0
CLRB    PSSCOR1
CLRB    PSSCOR2
CLRB    DIF100      ;CLEAR "CYL DIF = 100" FLAG
MOV      #6, R3     ;SET CYL COUNTER = 6
MOVB    LOGDRV, R0 ;GET LOGICAL DRIVE NO.
ADD     #60, R0     ;COMPUTE STARTING CYL NO.
MOV      #RWBUFF, P.BALO(R5) ;SET BUS ADDR
JSR      PC, LOOSEC ;LOAD R/W BUF WITH DATA PATTERN
MOV      #8-256, P.WC(R5) ;SET WORD COUNT FOR 1 SECTOR
MOVB    #4, P.SECT(R5) ;SET SECTOR = 4
CLRB    P.TRACK(R5) ;SET TRACK = 0
CLRB    NSCOR0      ;INIT SCRATCH SCORES TO 0
CLRB    NSCOR1
CLRB    NSCOR2
CLRB    PSCOR0
CLRB    PSCOR1
CLRB    PSCOR2
CLRB    TSTYP       ;CLEAR OFFSET TEST INDICATOR
CLRB    P.CYLN(R5) ;SET CYL = 0
MOVB    #SEEK, P.CMND(R5) ;SET SEEK COMMAND

```

```

4074 015332 004737 027302
4075 015336 010065 000002
4076 015342 112765 000123 000001
4077 015350 004737 027302
4078 015354 112737 000003 003133
4079 015362 004737 016726
4080 015366 105265 000005
4081 015372 126527 000005 000002
4082 015400 101745
4083 015402 004737 017272
4084 015406 062765 000005 000004
4085 015414 126527 000004 000023
4086 015422 101716
4087 015424 062700 000100
4088 015430 005303
4089 015432 001307
4090 015434 004737 017452
4091 015440 105037 003133

```

```

JSR PC,DRVCL ;PERFORM SEEK TO 0
MOV RO,P.CYLN(RS) ;SET CURRENT CYLINDER NUMBER
MOVW #WADATA,P.CMND(RS) ;SET WRITE DATA COMMAND
JSR PC,DRVCL ;WRITE A SECTOR
MOVW #3,TSTTYP ;SET INDICATOR FOR SELF-TEST
JSR PC,REDOFS ;READ SECTOR WITH RANGE OF OFFSETS
INCB P.TRCK(RS) ;INCREMENT TRACK
CMPB P.TRCK(RS),#2 ;SEE IF ALL TRACKS DONE ON THIS CYL
BLOS 6$ ;BR IF NOT YET
JSR PC,ADSCOR ;ADD SCRATCH SCORES TO SUMS
ADD #5,P.SECT(RS) ;INCREMENT SECTOR BY 5
CMPB P.SECT(RS),#23 ;SEE IF ALL SECTORS DONE ON THIS TRACK
BLOS 4$ ;BR IF NOT YET
ADD #100,RO ;INCR CYL NO. BY 100
DEC R3 ;DECR CYL CNTR
BNE 3$ ;BR IF NOT DONE WITH ALL CYLS YET
JSR PC,AVSCOR ;COMPUTE AVG SELF-TEST SCORES
CLRB TSTTYP ;CLEAR OVRHRT TEST INDICATOR

```

TEST 10 COMPATIBILITY DATA READ TEST

HAVING ESTABLISHED A SELF-TEST SCORE FOR THIS DRIVE, THE PROGRAM PROCEEDS TO PERFORM THE COMPATIBILITY DATA READS OF THE PATTERNS WRITTEN BY ALL THE DRIVES IN EACH CYLINDER BLOCK (ON EACH SURFACE). EACH COMPATIBILITY CYLINDER BLOCK SHOWN IN TABLE C IS READ WITH THE FOLLOWING OFFSET VALUES: 100,200,300,400,500,600,700,800,900,1000,1100,1200 MICRO-IN., IN (+) AND (-) DIRECTIONS. THE PROGRAM SCORES FOR READ ERRORS DURING EACH READ, AND IF ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA WAS BEING READ AT THAT INSTANT AND A SCORE FOR THAT DRIVE IS DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR, WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT, WHILE THE CURRENT DRIVE IS READING THE COMPATIBILITY DATA. THEN, EACH SCORE IS APPROPRIATELY ADJUSTED TO REFLECT THE SELF-TEST SCORE FOR THE CURRENT DRIVE ON THAT SURFACE. THE SCORES ARE THEN AVERAGED OVER ALL CYLINDER BLOCKS. EACH SCORE IS PROPORTIONAL TO THE CAPABILITY OF THE CURRENT DRIVE TO SUCCESSFULLY READ THE DATA WRITTEN BY ONE OF THE OTHER DRIVES, AND SCORES ARE COMPUTED SEPARATELY FOR EACH SURFACE (TRACK). THUS, THE SCORES REVEAL WHICH DRIVES ARE INVOLVED IN A SITUATION IN WHICH A PARTICULAR DRIVE HAS DIFFICULTY IN READING THE DATA OF ONE OR SEVERAL OTHER DRIVES.

21.

```

4122 015444 000004
4123 015446 012737 000010 001334
4124 015454 004737 021424
4125 015460 004737 020616
4126 015464 004737 020422
4127 015470 112737 000002 003133
4128 015476 105037 003137
4129 015502 012700 006750

```

```

*****
TST10: SCOPE
MOV #10,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
JSR PC,SETUP ;SET UP FOR LOOP ON ERROR
JSR PC,INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
MOVW #2,TSTTYP ;SET INDIC FOR READ TEST
CLRB DIF100 ;CLEAR "CYL DIF = 100" FLAG
MOV #NCSCRO,RO ;GET ADRS OF COMPAT READ SCORES

```

H07

```

4130 015506 005020 65: CLR (R0)+ ;INIT A SCORE TO 0
4131 015510 020027 007250 CMP R0,#NCSCRO+192. ;SEE IF ALL SCORES CLEARED YET
4132 015514 103774 BLO 65 ;BR IF NOT YET
4133 015516 012703 000006 MOV #6,R3 ;SET CYL BLK CNTR = 6
4134 015522 004737 022224 JSR PC,INTBLD ;INIT TABLE D
4135 015526 012701 000060 MOV #60,R1 ;SET START CYL = 60
4136 015532 012765 052476 000010 MOV #R0UF,P.BALO(R5) ;SET BUS ADRS
4137 015540 052765 100000 000014 BIS #OTBAII,P.PRST(R5) ;SET BUS ADRS INCR INHIBIT
4138 015546 012765 137000 000012 MOV #16896.,P.WC(R5) ;SET 22-SECTOR WORD COUNT
4139 015554 105737 003124 TSTB FCR,AT ;TEST THE FORMAT
4140 015560 001403 BEQ 105 ;BR IF 22 SECTORS
4141 015562 012765 142000 000012 MOV #15360.,P.WC(R5) ;SET 20-SECTOR WORD COUNT
4142 015570 012700 000020 105: MOV #16.,R0 ;INIT CYL CNTR = 16.
4143 015574 010165 000002 MOV R1,P.CYLN(R5) ;SET CYL NO.
4144 015600 004737 016726 125: JSR PC,REDOFS ;DO READS WITH OFSTS ON THIS CYL
4145 015604 004737 017272 JSR PC,ADSCOR ;ADD SCRATCH SCORES TO SUM
4146 015610 005265 000002 INC P.CYLN(R5) ;INCR CYL NO.
4147 015614 005300 DEC R0 ;SEE IF DONE WITH THIS CYL BLK
4148 015616 001370 BNE 125 ;BR IF NOT YET
4149 015620 004737 022404 JSR PC,R0*BLD ;ROTATE TABLED FOR NEXT CURRENT ZONE
4150 015624 062701 000100 ADD #100,R1 ;INCR CYL BY 100
4151 015630 112737 000001 003137 MOVB #1,DIF100 ;SET "CYL DIF = 100" FLAG
4152 015636 005303 DEC R3 ;SEE IF DONE WITH ALL CYL BLKS
4153 015640 001353 BNE 105 ;BR IF NOT YET
4154 015642 004737 017452 JSR PC,AVSCOR ;COMPUTE AVERAGES OF READ SCORES
4155 015646 105037 003133 CLRB TS1TYP ;CLEAR OFST TEST INDICATOR
4156 ;ADJUST COMPAT READ SCORES (- OFSTS) BY SELF-TEST SCORES
4157 015652 012701 006750 MOV #NCSCRO,R1 ;SET COMPAT SCORE POINTER
4158 015656 012703 007250 MOV #NSSCRO,R3 ;SET SELF-TEST SCORE POINTER
4159 015662 012700 005502 145: MOV #DRVLST,R0 ;SET DRIVE LIST POINTER
4160 015666 012737 000014 001304 MOV #12.,$TMP11 ;COMPUTE DIFFERENCE OF 12 (DEC) AND
; SELF-TEST SCORE FOR THIS TRACK
4161
4162 015674 16737 001304 SUB (R3)+,$TMP11
4163 015700 010104 MOV R1,R4 ;GET READ SCORE POINTER IN R4
4164 015702 005720 155: TST (R0)+ ;SEE IF A DRIVE LISTED HERE
4165 015704 001417 BEQ 185 ;BR IF NOT
4166 015706 013737 001304 001306 MOV $TMP11,$TMP12 ;PUT DIFFERENCE INTO $TMP12
4167 015714 061437 001306 ADD (R4),$TMP12 ;ADD DIFF TO READ SCORE
4168 015720 023727 001306 000014 CMP $TMP12,#12. ;SEE IF 12 EXCEEDED
4169 015726 101403 BLOS 165 ;BR IF NOT
4170 015730 012737 000014 001306 MOV #12,$TMP12 ;SET SCORE = 12.
4171 015736 013724 001306 165: MOV $TMP12,(R4)+ ;PUT BACK ADJUSTED SCORE
4172 015742 000757 BR 155 ;BR TO CONTINUE
4173 015744 062701 000040 185: ADD #32.,R1 ;POINT TO SCORES FOR NEXT TRACK
4174 015750 020127 007050 CMP R1,#NCSCRO+64. ;SEE IF SCORE TABLE EXCEEDED
4175 015754 101742 BLOS 145 ;BR IF NOT YET
4176 ;ADJUST COMPAT READ SCORES (+ OFSTS) BY SELF-TEST SCORES
4177 015756 012701 007110 MOV #PCSCRO,R1 ;SET COMPAT SCORE POINTER
4178 015762 012703 007256 MOV #PSSCRO,R3 ;SET SELF-TEST SCORE POINTER
4179 015766 012700 005502 245: MOV #DRVLST,R0 ;SET DRIVE LIST POINTER
4180 015772 012737 000014 001304 MOV #12.,$TMP11 ;COMPUTE DIFFERENCE OF 12 (DEC) AND
; SELF-TEST SCORE FOR THIS TRACK
4181
4182 016000 162337 001304 SUB (R3)+,$TMP11
4183 016004 010104 MOV R1,R4 ;GET READ SCORE POINTER IN R4
4184 016006 005720 255: TST (R0)+ ;SEE IF A DRIVE LISTED HERE
4185 016010 001417 BEQ 285 ;BR IF NOT

```

4186	016012	013737	001304	001306	MOV	\$TMP11,\$TMP12	;PUT DIFFERENCE INTO \$TMP12
4187	016020	061437	001306		ADD	(R4),\$TMP12	;ADD DIFF TO READ SCORE
4188	016034	023727	001306	000014	CMP	\$TMP12,#12.	;SEE IF 12 EXCEEDED
4189	016032	101403			BLOS	26\$;BR IF NOT
4190	016034	012737	000014	001306	MOV	#12,\$TMP12	;SET SCORE = 12.
4191	016042	013724	001306		26\$: MOV	\$TMP12,(R4)+	;PUT BACK ADJUSTED SCORE
4192	016046	000757			BR	25\$;BR TO CONTINUE
4193	016050	062701	000040		28\$: ADD	#32,R1	;POINT TO SCORES FOR NEXT TRACK
4194	016054	020127	000210		CMP	R1,#PCSCRO+64.	;SEE IF SCORE TABLE EXCEEDED
4195	016060	101742			BLOS	24\$;BR IF NOT YET

*TEST 11 TYPE TEST SCORES, DISMOUNT PACK, IN PASS 2

* THE OVERWRITE AND COMPATIBILITY DATA READ TEST SCORES FOR THIS DRIVE ARE CONVERTED AND TYPED. THEN, THE OPERATOR IS DIRECTED TO UNLOAD THE DRIVE CURRENTLY UNDER TEST, AND TO REMOVE THE TEST PACK. THEN, THE PROGRAM DOES 1 OF 4 THINGS:

* (1) IF THERE IS ANOTHER DRIVE TO TEST ON THIS SUBSYS, THE PROGRAM JUMPS TO TEST 5 FOR THE NEXT DRIVE.

* (2) IF THERE IS ONLY ONE SUBSYS (FROM 200 START), AND IF PASS 2 HAS BEEN COMPLETED ON ALL DRIVES, THE ENTIRE TESTING AND REPORTING HAVE BEEN COMPLETED, AND THE PROGRAM TYPES A COMPLETION MESSAGE, AND THEN IT RESTARTS AT ADDR 200.

* (3) IF THERE IS ANOTHER SUBSYS TO PERFORM PASS 2 ON, THE PROGRAM TELLS THE OPERATOR TO RESTART AT ADDRESS 220, AND THEN IT HALTS.

* (4) IF THERE ARE NO MORE DRIVES TO TEST ON ANY OTHER SUBSYS, THE PROGRAM TYPES A COMPLETION MESSAGE AND THEN IT HALTS.

*TST11: SCOPE

4219	016062	000004			MOV	#11,\$TESTN	;SET TEST NUMBER IN APT MAIL BOX
4220	016064	012737	000011	001334	JSR	PC,SETUP	;SET UP FOR LOOP ON ERROR
4221	016072	004737	021424		JSR	PC,INITSS	;INITIALIZE DRIVER PARAMS AND SUB-SYS
4222	016076	004737	020616		JSR	PC,F_CPKB	;PREPARE FOR POSSIBLE KBD INPUT
4223	016102	004737	020422		MOV	#1,\$TESTING	;ALLOW TTY ESCAPE
4224	016106	112737	000001	003117			
4225					;CONVERT AND TYPE TEST SCORES		
4226	016114	013700	005544		MOV	DRVPT,RO	;GET DRIVE LIST POINTER
4227	016120	112037	010513		MOVB	(R1)+,\$SLTS+20.	;GET NO. OF THIS DRIVE
4228	016124	111037	010512		MOVB	(R1),RESULTS+19.	
4229	016130	104401	010467		TYPE	\$SLTS	;TYPE TEST RESULT HEADINGS
4230	016134	012737	006450	001306	MOV	\$PCSCRO,\$TMP12	;GET POINTER TO OVRWRT SCORES
4231	016142	012700	000060		MOV	#0,RO	;INIT TRACK NO. TO 0
4232	016146	012701	005502		5\$: MOV	#DRVLIST,R1	;GET DRIVE LIST POINTER
4233	016152	110037	010775		MOVB	RO,TRKBUF+1	;SET TRACK NO. FOR PRINTOUT
4234	016156	013704	001306		MOV	\$TMP12,R4	;GET SCORE POINTER
4235	016162	004737	024722		6\$: JSR	PC,CTLOUT	;CHECK FOR TTY ESCAPE RQST
4236	016166	104401	010774		TYPE	TRKBUF	;TYPE TRACK (HEAD) NUMBER
4237	016172	012103			MOV	(R1)+,R3	;GET A DRIVE NAME
4238	016174	020377	167344		CMP	R3,\$DRVPT	;SEE IF IT IS CURRENT DRIVE
4239	016200	001003			BNE	8\$;BR IF NOT
4240	016202	104401	011005		TYPE	\$SELF	;TYPE "SELF"
4241	016206	000407			BR	10\$;CONTINUE

4242	016210	110337	011002	85:	MOV	R3, DRVBUF+2	; GET DRIVE NO.	
4243	016214	000303			SWAB	R3		
4244	016216	110337	011001		MOV	R3, DRVBUF+1	; GET SUBSYS NAME	
4245	016222	104401	011000		TYPE	DRVBUF	; TYPE NO. OF DRIVE READ	
4246	016226	000337	005202	105:	CLR	BUFF0	; CLEAR TTY CHAR BUF	
4247	016232	000337	005204		CLR	BUFF0+2		
4248	016236	010437	001310		MOV	R4, STMP13	; GET SCORE POINTER	
4249	016242	012437	005202		MOV	(R4)+, BUFF0	; PUT AN OVRWRT SCORE INTO BUF	
4250	016246	012737	000004	001304	MOV	R4, STMP11	; SET COUNTER = 4	
4251	016254	023727	005202	000012	115:	CMP	BUFF0, #10.	; SEE IF SCORE < 10.
4252	016262	103002			BHS	125	; BR IF NOT	
4253	016264	104401	011720		TYPE	SPACE1	; TYPE 1 SPACE	
4254	016270	023727	005202	000007	125:	CMP	BUFF0, #7	; SEE IF SCORE > 7
4255	016276	101003			BHI	145	; BR IF YES	
4256	016278	104401	011740		TYPE	PROMPT	; TYPE "*" "	
4257	016304	000402			BR	165	; CONTINUE	
4258	016306	104401	011717	145:	TYPE	SPACE2	; TYPE 2 SPACES	
4259	016312	012746	005202	165:	MOV	BUFF0, -(SP)	; GET BUF ADRS ON STACK	
4260	016316	004737	042330		JSR	PC, @#S0820	; CONVERT SCORE TO DEC.	
4261	016322	004737	042524		JSR	PC, @#SSUPRS	; TYPE IT	
4262	016326	104401	011013		TYPE	TAB	; TYPE A TAB	
4263	016332	062737	000140	001310	ADD	#96, STMP13		
4264	016340	017737	162744	005202	MOV	@STMP13, BUFF0	; PUT NEXT SCORE INTO BUFFER	
4265	016346	005337	001304		DEC	STMP11	; DECREMENT SCORE COUNTER	
4266	016352	001340			BNE	115	; BR IF READ SCORE SHOULD BE TYPED	
4267	016354	104401	001325		TYPE	CR LF	; TYPE (CR), (LF)	
4268	016360	005711			TST	(R1)	; SEE IF ALL SCORES TYPED FOR THIS TRACK	
4269	016362	001277			BNE	65	; BR IF NOT	
4270	016364	104401	001325		TYPE	CR LF	; TYPE (CR), (LF)	
4271	016370	005200			INC	R0	; INCR TRACK NO.	
4272	016372	062737	000040	001306	ADD	#32, STMP12	; INCREMENT SCORE POINTER FOR NEXT TRACK	
4273	016400	020027	000062		CMP	R0, #12	; SEE IF ALL TRACKS TYPED YET	
4274	016404	101660			BLOS	55	; BR IF NOT YET	
4275					; COMPUTE, CONVERT, AND TYPE TEST AVERAGES (OVER ALL SURFACES)			
4276	016406	004737	017710		JSR	PC, TRKAVG	; GET AVGS OVER ALL TRKS	
4277	016412	013700	005544		MOV	DRVPTR, R0	; PUT DRIVE NAME INTO BUF	
4278	016416	112037	011062		MOV	(R0)+, BADDRV+7		
4279	016422	111037	011061		MOV	(R0), BADDRV+6		
4280	016426	005037	001304		CLR	STMP11	; CLEAR SCORE INDICATOR	
4281	016432	013737	007264	005202	MOV	OVRVAVG, BUFF0	; GET OVRWRT AVG SCORE	
4282	016440	012737	010702	016454	MOV	#OAVERG, 225		
4283	016446	104401	011053	205:	TYPE	, BADDRV	; TYPE "DRIVE XX"	
4284	016452	104401			TYPE		; TYPE WHICH KIND OF AVERAGE	
4285	016454	000000		225:	.WORD	0		
4286	016456	012746	005202		MOV	BUFF0, -(SP)	; PUT POINTER TO AVG ON STACK	
4287	016462	004737	042330		JSR	PC, @#S0820	; CONVERT TO DECIMAL	
4288	016466	004737	042524		JSR	PC, @#SSUPRS	; TYPE IT	
4289	016472	104401	001325		TYPE	CR LF	; TYPE (CR), (LF)	
4290	016476	013737	007266	005202	MOV	COMAVG, BUFF0	; GET READ AVG SCORE	
4291	016504	012737	010721	016454	MOV	#RAVERG, 225		
4292	016512	005137	001304		COM	STMP11	; COMPLEMENT SCORE INDIC	
4293	016516	001353			BNE	205	; BR TO TYPE READ SCORE	
4294	016520	104401	001325		TYPE	CR LF	; TYPE (CR), (LF)	
4295					; DISMOUNT PACK ON THIS DRIVE			
4296	016524	105037	003117		CLRB	TSTING	; DON'T ALLOW TTY ESCAPE	
4297	016530	004737	022054		JSR	PC, DMCART	; DIRECT OPERATOR TO DISMOUNT PACK	

```

4298 016534 062737 000002 005544 ADD #2,DRVPTA ;INCREMENT DRIVE LIST POINTER
4299 016542 013700 005544 MOV DRVPTA,R0
4300 016546 005710 TST (R0) ;SEE IF ALL DONE WITH PASS 2 YET
4301 016550 001421 BEQ 28$ ;BR IF DONE WITH PASS 2
4302 016552 126037 000001 005500 CMPB 1(R0),SUBSYS ;SEE IF DONE WITH THIS SUBSYS YET
4303 016560 001005 BNE 24$ ;BR IF DONE WITH THIS SUBSYS
4304 016562 112737 000004 001102 MOVB #4,$STSTNM ;SET TEST NO. = 4
4305 016570 000137 014574 JMP TST5 ;DO PASS 2 ON NEXT DRIVE OF SUBSYS
4306 016574 116037 000001 010344 24$: MOVB 1(R0),STR220+39. ;PUT SUBSYS NAME INTO MSG
4307 016602 104401 010275 TYPE ,STR220 ;TYPE RESTART MSG FOR PASS 2
4308 016606 000000 26$: HALT ;HALT THE PROCESSOR SO THAT OPERATOR
4309 016610 000137 000220 JMP 220 ;CAN RESTART AT PROPER ADRS
4310 016614 104401 010740 28$: TYPE ,ENDTST ;TYPE "** END OF TESTING **"
4311 016620 012737 005502 005544 MOV #DRVLIST,DRVPTA ;RE-INIT DRIVE LIST POINTER
4312 016626 105737 003116 TSTB MDFLAG ;SEE IF 200 START
4313 016632 001002 BNE 30$ ;BR IF NOT
4314 016634 000137 000200 JMP 200 ;RESTART AT ADRS 200
4315 016640 000000 30$: HALT ;HALT THE PROCESSOR
4316 016642 000137 000204 JMP 204 ;RESTART AT ADRS 204

```

.SBTTL END OF PASS ROUTINE

```

*****
;INCREMENT THE PASS NUMBER ($PASS)
;IF THERES A MONITOR GO TO IT
;IF THERE ISN'T JUMP TO 200

```

```

4327 016646 SEOP:
4328 016646 000004 SCOPE
4329 016650 005037 001102 CLR $STSTNM ;ZERO THE TEST NUMBER
4330 016654 005237 001336 INC $PASS ;INCREMENT THE PASS NUMBER
4331 016660 042737 100000 001336 BIC #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
4332 016666 005327 DEC (PC)+ ;LOOP?
4333 016670 000001 SEOPCT: .WORD 1
4334 016672 003013 BGT $DOAGN ;YES
4335 016674 012737 MOV (PC)+,2(PC)+ ;RESTORE COUNTER
4336 016676 000001 SENDCT: .WORD 1
4337 016700 016670 SEOPCT
4338 016702 013700 000042 $GET42: MOV #42,R0 ;GET MONITOR ADDRESS
4339 016706 001405 $DOAGN ;BRANCH IF NO MONITOR
4340 016710 000005 RESET ;CLEAR THE WORLD
4341 016712 004710 SENDAD: JSR PC,(R0) ;GO TO MONITOR
4342 016714 000240 NOP ;SAVE ROOM
4343 016716 000240 NOP ;FOR
4344 016720 000240 NOP ;ACT11
4345 016722 $DOAGN:
4346 016722 000137 000200 JMP #200 ;RETURN

```

```

*****
;REDOFS - READ DATA WITH RANGE OF OFFSETS
;THIS SUBROUTINE PERFORMS READS WITH OFFSETS LISTED IN OFSTBL.
;THE OFFSETS ARE APPLIED MINIMUM TO MAXIMUM, IN BOTH DIRECTIONS,

```

```

4354      ;#UNTIL THERE IS NO FAILURE. EACH TIME A READ ERROR OCCURS.
4355      ;#THE APPROPRIATE SCORE FOR THE DRIVE INVOLVED IS INCREMENTED NO FURTHER
4356      ;#(IN THE SCRATCH SCORE TABLE). EACH READ IS DONE TO A PACK
4357      ;#AREA, WHOSE ADDRESS IS IN THE PARAMETER BLOCK ON ENTRY.
4358      ;*****
4359      016726 104407      REDOFS: SAVREG      ;SAVE RD-RS
4360      ;INITIALIZE SCRATCH SCORES
4361      016730 122737 000003 003133      CMPB      #3,TSTTYP      ;SEE IF DRIVE SELF TEST
4362      016736 001406      BEQ      55              ;BR IF YES
4363      016740 012700 007270      MOV      #NSCORD,R0      ;GET ADORS OF SCRATCH SCORE TABLE
4364      016744 105020      45:      CLRB      (R0)+      ;INIT A SCORE TO 0
4365      016746 020027 007430      CMP      R0,#NSCORD+96.  ;SEE IF DONE YET
4366      016752 103774      BLO      45              ;BR IF NOT YET
4367      ;PERFORM READS WITH OFFSET
4368      016754 112765 000117 000001      55:      MOVB      #SEEK,P.CMND(R5) ;SET SEEK COMMAND
4369      016762 105737 003137      TSTB      DIF100      ;SEE IF CYL DIF = 100 (OCT)
4370      016766 001412      BEQ      95              ;BR IF NOT
4371      016770 162765 000040 000002      SUB      #40,P.CYLN(R5) ;SEEK HALF THE CYLS
4372      016776 004737 027302      JSR      PC,DRVCAL
4373      017002 062765 000040 000002      ADD      #40,P.CYLN(R5) ;SEEK THE REST OF THE WAY
4374      017010 105037 003137      CLRB      DIF100      ;CLEAR "CYL DIF = 100 FLAG"
4375      017014 004737 027302      95:      JSR      PC,DRVCAL      ;PERFORM SEEK
4376      017020 105037 003146      CLRB      OFSDIR      ;CLEAR OFFSET DIRECTION FLAG
4377      017024 012703 000014      MOV      #12,R3        ;SET OFFSET COUNTER FOR NEG OFSTS
4378      017030 012700 000204      MOV      #204,R0       ;INIT NEG OFST VALUE TO 100 UIN
4379      017034 105037 003147      125:     CLRB      OFSCNT      ;INIT OFFSET NUMBER
4380      017040 122737 000003 003133      65:      CMPB      #3,TSTTYP      ;SEE IF SELF-TEST
4381      017046 001020      BNE      155           ;BR IF NOT
4382      017050 116501 000005      MOVB      P.TRCK(R5),R1 ;GET TRACK NUMBER
4383      017054 006301      ASL      R1            ;MULT BY 16.
4384      017056 006301      ASL      R1
4385      017060 006301      ASL      R1
4386      017062 006301      ASL      R1
4387      017064 062701 007270      ADD      #NSCORD,R1      ;GET POINTER TO SCORES
4388      017070 105737 003146      TSTB      OFSDIR      ;SEE WHICH OFST DIRECTION
4389      017074 001402      BEQ      175           ;BR IF NEGATIVE
4390      017076 062701 000060      ADD      #48,R1        ;POINT TO POSITIVE SCORES
4391      017102 012704 000001      175:     MOV      #1,R4        ;SET COUNTER TO 1
4392      017106 000411      BR      145           ;CONTINUE
4393      017110 012701 007270      155:     MOV      #NSCORD,R1      ;GET POINTER TO SCORES
4394      017114 012704 000060      MOV      #48,R4        ;SET COUNTER TO 48
4395      017120 105737 003146      TSTB      OFSDIR      ;SEE WHICH OFST DIRECTION
4396      017124 001402      BEQ      145           ;BR IF NEGATIVE
4397      017126 012701 007350      MOV      #PSCORD,R1     ;GET POINTER FOR POS SCORES
4398      017132 122137 003147      145:     CMPB      (R1)+,OFSCNT  ;SEE IF THIS IS A PERFECT SCORE YET
4399      017136 001002      BNE      185           ;BR IF NOT
4400      017140 105261 177777      INCB      -1(R1)       ;INCREMENT THIS SCORE
4401      017144 005304      185:     DEC      R4            ;DECREMENT COUNTER
4402      017146 001371      BNE      145           ;BR IF NOT DONE YET
4403      017150 110065 000006      165:     MOVB      R0,P.OFST(R5) ;GET AN OFFSET VALUE
4404      017154 112765 000115 000001      MOVB      #OFFSET,P.CMND(R5) ;SET OFFSET CMND
4405      017162 004737 027302      JSR      PC,DRVCAL      ;PERFORM OFFSET
4406      017166 105237 003147      INCB      OFSCNT      ;INCR OFFSET NUMBER
4407      017172 112765 000121 000001      MOVB      #RODATA,P.CMND(R5) ;SET READ COMMAND
4408      017200 004737 026634      JSR      PC,SVPRMS      ;SAVE PARAMETERS OF TRANSFER
4409      017204 004737 026724      JSR      PC,TRNSFR      ;DO XFER, HANDLE DCK ERRORS

```

```

4410 017210 112765 000177 000001
4411 017216 004737 027302
4412 017222 105065 000006
4413 017226 112765 000115 000001
4414 017234 004737 027302
4415 017240 062700 000004
4416 017244 005303
4417 017246 001274
4418 017250 012703 000014
4419 017254 012700 000004
4420 017260 105137 003146
4421 017264 001263
4422 017266 104410
4423 017270 000207

```

```

MOV B #SUBCLR,P.CHND(R5) ; SET SUBSYS CLEAR COMMAND
JSR PC,DRVCAL ; DO A SUBSYSTEM CLEAR
CLRB P.OFST(R5) ; CLEAR OFFSET VALUE
MOV B #OFFSET,P.CHND(R5) ; SET OFFSET COMMAND
JSR PC,DRVCAL ; PERFORM OFFSET TO 0
ADD #4,R0 ; INCR OFST VALUE BY 100 UIN
DEC R3 ; DECR COUNTER
BNE 6$ ; BR IF NOT DONE IN THIS DIRECTION YET
MOV #12,R3 ; SET COUNTER FOR POS SFSTS
MOV #4,R0 ; INIT POS OFST TO 100 UIN
COMB OFSDIR ; COMPLEMENT OFFSET DIRECTION FLAG
BNE 12$ ; BR IF POS OFSTS NOT DONE YET
RESREG ; RESTORE R0-R5
RTS PC ; RETURN

```

```

*****
*ADSCOR - ADD THE + AND - SCRATCH SCORES TO THE + AND - SCORE
*SUMS, FOR EITHER THE OVRWRT, SELF, OR COMPAT READ TEST, AS DETERMINED
*BY "TSTYP".
*****

```

```

4431 017272 104407
4432 017274 122737 000003 003133
4433 017302 001031
4434 017304 113700 007270
4435 017310 060037 007250
4436 017314 113700 007310
4437 017320 060037 007252
4438 017324 113700 007330
4439 017330 060037 007254
4440 017334 113700 007350
4441 017340 060037 007256
4442 017344 113700 007370
4443 017350 060037 007260
4444 017354 113700 007410
4445 017360 060037 007262
4446 017364 000430
4447 017366 012700 006450 2$:
4448 017372 012705 006610
4449 017376 122737 000001 003133
4450 017404 001404
4451 017406 012700 006750
4452 017412 012705 007110
4453 017416 012704 000060 4$:
4454 017422 012701 007270
4455 017426 012703 007350
4456 017432 112102 6$:
4457 017434 060220
4458 017436 112302
4459 017440 060225
4460 017442 005304
4461 017444 001372
4462 017446 104410
4463 017450 000207
4464
4465

```

```

ADSCOR: SAVREG ; SAVE R0-R5
CMPB #3,TSTYP ; SEE IF SELF-TEST
BNE 2$ ; BR IF NOT
MOV #NSCOR0,R0 ; ADD SELF-TEST SCORES
ADD R0,NSSCR0
MOV #NSCOR1,R0
ADD R0,NSSCR1
MOV #NSCOR2,R0
ADD R0,NSSCR2
MOV #PSCOR0,R0
ADD R0,PSSCR0
MOV #PSCOR1,R0
ADD R0,PSSCR1
MOV #PSCOR2,R0
ADD R0,PSSCR2
BR 8$ ; EXIT
MOV #NOSCR0,R0 ; GET POINTER TO (-) OVRWRT SCORES
MOV #POSCR0,R5 ; GET POINTER TO (+) OVRWRT SCORES
CMPB #1,TSTYP ; SEE IF READING OVRWRT DATA
BEQ 4$ ; BR IF YES
MOV #NCSCR0,R0 ; GET POINTER TO (-) COMPAT DATA SCORES
MOV #PCSCR0,R5 ; GET POINTER TO (+) COMPAT DATA SCORES
MOV #48,R4 ; INIT COUNTER TO 48.
MOV #NSCOR0,R1 ; GET POINTER TO NEG SCRATCH SCORES
MOV #PSCOR0,R3 ; GET POINTER TO POS SCRATCH SCORES
MOV B (R1)+,R2 ; GET A SCRATCH NEG SCORE
ADD R2,(R0)+ ; ADD IT TO (-) SCORE SUM
MOV B (R3)+,R2 ; GET PNTR TO POS SCRATCH SCORES
ADD R2,(R5)+ ; ADD IT TO (+) SCORE SUM
DEC R4 ; DECREMENT COUNTER
BNE 6$ ; BR IF NOT DONE YET
RESREG ; RESTORE R0-R5
RTS PC ; RETURN

```

```

4466
4467
4468
4469
4470
4471
4472
4473
4474 017452 104407
4475 017454 122737 000003 003133
4476 017462 001014
4477 017464 012737 007250 001304
4478 017472 012737 007256 001310
4479 017500 012705 000030
4480 017504 012737 000003 001306
4481 017512 000427
4482 017514 012737 006450 001304 25:
4483 017522 012737 006610 001310
4484 017530 012705 000007
4485 017534 122737 000001 003133
4486 017542 001410
4487 017544 012737 006750 001304
4488 017552 012737 007110 001310
4489 017560 012705 000140
4490 017564 012737 000060 001306 45:
4491 017572 105037 003146 125:
4492 017576 005000 65:
4493 017600 005001
4494 017602 005002
4495 017604 005004
4496 017606 017703 161472
4497 017612 105737 003146
4498 017616 001402
4499 017620 017703 161464
4500 017624 004737 042032 105:
4501 017630 010500
4502 017632 006200
4503 017634 020100
4504 017636 103401
4505 017640 005203
4506 017642 105137 003146 85:
4507 017646 001406
4508 017650 010377 161430
4509 017654 062737 000002 001304
4510 017662 000745
4511 017664 010377 161420 145:
4512 017670 062737 000002 001310
4513 017676 005337 001306
4514 017702 001333
4515 017704 104410
4516 017706 000207
4517
4518
4519
4520
4521

```

```

*****
;AVSCOR - COMPUTE THE + AND - AVERAGE SCORES FOR THE OVRWRT, COMPATIBILITY
;DATA TEST, OR SELF-TEST, AS DETERMINED BY "TSTYP". THE SUMS IN THE APPROPRIATE
;SCORE TABLES ARE EACH DIVIDED BY THE NUMBER OF CYLINDERS READ, TO
;COMPUTE THE AVERAGES.
*****

```

```

AVSCOR: SAVREG          ;SAVE R0-R5
          CMPB          #3,TSTYP      ;SEE IF SELF-TEST
          BNE          25             ;BR IF NOT
          MOV          #N5SCRO,$TMP11 ;GET ADRS OF (-) SELF-TEST SCORES
          MOV          #P5SCRO,$TMP13 ;GET ADRS OF (+) SELF-TEST SCORES
          MOV          #24,R5         ;SET DIVISOR = 24.
          MOV          #3,$TMP12      ;SET COUNTER = 3
          BR          125            ;CONTINUE
          MOV          #N0SCRO,$TMP11 ;GET POINTER TO (-) OVRWRT SCORES
          MOV          #P0SCRO,$TMP13 ;GET POINTER TO (+) OVRWRT SCORES
          MOV          #7,R5          ;SET DIVISOR = 7.
          CMPB          #1,TSTYP      ;SEE IF READING OVRWRT DATA
          BEQ          45             ;BR IF YES
          MOV          #NCSRO,$TMP11 ;GET POINTER TO (-) COMPAT DATA SCORES
          MOV          #PCSCRO,$TMP13 ;GET POINTER TO (+) COMPAT DATA SCORES
          MOV          #96,R5         ;SET DIVISOR = 96.
          MOV          #48,$TMP12     ;SET COUNTER = 48.
          CLR          OFSDIR         ;INIT FLAG FOR (-) DIRECTION
          CLR          R0             ;CLEAR HI BITS OF DIVIDEND AND DIVISOR
          CLR          R1
          CLR          R2
          CLR          R4
          MOV          @TMP11,R3      ;SET DIVIDEND = (-) SUM
          TSTB          OFSDIR        ;SEE WHICH OFST DIRECTION
          BEQ          105
          MOV          @TMP13,R3      ;SET DIVIDEND = (+) SUM
          JSR          PC,M.DP10      ;DIVIDE TO GET AVERAGES
          MOV          R5,R0          ;GET DIVISOR
          ASR          R0             ;DIVIDE IT BY 2
          CMP          R1,R0          ;SEE IF REMNDR > HALF DIVISOR
          BLO          85             ;BR IF NOT
          INC          R3             ;ADD 1 TO QUOTIENT
          COMB          OFSDIR        ;COMPL DIR FLAG
          BEQ          145            ;BR FOR (+)
          MOV          R3,@TMP11      ;PUT (-) AVG INTO TABLE
          ADD          #2,$TMP11
          BR          65
          MOV          R3,@TMP13      ;PUT (+) AVG INTO TABLE
          ADD          #2,$TMP13
          DEC          $TMP12         ;DECREMENT COUNTER
          BNE          125            ;BR IF NOT DONE YET
          RESREG
          RTS          PC             ;RESTORE R0-R5
                                     ;RETURN

```

```

*****
;TRKAVG - COMPUTE AVERAGES OF TEST SCORES OVER ALL SURFACES
*****

```

```

4522
4523
4524
4525
4526
4527 017710 104407
4528
4529 017712 012700 005502
4530 017716 005001
4531 017720 005037 007264
4532 017724 005037 007266
4533 017730 012702 006450
4534 017734 012703 006750
4535 017740 005720
4536 017742 001402
4537 017744 0052C1
4538 017746 000774
4539 017750 010100
4540 017752 010204
4541 017754 010305
4542 017756 062437 007264
4543 017762 066437 000136 007264
4544 017770 062537 007266
4545 017774 066537 000136 007266
4546 020002 005300
4547 020004 001364
4548 020006 062702 000040
4549 020012 062703 000040
4550 020016 020227 006550
4551 020022 101752
4552
4553 020024 005004
4554 020026 005005
4555 020030 060105
4556 020032 060105
4557 020034 060105
4558 020036 006305
4559 020040 012737 007264 001304
4560 020046 005037 001306
4561 020052 005000
4562 020054 005001
4563 020056 005002
4564 020060 017703 161220
4565 020064 004737 042032
4566 020070 010500
4567 020072 006200
4568 020074 020100
4569 020076 103401
4570 020100 005203
4571 020102 010377 161176
4572 020106 012737 007266 001304
4573 020114 005137 001306
4574 020120 001354
4575 020122 104410
4576 020124 000207
4577

```

```

; *THIS SUBROUTINE COMPUTES THE AVERAGE OF THE OVERWRITE TEST SCORES
; *OVER ALL TRACKS (SURFACES) AND PLACES THE AVERAGE IN OVR AVG.
; *THEN, IT COMPUTES THE AVERAGE OF THE COMPATIBILITY DATA READ SCORES
; *OVER ALL TRACKS AND PLACES THE AVERAGE IN COMAVG.
; *****
TAKAVG: SAVREG ; SAVE RO-R5
; TAKE SUMS OF OVERWRITE AND COMPAT READ TEST SCORES OVER ALL TRACKS
MOV #DVLST, R0 ; INIT DRIVE LIST POINTER
CLR R1 ; INIT DRIVE COUNTER
CLR OVR AVG ; INIT SUMS TO 0
CLR COMAVG
MOV #NOSCR0, R2 ; SET POINTER TO OVRWRT SCORES
MOV #NCSCR0, R3 ; SET POINTER TO READ SCORES
85: TST (R0)+ ; COMPUTE TOTAL NUMBER OF DRIVES
BEQ 85 ; IN R1
INC R1
BR 65
85: MOV R1, R0 ; PUT NO. OF DRIVES IN R0
MOV R2, R4 ; SET POINTERS TO SCORES FOR THIS TRACK
MOV R3, R5
95: ADD (R4)+, OVR AVG ; ADD SCORES TO SUMS
ADD 94.(R4), OVR AVG
ADD (R5)+, COMAVG
ADD 94.(R5), COMAVG
DEC R0 ; SEE IF ALL SCORES ADDED FOR THIS TRACK
BNE 95 ; BR IF NOT YET
ADD #32., R2 ; POINT TO SCORES FOR NEXT TRACK
ADD #32., R3
CMP R2, #NOSCR0+64. ; SEE IF ALL TRACKS DONE YET
BLOS 85 ; BR IF NOT YET
; DIVIDE TO GET AVERAGES
CLR R4 ; GET DIVISOR IN R4-R5
CLR R5
ADD R1, R5
ADD R1, R5
ADD R1, R5
ASL R5
MOV #OVR AVG, STMP11 ; SET POINTER TO OVRWRT SCORE SUM
CLR STMP12 ; CLEAR SCORE INDICATOR
105: CLR R0 ; GET DIVIDEND IN R0-R1-R2-R3
CLR R1
CLR R2
MOV @STMP11, R3
JSR PC, M.OP10 ; DIVIDE TO GET AVERAGE
MOV R5, R0 ; GET DIVISOR
ASR R0 ; DIVIDE IT BY 2
CMP R1, R0 ; SEE IF REMNDR > HALF DIVISOR
BLO 125 ; BR IF NOT
INC R3 ; ADD 1 TO QUOTIENT
MOV R3, STMP11 ; STORE AVERAGE SCORE
MOV #COMAVG, STMP11 ; SET POINTER FOR READ SCORE SUM
COM STMP12 ; COMPLEMENT SCORE INDICATOR
105: BNE 105 ; BR TO COMPUTE READ SCORE AVERAGE
RESREG ; RESTORE RO-R5
RTS PC ; RETURN

```

4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633

020126 032765 010000 000020
020134 001002
020136 000137 030640
020142 010046
020144 010146
020146 012700 000001
020152 005001
020154 120137 005420
020160 001403
020162 005201
020164 006300
020166 000772
020170 040037 005466
020174 012601
020176 012600
020200 000137 033110

```
*****  
: #NEDHDL - NON-EXISTENT DRIVE HANDLER  
: *THIS IS THE ABNORMAL RETURN FROM THE DRIVER,  
: * WHICH IS USED WHEN NED INDICATION IS EXPECTED (AND VALID).  
: * (NED = NON-EXISTENT DRIVE)  
: *****  
NEDHDL: BIT #NED,P.CS2(R5) ;SEE IF NED ON DRIVE SELECT  
;BNE 1$ ;BR IF NED SET  
;JMP ERRHDL ;GO HANDLE OTHER ERROR  
;SAVE R0,R1  
1$: MOV R0,-(SP)  
MOV R1,-(SP)  
MOV #1,R0 ;SET BIT POINTER  
CLR R1 ;CLEAR COUNTER  
2$: CMPB R1,DRIVE ;SEE IF R1 = CURRENT DRIVE NUMBER  
;BEQ 3$ ;BR IF =  
;INC R1 ;INCREMENT COUNTER  
;ASL R0 ;SHIFT BIT POINTER  
;BR 2$ ;TRY AGAIN  
3$: BIC R0,NEWON ;CLEAR ONLINE BIT FOR THIS DRIVE  
;MOV (SP)+,R1 ;RESTORE R0,R1  
;MOV (SP)+,R0  
;JMP RETNML ;TAKE NORMAL RETURN
```

```
*****  
: #DCKHDL - DATA CHECK ERROR HANDLER  
: *THIS IS THE ABNORMAL RETURN FROM THE DRIVER, WHICH IS USED  
: *WHEN DCK ERROR IS EXPECTED (AND VALID).  
: *IF A HEADER VRC ERROR OCCURS, IT IS HANDLED, ALSO.  
: *****  
DCKHDL: CLR RECODE ;CLEAR ERROR FLAGS  
BIT #HVRC,P.ER(R5) ;SEE IF HEADER VRC ERROR  
;BEQ 4$ ;BR IF NOT  
;BIS #HVRCER,RECODE ;SET HVRC ERROR FLAG  
;BR 8$ ;EXIT  
4$: BIT #DCK,P.ER(R5) ;SEE IF DCK ERROR OCCURRED  
;BNE 6$ ;BR IF YES  
;JMP ERRHDL ;GO HANDLE OTHER ERROR  
6$: BIS #DCKERR,RECODE ;SET DCK ERROR FLAG  
8$: INCB DCEFLG ;INCREMENT DCK ERROR FLAG  
;JMP RETNML ;TAKE NORMAL RETURN FROM DRIVER
```

```
*****  
: #BTTL KBDHDL - TTY KEYBOARD INTERRUPT HANDLER  
: *****  
KBDHDL: MOV R1,-(SP) ;SAVE R1 ON STACK  
;MCP #17701,R1 ;READ A CHAR FROM KBD BUFFER  
;BIC #177600,R1 ;MASK OUT UNUSED BITS
```

```

4634 020274 120127 000172      CMPB   R1,#172      ;SEE IF LOWER CASE TYPED
4635 020300 003005                BGT    20$          ;BR IF NOT
4636 020302 120127 000141      CMPB   R1,#141
4637 020306 002402                BLT    20$          ;BR IF NOT
4638 020310 042701 000040      BIC    #815,R1      ;MAKE IT UPPER CASE
4639 020314 010137 005432      20$:  MOV    R1,INTCHR ;SAVE INPUT CHARACTER
4640 020320 122701 000003      CMPB   #003,R1     ;SEE IF (↑C) TYPED
4641 020324 001007                BNE    2$          ;BR IF NOT (↑C)
4642 020326 104401 011652      TYPE  ,CNTRLC      ;ECHO (↑C)
4643 020332 042777 000100      1$:  BIC    #BIT6,#STKS ;CLEAR KBD INTERRUPT ENABLE BIT
4644 020340 012601                MOV    (SP)+,R1    ;RESTORE R1
4645 020342 000002                RTI
4646 020344 122701 000032      2$:  CMPB   #032,R1   ;SEE IF (↑Z) TYPED
4647 020350 001003                BNE    3$          ;BR IF NOT (↑Z)
4648 020352 104401 011657      TYPE  ,CNTRLZ      ;ECHO (↑Z)
4649 020356 000765                BR     1$          ;RETURN
4650 020360 122701 000022      3$:  CMPB   #022,R1   ;SEE IF (↑R) TYPED
4651 020364 001003                BNE    4$          ;BR IF NOT (↑R)
4652 020366 104401 011664      TYPE  ,CNTRLR      ;ECHO (↑R)
4653 020372 000757                BR     1$          ;RETURN
4654 020374 122701 000007      4$:  CMPB   #007,R1   ;SEE IF (↑G) TYPED
4655 020400 001003                BNE    5$          ;BR IF NOT (↑G)
4656 020402 104401 011676      TYPE  ,CNTRLG      ;ECHO (↑G)
4657 020406 000751                BR     1$          ;RETURN
4658 020410 104401 005432      5$:  TYPE  ,INTCHR    ;ECHO INPUT
4659 020414 104401 001325      TYPE  ,$CRLF       ;DO <CR> AND <LF>
4660 020420 000744                BR     1$          ;RETURN

```

4661
4662
4663

```

;*****
;SBTTL PREPKB - PREPARE FOR KEYBOARD INPUT
;THIS SUBROUTINE CLEARS THE KEYBOARD BUFFER AND THE
;DONE BIT, AND CLEARS THE TTY INTERRUPT INPUT WORD
;INTCHR, IN PREPARATION FOR NEW TTY INPUT. IT ALSO
;ENABLES KBD INTERRUPT.
;CALL:
; JSR    PC,PREPKB
;*****

```

4674 020422 005077 160520
4675 020426 005037 005432
4676 020432 052777 000100 160504
4677 020440 000207

```

PREPKB: CLR    #STKB      ;CLEAR KBD BUFFER AND DONE BIT
        CLR    INTCHR    ;CLEAR TTY INPUT WORD
        BIS    #BIT6,#STKS ;ENABLE KBD INTERRUPT
        RTS    PC        ;RETURN

```

4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689

```

;*****
;SBTTL ECOBAD - ECHO BAD TTY INPUT
;THIS SUBROUTINE ECHOS A CHARACTER WHICH HAD BEEN
;TYPED IN AND WAS DETERMINED TO BE INVALID FOR SOME
;REASON, FOLLOWED BY A QUESTION MARK (?). THEN, A <CR>
;AND <LF> ARE DONE.
;CALL - JSR    PC,ECOBAD
;*****

```


ECOBAD: TYPE ,INTCHR ;ECHO BAD CHARACTER
TYPE \$QUES ;TYPE (?) AND (CR), (LF)
RTS PC ;RETURN

4690 020442 104401 005432
4691 020446 104401 001324
4692 020452 000207
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705 020454 016646 000002
4706 020460 104403
4707 020462 006
4708 020463 000
4709 020464 104401 011024
4710 020470 004737 023006
4711 020474 020526
4712 020476 020526
4713 020500 020526
4714 020502 005700
4715 020504 001001
4716 020506 000207
4717 020510 020027 000006
4718 020514 003407
4719 020516 104401 005202
4720 020522 104401 001324
4721 020526 162716 000010
4722 020532 000207
4723 020534 012746 005202
4724 020540 004737 041252
4725 020544 020516
4726 020546 012600
4727 020550 005737 041404
4728 020554 001360
4729 020556 010066 000002
4730 020562 000207

* GETPRM - INPUT A SIX-DIGIT OCTAL NO. ON TTY KBD
* ENTER SUBROUTINE WITH OLD PARAMETER VALUE ON STACK. SUBROUTINE
* TYPES OLD VALUE, AND INPUTS NEW VALUE, RETURNING NEW VALUE ON
* TOP OF STACK.

GETPRM: MOV 2(SP),-(SP) ;GET OLD VALUE
TYP0S ;TYPE IT
.BYTE 6 ;SIX DIGITS
.BYTE 0 ;SUPPRESS LEADING ZEROS
TYPE NEWMSG ;TYPE " NEW = "
JSR PC, RDCHRS ;READ NEW VALUE FROM KBD
7\$;(1C) RETURN ADDRESS
7\$;(1Z) RETURN ADDRESS
7\$;(1U) OR ERROR RETURN ADDRESS
TST R0 ;SEE IF ANY CHARS TYPED
BNE 4\$;BR IF YES
RTS PC ;RETURN - OLD VALUE UNCHANGED
4\$: CMP R0, #6 ;SEE IF > 6 CHARS TYPED
BLE 8\$;BR IF NOT BAD
6\$: TYPE ,BUFF0 ;ECHO BAD INPUT
TYPE \$QUES
7\$: SUB #10,(SP) ;FIX ERROR RETURN PC
RTS PC ;ERROR RETURN
8\$: MOV #BUFF0,-(SP) ;PUT POINTER TO CHARS ON STACK
JSR PC,OCTBIN ;CONVERT DIGITS TO BINARY
6\$;ERROR RETURN ADDRESS
MOV (SP)+,R0 ;GET NEW BINARY VALUE
TST \$HI0CT ;SEE IF HI BITS ARE 0
BNE 6\$;BR IF NOT
MOV R0,2(SP) ;PUT NEW VALUE ON STACK
RTS PC ;RETURN

* SBTTL GTSWRG - OPEN SOFTWARE SWITCH REGISTER FOR MODIFICATION
* THIS SUBROUTINE ALLOWS THE CONTENTS OF THE SOFTWARE SWITCH
* REGISTER TO BE MODIFIED BY TTY INPUT. THE SUBROUTINE TYPES
* "SWR = XXXXXX NEW = " AND WAITS FOR A NEW OCTAL VALUE
* OF UP TO SIX DIGITS TO BE TYPED.

4741 020564 022737 000176 001140 GTSWRG: CMP #SWREG,SWR ;SEE IF SOFTWARE SWR SELECTED
4742 020572 001010 BNE 6\$;BR IF NOT
4743 020574 013746 000176 MOV SWREG,-(SP) ;PUT OLD VALUE ON STACK
4744 020600 104401 011015 TYPE ,SWRMSG ;TYPE "SWR = "
4745 020604 004737 020454 JSR PC,GETPRM ;TYPE OLD, GET NEW SWREG VALUE

F08

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 96
 GTSWRG - OPEN SOFTWARE SWITCH REGISTER FOR MODIFICATION

SEQ 0095

4746 020610 012637 000176
 4747 020614 000207
 4748
 4749
 4750
 4751
 4752
 4753
 4754
 4755
 4756
 4757
 4758
 4759
 4760
 4761
 4762
 4763
 4764
 4765
 4766 020616 012737 030640 003046
 4767 020624 012737 027426 003044
 4768 020632 013702 003036
 4769 020636 012705 002630
 4770 020642 105037 003140
 4771 020646 004737 020710
 4772 020652 112765 000177 000001
 4773 020660 004737 027302
 4774 020664 113765 005420 000000
 4775 020672 113737 005420 002714
 4776 020700 113765 003124 000007
 4777 020706 000207
 4778
 4779
 4780
 4781
 4782
 4783
 4784
 4785
 4786
 4787
 4788 020710 010046
 4789 020712 010546
 4790 020714 010500
 4791 020716 062705 000016
 4792 020722 005020
 4793 020724 020005
 4794 020726 001375
 4795 020730 012605
 4796 020732 012600
 4797 020734 000207
 4798
 4799
 4800
 4801

```

MOV (SP)+,SWREG ;STORE NEW VALUE
65: RTS PC ;RETURN
  
```

```

;*****
;SBTTL INITSS - INITIALIZE SUBSYSTEM
;
;THIS SUBROUTINE INITIALIZES THE DRIVER AND ITS PARAMETERS
;AND DOES A SUBSYSTEM CLEAR.
;USES - R2,R5
;CALL:
; JSR PC,INITSS
;*****
  
```

```

INITSS: MOV #ERRHDL,A.ABNL ;SET UP ABNORMAL ERROR RETURN ADDRESS
MOV #ERRFRE,A.NORM ;SET UP NORMAL RETURN ADDRESS
MOV RKBAS,R2 ;GET ADDRESS OF RK611 REGISTERS
MOV #PARMO,R5 ;GET ADDRESS OF PARAMETER BLOCK
CLRB NORTRY ;CLEAR "NO-RETRY" FLAG
JSR PC,CLRPRM ;CLEAR DRIVER INPUT PARAMETERS
MOV# #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR COMMAND
JSR PC,DRVCAL ;DO SUBSYSTEM CLEAR
MO 3 DRIVE,P.DRVN(R5) ;SET CURRENT DRIVE NO.
MO 8 DRIVE,PARMI ;SET DRIVE NO. IN ALTERNATE P.B.
MO 8 FORMAT,P.CSIH(R5) ;SET CURRENT DRIVE FORMAT
RTS PC ;SUBROUTINE EXIT
  
```

```

;*****
;SBTTL CLRPRM - CLEAR DRIVER INPUT PARAMETERS
;THIS SUBROUTINE ZEROS THE FIRST 14 BYTES IN THE DRIVER
;PARAMETER BLOCK (WHOSE ADDRESS IS IN R5).
;CALL - JSR PC,CLRPRM
;*****
  
```

```

CLRPRM: MOV RO,-(SP) ;SAVE RO
MOV RS,-(SP) ;SAVE RS
MOV RS,RO ;GET PARAMETER BLOCK ADDRESS
ADD #P.CSI,R5 ;COMPUTE LIMIT ADDRESS
1$: CLR (RO)+ ;CLEAR A WORD IN PARAMETER BLOCK
CMP RO,RS ;SEE IF DONE YET
BNE 1$ ;BR IF NOT DONE YET
MOV (SP)+,RS ;RESTORE RS
MOV (SP)+,RO ;RESTORE RO
RTS PC ;RETURN
  
```

```

;*****
  
```

4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857

020736 004737 020616
020742 042712 000100
020746 113762 005420 000010
020754 012712 000001
020760 005037 005434
020764 005237 005434
020770 001375
020772 032762 001000 000010
021000 001417
021002 112765 000101 000001
021010 011265 000016
021014 004737 037000
021020 004737 030360
021024 104052
021026 052737 000200 005414
021034 000137 032472
021040 004737 020616
021044 112765 000141 000001
021052 012737 020126 003046
021060 012737 000377 005466
021066 004737 027302
021072 112737 000040 011062
021100 012737 030640 003046
021106 022737 000377 005466
021114 001420
021116 113737 005420 011061
021124 152737 000060 011061
021132 104401 011053
021136 104401 011064
021142 042762 000100 000000
021150 017616 000000
021154 000207
021156 032765 000200 000040
021164 001013
021166 113737 005420 011061
021174 152737 000060 011061
021202 104401 011053
021206 104401 011103
021212 000753

```
.SBTTL SCNDRV - SCAN DRIVE FOR STATUS
;THIS SUBROUTINE SCANS THE SELECTED DRIVE TO INSURE
;THAT THE DRIVE IS ON-LINE, READY, NOT WRITE-LOCKED,
;AND NOT LOADED WITH AN ALIGNMENT CARTRIDGE, IF ANY
;OF THESE CONDITIONS ARE NOT MET, AN APPROPRIATE
;MESSAGE IS TYPED, RK06 INTERRUPT IS DISABLED, AND
;A RETURN IS MADE TO THE ADDRESS LISTED IMMEDIATELY
;AFTER THE CALL TO SCNDRV. THE REQUESTED DRIVE NO.
;MUST BE PASSED TO THE SUBROUTINE IN WORD "DRIVE".
; CALL - JSR PC,SCNDRV
; (ERROR RETURN ADDRESS)
;*****
SCNDRV: JSR PC,INITSS ;INITIALIZE DRIVER AND CLEAR SUBSYSTEM
3$: BIC #IE,(R2) ;DISABLE RK06 INTERRUPT
MOVB DRIVE,RKCS2(R2) ;SET DRIVE NO. IN RKCS2
MOV #BIT0,(R2) ;SELECT THE DRIVE
CLR SCRACH ;CLEAR STALL COUNTER
15$: INC SCRACH ;INCREMENT STALL COUNTER
BNE 15$ ;STALL FOR SEVERAL MILLI-SEC
BIT #MDS,RKCS2(R2) ;SEE IF MDS ERROR
BEQ 18$ ;BR IF NOT
MOVB #SELDRV,P.CMND(R5) ;SET CMND FOR ERROR REPORT
MOV (R2),P.CS1(R5) ;GET RKCS1 FOR REPORT
JSR PC,I.CST1 ;GET OTHER RK611 REGS FOR REPORT
JSR PC,REPSUP ;SET UP ERROR REPORT
ERROR 52 ;REPORT MDS ERROR
BIS #ABORT,RECODE ;SET ABORT FLAG
JMP ALLTRM ;GO ABORT TESTING
18$: JSR PC,INITSS ;INIT THE S.S.
MOVB #ROSTAT,P.CMND(R5) ;SET READ DRIVE STATUS COMMAND
MOV #MEDHDL,A.ABNL ;SET MED ABNORMAL RETURN ADDRESS
MOV #377,NEWON ;INIT. ON-LINE INDICATOR
JSR PC,DRVCAL ;READ ALL DRIVE STATUS
MOVB #40,BADDRV+7 ;PUT A SPACE IN MSG BUF
;SEE IF THIS DRIVE EXISTS AND IS ON-LINE
MOV #ERRHDL,A.ABNL ;RESTORE ABNL RETURN ADDRESS
CMP #377,NEWON ;SEE IF MED INDICATION ON THIS DRIVE
BEQ 4$ ;BR IF MED NOT SET
MOVB DRIVE,BADDRV+6 ;GET DRIVE NO. INTO BUF
BISB #'0,BADDRV+6 ;CONVERT TO ASCII
TYPE ,BADDRV ;TYPE "DRIVE X"
TYPE ,NXDRIV ;TYPE "NON-EXISTENT"
;SERVICE ERRORS HERE
2$: BIC #IE,RKCS1(R2) ;DISABLE RK06 INTERRUPT
MOV @2(SP),(SP) ;SET UP ERROR RETURN ADDRESS
RTS PC ;ERROR RETURN
;SEE IF DRIVE IS READY
4$: BIT #S.DRY,P.A00(R5) ;TEST FOR DRIVE READY
BNE 6$ ;BR IF DRIVE IS READY
MOVB DRIVE,BADDRV+6 ;GET DRIVE NO. INTO BUF
BISB #'0,BADDRV+6 ;CONVERT TO ASCII
TYPE ,BADDRV ;TYPE "DRIVE X"
TYPE ,NTREDY ;TYPE "NOT READY"
BR 2$ ;TAKE ERROR EXIT
```

```

4858      :SEE IF DRIVE IS WRITE ENABLED
4859 021214 032765 004000 000040 6S: BIT #5,WRL,P.AOD(RS) :SEE IF WRITE LOCK SET
4860 021222 001413 BEQ 8S :BR IF WRITE LOCK NOT SET
4861 021224 113737 005420 011061 MOVB DRIVE,BADDRV+6 :GET DRIVE NO.
4862 021232 152737 000060 011061 BLSB #'0,BADDRV+6 :CONVERT TO ASCII
4863 021240 104401 011053 TYPE ,BADDRV :TYPE "DRIVE X"
4864 021244 104401 011117 TYPE ,WRTLOK :TYPE "WRITE-LOCKED"
4865 021250 000734 BR 2S :TAKE ERROR EXIT
4866      :SEE IF DRIVE NOT LOADED WITH ALIGNMENT CARTRIDGE
4867 021252 112765 000103 000001 8S: MOVB #PACK,P.CMND(RS) :SET PACK ACKNOWLEDGE COMMAND
4868 021260 004737 027302 JSR PC,DRVCL :SET VOLUME VALID
4869 021264 112765 000113 000001 MOVB #RECAL,P.CMND(RS) :SET RECAL CMND
4870 021272 004737 027302 JSR PC,DRVCL :RECAL THE DRIVE
4871 021276 112765 000121 000001 MOVB #RODATA,P.CMND(RS) :SET READ COMMAND
4872 021304 012765 000632 000002 MOV #LSTCYL,P.CYLN(RS) :SET CYLINDER = 632(8)
4873 021312 112765 000002 000005 MOVB #LSTTRK,P.TRCK(RS) :SET TRACK = 2
4874 021320 012765 052476 000010 MOV #RMBUF,P.BALO(RS) :BUS ADDRESS
4875 021326 012765 177774 000012 MOV #-4,P.WC(RS) :READ 4 WORDS
4876 021334 105065 000007 CLR8 P.CS1H(RS) :SET 22-SECTOR FORMAT
4877 021340 004737 027302 JSR PC,DRVCL :READ 4 WORDS OF BAD SECTOR FILE
4878 021344 032737 100000 005414 BIT #ANYDER,RECODE :SEE IF DATA ERROR
4879 021352 001402 BEQ 10S :BR IF OK
4880 021354 104401 011606 TYPE ,BAD632 :TYPE READ ERROR MESSAGE
4881 021360 022737 177777 052504 10S: CMP #177777,RMBUF+6 :SEE IF ALL 1'S IN I.D. WORD 3
4882 021366 001013 BNE 12S :BR IF NOT ALL 1'S
4883 021370 113737 005420 011061 MOVB DRIVE,BADDRV+6 :GET DRIVE NO.
4884 021376 152737 000060 011061 BLSB #'0,BADDRV+6 :CONVERT TO ASCII
4885 021404 104401 011053 TYPE ,BADDRV :TYPE "DRIVE X"
4886 021410 104401 011136 TYPE ,ALNPAK :TYPE "LOADED WITH ALIGN PACK"
4887 021414 000652 BR 2S :TAKE ERROR EXIT
4888      :ERROR FREE RETURN
4889 021416 062716 000002 12S: ADD #2,(SP) :FIX UP RETURN PC
4890 021422 000207 RTS PC :RETURN
4891
4892
4893      :*****
4894      :*SETUP - SET UP FOR LOOP ON ERROR
4895      :*THIS SUBROUTINE CANNOT BE CALLED BY ANY OTHER
4896      :*SUBROUTINE --- ONLY MAIN-LINE CODE !!!!
4897      :*****
4898 021424 011637 001076 SETUP: MOV (SP),@#STACK-2 :MOVE RETURN PC ON STACK
4899 021430 012706 001076 MOV #STACK-2,SP :RE-INIT THE STACK POINTER
4900 021434 005037 005414 CLR RECODE :CLEAR ERROR RECOVERY FLAGS
4901 021440 105037 003125 CLR8 ERRCNT :CLEAR RETRY ERROR COUNT
4902 021444 012705 002630 MOV #PARAM,RS :SET PARAM BLK ADRS
4903 021450 112765 000177 000001 MOVB #SUBCLA,P.CMND(RS) :SET SUBSYSTEM CLEAR CMND
4904 021456 004737 027302 JSR PC,DRVCL :CLEAR THE SUBSYSTEM
4905 021462 012737 000000 177776 MOV #PRD,@#PS :RE-ESTABLISH PRIORITY 0
4906 021470 000207 RTS PC :RETURN
4907
4908
4909
4910
4911
4912      :*****
4913      :SBTTL MTCART - MOUNT TEST CARTRIDGE, CHECK DRIVE STATUS

```

```

4914
4915
4916
4917
4918
4919
4920 021472
4921 021472 013700 005544
4922 021476 112037 010136
4923 021502 111037 010135
4924 021506 162700 005502
4925 021512 006200
4926 021514 110037 003144
4927 021520 004737 020616
4928 021524 042762 000100 000000
4929 021532 104401 010110
4930 021536 104401 010352
4931 021542 104401 010373
4932 021546 004737 023006
4933 021552 021576
4934 021554 021620
4935 021556 021536
4936 021560 005700
4937 021562 001025
4938 021564 104401 005202 65:
4939 021570 104401 001324
4940 021574 000760
4941 021576 012706 001100 125:
4942 021602 105737 003116
4943 021606 001002
4944 021610 000137 011746
4945 021614 000137 012546 145:
4946 021620 105737 003116 165:
4947 021624 001744
4948 021626 012706 001100
4949 021632 000137 013212
4950 021636 023727 005202 000122 85:
4951 021644 001347
4952 021646 004737 020616
4953
4954 021652 117737 163666 005420
4955 021660 142737 000060 005420
4956 021666 004737 020736
4957 021672 021532
4958
4959
4960 021674 004737 020616
4961 021700 112765 000125 000001
4962 021706 004737 027302
4963 021712 012762 000025 000000
4964 021720 032762 000200 000000 245:
4965 021726 001774
4966 021730 012762 010025 000000
4967 021736 032762 000200 000000 265:
4968 021744 001774
4969 021746 105037 003124

```

```

; *THIS SUBROUTINE PERFORMS THE FOLLOWING FUNCTIONS :
; *IT TYPES "STARTING PASS X", DIRECTS THE OPERATOR TO MOUNT THE
; *TEST CARTRIDGE, CHECKS THE DRIVE STATUS, DETERMINES THE PACK FORMAT,
; *READS THE BAD SECTOR FILES, AND TYPES THE DRIVE AND CART. SERIAL
; *NUMBERS.
; *****
MTCART:
MOV DRVPTR,RO ;GET POINTER TO FIRST DRIVE FOR THIS SUBSYS
MOVB (RO)+,MNTPAK+22. ;PUT DRIVE NAME INTO MSG
MOVB (RO),MNTPAK+21.
SUB #DRVLST,RO ;SUBTRACT LIST ADRS FROM POINTER
ASR RO ;DIVIDE BY TWO TO GET DRIVE NO.
MOVB RO,LOGDRV ;STORE LOGICAL DRIVE NUMBER
JSR PC,INITSS ;INITIALIZE SUBSYSTEM
BIC #IE,RKCSI(R2) ;DISABLE RK06 INTERRUPT FOR MAN. INTERVENTION
105: TYPE MNTPAK ;TYPE MOUNT PACK MSG
55: TYPE ,TYPRUN ;TYPE "TYPE R(CR) WHEN "
TYPE ,DRIRDY ;TYPE "DRIVE READY : "
JSR PC,ROCHRS ;WAIT FOR R(CR) RESPONSE
125: ;(1C) RETURN ADDRESS
165: ;(1Z) RETURN ADDRESS
55: ;(1U) RETURN ADDRESS
TST RO ;SEE IF NULL INPUT
BNE 85 ;BR IF NOT NULL
65: TYPE ,BUFFD ;ECHO BAD INPUT
TYPE ,SQUES ;TYPE '?' AND (CR), (LF)
BR 55 ;GO ASK AGAIN
125: MOV #STACK,SP ;RE-INIT THE STACK
TSTB MOFLAG ;SEE IF 200 START
BNE 145 ;BR IF NOT
JMP DFSTRT ;RESTART PROGRAM - 200 START
145: JMP ALLSYS ;GO ASK FOR SUBSYS'S AND DRIVES AGAIN
165: TSTB MOFLAG ;SEE IF 200 START
BEQ 55 ;BR IF YES, TO ASK AGAIN
MOV #STACK,SP ;RE-INIT THE STACK
JMP ASKSYS ;GO ASK FOR SUBSYS TO TEST AGAIN
85: CMP BUFFD,#'R ;SEE IF R(CR) TYPED
BNE 65 ;BR IF NOT
JSR PC,INITSS ;INITIALIZE S.S.
;CHECK STATUS OF DRIVE TO BE TESTED NEXT
MOVB #DRVPTR,DRIVE ;SET DRIVE NUMBER
BICB #'0,DRIVE ;STRIP THE ASCII
JSR PC,SCNDRV ;CHECK STATUS OF THIS DRIVE -
105: ; IF NOT VALID, TYPE MSG AND RETURN
; TO WAIT FOR R(CR) AGAIN
;READ THE HEADER ON SECTOR 0, TRACK 0, CYL 0, AND GET THE DRIVE FORMAT
JSR PC,INITSS ;INIT. DRIVER PARAMS AND S.S.
MOVB #ROHEAD,P.CMND(R5) ;SET READ HEADER COMMAND
JSR PC,DRVCAL ;DO A READ HEADER
MOV #25,RKCSI(R2) ;DO A READ HDR, WITH FMT BIT = 0
245: BIT #RDY,RKCSI(R2) ;WAIT FOR READY
BEQ 245
MOV #10025,RKCSI(R2) ;DO A READ HDR, WITH FMT BIT = 1
265: BIT #RDY,RKCSI(R2) ;WAIT FOR READY
BEQ 265
CLRB FORMAT ;INITIALIZE FORMAT BYTE TO 0

```

```

4970 021752 005762 000024          TST    RKDB(R2)          ;POP SILO ONE TIME
4971 021756 032762 001000 000024  BIT    #BIT9,RKDB(R2)  ;TEST FORMAT BIT IN HEADER WORD 2
4972 021764 001403                BEQ    28$              ;BR IF 22 SECTOR FORMAT
4973 021766 152737 000020 003124  BLSB  #B.CFMT,FORMAT  ;SET 20 SECTOR FORMAT FOR THIS DRIVE
4974 021774 004737 020616          JSR    PC,INITSS       ;INIT THE S.S.
4975 022000 004737 025676          JSR    PC,REDBSF       ;READ BAD SECTOR FILE FOR THIS DRIVE
4976 022004 105737 003116          TSTB  MDFLAG          ;SEE IF 200 START
4977 022010 001004                BNE    32$              ;BR IF NOT 200 START
4978 022012 122737 000061 003145  CMPB  #'1,PASSNO      ;SEE IF FIRST PASS
4979 022020 001012                BNE    30$              ;BR IF NOT FIRST PASS
4980 022022 004737 023320          JSR    PC,DRVSR        ;TYPE "DRIVE SER. NO. XXX"
4981 022026 004737 023440          JSR    PC,CRTSR        ;TYPE "CART. SER. NO. XXXXXXXXXXXX"
4982 022032 032777 000200 157100  BIT    #BIT7,JSWR      ;SEE IF BAD SECTORS SHOULD BE TYPED
4983 022040 001402                BEQ    30$              ;BR IF NOT
4984 022042 004737 026164          JSR    PC,TYPBSF       ;TYPE BAD SECTOR FILES
4985 022046 005037 005432          CLR   INTCHR           ;INIT. TTY INPUT CHAR BUFFER
4986 022052 000207                RTS    PC               ;RETURN

```

```

;*****
;SBTTL DMCART - DISMOUNT TEST CARTRIDGE
;THIS SUBROUTINE DIRECTS THE OPERATOR IN THE DISMOUNTING
;OF THE TEST CARTRIDGE ON THE DRIVE CURRENTLY UNDER TEST.
;*****

```

```

DMCART:
MOV    DRVPTA,R0          ;GET DRIVE LIST POINTER
MOVB   (R0)+,DISPAK+15.  ;PUT DRIVE NAME INTO MSG
MOVB   (R0),DISPAK+14.
JSR    PC,INITSS         ;INITIALIZE SUBSYS
BIC    #IE,RKCSI(R2)     ;DISABLE RK06 INTRPT FOR MANUAL INTERVENTION
TYPE   ,DISPAK           ;TYPE "UNLOAD DRIVE XX AND REMOVE PACK"
5$:    TYPE   ,TYPRWN      ;TYPE "TYPE R<CR> WHEN "
TYPE   ,DRIDON           ;TYPE "DONE : "
JSR    PC,ROCHRS         ;WAIT FOR R<CR> RESPONSE
12$:   12$              ;(1C) RETURN ADDRESS
1E$:   1E$              ;(1Z) RETURN ADDRESS
5$:    5$               ;(1U) RETURN ADDRESS
TST    R0                ;SEE IF NULL INPUT
BNE    8$                ;BR IF NOT NULL
6$:    TYPE   ,BUFFO      ;ECHO BAD INPUT
TYPE   ,SQUES           ;TYPE (<?> AND <CR>),(LF)
BR     5$                ;GO ASK AGAIN
12$:   MOV    #STACK,SP  ;RE-INIT THE STACK
TSTB  MDFLAG           ;SEE IF 200 START
BNE    14$              ;BR IF NOT
14$:   JMP    DFSTR      ;RESTART PROGRAM - 200 START
16$:   JMP    ALLSYS     ;GO ASK FOR SUBSYS'S AND DRIVES AGAIN
TSTB  MDFLAG           ;SEE IF 200 START
BEQ    5$                ;BR IF YES, TO ASK AGAIN
MOV    #STACK,SP       ;RE-INIT THE STACK
JMP    ASKSYS          ;GO ASK FOR SUBSYS TO TEST AGAIN
8$:    CMP    BUFFO,#'R  ;SEE IF R<CR> TYPED
BNE    6$                ;BR IF NOT
JSR    PC,INITSS       ;INITIALIZE S.S.
RTS    PC               ;RETURN

```

```

*****
:SBTTL INTBLD - INITIALIZE TABLED
:THIS SUBROUTINE LOADS TABLED WITH THE INITIAL PATTERN OF DRIVE NUMBERS
:WHICH IS THE CYLINDER BLOCK LAYOUT FOR THE FIRST CURRENT ZONE. IT USES
:THE STARTING DRIVE NUMBER LIST, STDLST, FOR THE FIRST COLUMN, AND
:INCREMENTS THESE DRIVE NUMBERS ACROSS THE TABLE, FORMING ROWS. FOR
:ANY DRIVE WHICH IS NOT PRESENT, THE ENTRY IS SET = 377.
*****
INTBLD: SAVREG          ;SAVE R0-R5
;LOAD THE FIRST COLUMN OF TABLED FROM STDLST
      MOV      #STDLST,R0  ;STARTING ADDR OF STDLST
      MOV      #TABLED,R1 ;STARTING ADDR OF TABLED
      MOV      #16,R3      ;SET COUNTER = 16
4$:   MOV      (R0),R1     ;LOAD A BYTE INTO TABLED
      ADD      #16,R1      ;INCREMENT TABLED POINTER
      DEC      R3          ;DECR COUNTER
      BNE     4$          ;BR IF NOT DONE YET
;GET INDEX OF FIRST ZERO ENTRY IN DRVLST INTO R1
      MOV      #DRVLST,R0 ;STARTING ADDR OF DRVLST
      CLR      R1          ;INIT INDEX TO 0
6$:   TST      (R0)+       ;SEE IF ENTRY = 0
      BEQ     8$          ;BR IF YES
      INC      R1          ;INCREMENT POINTER
      CMP      R1,#16.    ;SEE IF DONE LOOKING YET
      BLT     6$          ;BR IF NOT DONE YET
;INCREMENT THE DRIVE NOS. TO FORM THE ROWS
8$:   MOV      #TABLED,R0 ;INIT TABLED POINTER
16$:  MOV      #15,R4      ;INIT ROW ELEMENT COUNTER
      MOV      (R0),R3     ;INIT ROW ELEMENT VALUE
18$:  INC      R3          ;INCR ROW ELEMENT VALUE
      CMP      R3,#20     ;SEE IF ROW ELEMENT IS VALID
      BLT     20$         ;BR IF VALUE IS OK
      CLR      R3          ;RESET VALUE TO 0
20$:  CMP      R3,R1       ;SEE IF VALUE IS > ACTUAL DRIVES
      BLT     22$         ;BR IF VALUE OK
      MOV      #377,(R0)+ ;SET INVALID ENTRY = 377
      BR      24$         ;PROCEED
22$:  MOV      R3,(R0)+   ;SET VALUE IN TABLE
24$:  DEC      R4          ;DECR ROW ELEMENT COUNTER
      BNE     18$         ;BR IF NOT DONE WITH ROW YET
      CMP      R0,#TABLED+256. ;SEE IF DONE WITH TABLE YET
      BLO     16$         ;BR IF NOT DONE YET
;SET INVALID ELEMENTS OF FIRST COLUMN = 377
      MOV      #TABLED,R0 ;INIT COLUMN POINTER
      MOV      #16,R3      ;INIT COL. ELEMENT COUNTER
26$:  CMP      (R0),R1     ;SEE IF ELEMENT IS VALID
      BLT     30$         ;BR IF YES
      MOV      #377,(R0) ;SET INVALID ELEMENT = 377
30$:  ADD      #16,R0      ;INCREMENT COLUMN POINTER
      DEC      R3          ;SEE IF ALL DONE YET
      BNE     26$         ;BR IF NOT ALL DONE YET
      RESREG          ;RESTORE R0-R5
      RTS             ;RETURN

```

5026			
5027			
5028			
5029			
5030			
5031			
5032			
5033			
5034			
5035			
5036			
5037	022224	104407	
5038			
5039	022226	012700	006410
5040	022232	012701	005750
5041	022236	012703	000020
5042	022242	112011	
5043	022244	062701	000020
5044	022250	005303	
5045	022252	001373	
5046			
5047	022254	012700	005502
5048	0	005001	
5049	0	005720	
5050	022264	001404	
5051	0	005201	
5052	0	00127	000020
5053	022274	002772	
5054			
5055	022276	012700	005750
5056	022302	012704	000017
5057	022306	112003	
5058	022310	005203	
5059	022312	020327	000020
5060	022316	002401	
5061	022320	005003	
5062	022322	020301	
5063	022324	002403	
5064	022326	112720	000377
5065	022332	000401	
5066	022334	110320	
5067	022336	005304	
5068	022340	001363	
5069	022342	020027	006350
5070	022346	103755	
5071			
5072	022350	012700	005750
5073	022354	012703	000020
5074	022360	121001	
5075	022362	002402	
5076	022364	112710	000377
5077	022370	062700	000020
5078	022374	005303	
5079	022376	001370	
5080	022400	104410	
5081	022402	000207	

5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5130
5131
5132
5133
5134
5135
5136
5137

022404 104407
022406 012700 006350
022412 010001
022414 012704 000015
022420 114037 001304
022424 114037 001306
022430 114037 001310
022434 114041
022436 115304
022440 101375
022442 113741 001304
022446 113741 001306
022452 113741 001310
022456 020027 005750
022462 101354
022464 104410
022466 000207

022470 104407
022472 113700 003144
022476 006300
022500 016003 006350
022504 012701 005750
022510 012704 000020
022514 010100
022516 123720 003144
022522 001375
022524 005300
022526 160100
022530 116065 006430 000004
022536 105065 000005
022542 004737 025126
022546 105265 000005
022552 126527 000005 000002
022560 003770
022562 062701 000020

```
*****  
: SBTTL ROTBLD - ROTATE TABLED RIGHT 3 COLUMNS  
: *THIS SUBROUTINE ROTATES ALL ELEMENTS OF TABLED RIGHT 3 COLUMNS,  
: *SO THAT THE DRIVE DATA WILL BE ROTATED SEVERAL SECTORS IN EACH  
: *CURRENT ZONE.  
*****  
ROTBLD: SAVREG ; SAVE RO-RS  
      MOV      #TABLED+256.,R0 ; POINT TO END OF TABLED  
      MOV      R0,R1 ; GET A COPY IN R1  
4$:   MOV      #13.,R4 ; INIT ROW ELEMENT COUNTER  
      MOV      -(R0),$TMP11 ; SAVE LAST 3 ELEMENTS IN THIS ROW  
      MOV      -(R0),$TMP12  
      MOV      -(R0),$TMP13  
6$:   MOV      -(R0),-(R1) ; MOVE AN ELEMENT RIGHT 3 PLACES  
      DEC      R4 ; DECR ROW ELEMENT COUNTER  
      BNE      6$ ; BR IF NOT DONE YET  
      MOV      $TMP11,-(R1) ; MOVE IN 3 SAVED ELEMENTS  
      MOV      $TMP12,-(R1)  
      MOV      $TMP13,-(R1)  
      CMP      R0,#TABLED ; SEE IF DONE WITH TABLE YET  
      BHI      4$ ; BR IF NOT DONE YET  
      RESREG ; RESTORE RO-RS  
      RTS      PC ; RETURN  
*****  
: SBTTL WRTBLK - WRITE ONE CYLINDER BLOCK  
: *THIS SUBROUTINE WRITES ALL THE SECTORS FOR THE CURRENT DRIVE  
: *INTO THE CYLINDER BLOCK WHOSE STARTING CYL NUMBER IS IN  
: *P.CYLN(R5) ON ENTRY. THE DATA IS WRITTEN ON ALL SURFACES  
: *THE LOGICAL DRIVE NUMBER MUST BE IN LOGDRV ON ENTRY.  
: *  
: *IT CALLS WRTSEC TO ACTUALLY WRITE EACH 256-WORD SECTOR.  
*****  
WRTBLK: SAVREG ; SAVE RO-RS  
      MOV      LOGDRV,R0 ; GET LOGICAL DRIVE NO.  
      ASL      R0 ; CONVERT IT TO DATA PATTERN INDEX  
      MOV      TABLEG(R0),R3 ; GET DATA PATTERN WORD INTO R3  
      MOV      #TABLED,R1 ; GET TABLED POINTER  
      MOV      #16.,R4 ; INIT CYLINDER COUNTER  
4$:   MOV      R1,R0  
6$:   CMPB     LOGDRV,(R0)+ ; SEE IF DRIVE LISTED HERE  
      BNE      6$ ; BR IF NOT FOUND YET IN THIS ROW  
      DEC      R0 ; BACK UP BY 1 BYTE  
      SUB      R1,R0 ; COMPUTE ROW POSITION  
      MOV      SECT(R0),P.SECT(R5) ; GET SECTOR NO. FROM ROW POSITION  
      CLRB     P.TRCK(R5) ; SET TRACK = 0  
8$:   JSR      PC,WRTSEC ; WRITE 256 WORDS AT THIS SECTOR  
      INCB     P.TRCK(R5) ; INCR TRACK NO.  
      CMPB     P.TRCK(R5),#2 ; SEE IF ALL TRACKS DONE YET  
      BLE      8$ ; BR IF NOT DONE YET  
      ADD      #16.,R1 ; POINT TO START OF NEXT ROW  
*****
```


M08

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 103
WRTRLK - WRITE ONE CYLINDER BLOCK

SEQ 0102

5138 022566 005265 000002
5139 022572 005304
5140 022574 001347
5141 022576 104410
5142 022600 000207
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158

INC P.CYLN(R5) ; INCR CYLINDER NO.
DEC R4 ; DECREMENT CYLINDER COUNTER
BNE 48 ; BR IF 16 CYLS NOT DONE YET
RESREG ; RESTORE R0-R5
RTS PC ; RETURN

```

*****
;CKSCOR - DECREMENT DRIVE SCORES ROUTINE
;THIS SUBROUTINE CHECKS TO SEE IF THE PROGRAM IS CURRENTLY
;READING OVRWRT OR COMPAT DATA WITH OFFSETS AND HANDLING DATA CHECK
;ERRORS. IF SO, THE SCORE FOR THE DRIVE WHICH WROTE THE
;FAILING DATA IS DECREMENTED.
;ON ENTRY, THE BYTE LABELED "TSTTYP" HAS THE FOLLOWING
;POSSIBLE VALUES :
*   TSTTYP = 0 - NOT PRESENTLY READING WITH OFFSET
*           = 1 - PRESENTLY READING OVRWRT DATA WITH OFFSET
*           = 2 - PRESENTLY READING COMPAT DATA WITH OFFSET
*           = 3 - PRESENTLY READING SELF TEST DATA WITH OFFSET
*****
    
```

5159 022602 104407
5160 022604 105737 003133
5161 022610 001474
5162 022612 032737 000024 005414
5163 022620 001470
5164
5165 022622 004737 033444
5166 022626 122737 000003 003133
5167 022634 001002
5168 022636 005004
5169 022640 000431
5170 022642 013701 001174
5171 022646 042701 177760
5172 022652 006301
5173 022654 006301
5174 022656 006301
5175 022660 006301
5176 022662 062701 005750
5177 022666 005000
5178 022670 123760 001200 006430
5179 022676 001405
5180 022700 005200
5181 022702 020027 000020
5182 022706 002770
5183 022710 000434
5184
5185 022712 060001
5186 022714 111104
5187 022716 122704 000377
5188 022722 001427
5189 022724 012700 007270
5190 022730 105737 003146
5191 022734 001402
5192 022736 012700 007350
5193 022742 105737 001176

```

CKSCOR: SAVREG ; SAVE R0-R5
        TSTB TSTTYP ; SEE IF CURRENTLY TESTING WITH OFFSET
        BEQ 20$ ; BR TO EXIT, IF NOT
        BIT #DCKERR!HVRCER,RECODE ; SEE IF A DCK OR HVRC ERR OCCURRED
        BEQ 20$ ; BR IF NOT
; GET THE NO. OF THE DRIVE WHICH WROTE DATA WHICH CAUSED DCK OR HVRC ERR
        JSR PC,GTPKAD ; COMPUTE PACK ADDRESS OF ERROR
        CMPB #3,TSTTYP ; SEE IF DOING SELF-TEST
        BNE 4$ ; BR IF NOT
        CLR R4 ; SET DRIVE NO. = 0
        BR 10$ ; CONTINUE
4$: MOV $REG5,R1 ; GET CYL NO. OF ERROR
    BIC #177760,R1 ; CLEAR HI CYL BITS
    ASL R1 ; CONVERT TO TABLED INDEX FOR ROW
    ASL R1
    ASL R1
    ADD #TABLED,R1 ; COMPUTE POINTER TO TABLED ROW
    CLR R0 ; INIT SECLST INDEX TO 0
6$: CMPB $REG7,SECLST(R0) ; SEE IF SECTOR IS LISTED IN SECLST
    BEQ 8$ ; BR IF YES
    INC R0 ; INCR SECLST INDEX
    CMP R0,#16. ; SEE IF WHOLE LIST CHECKED YET
    BLT 6$ ; BR IF NOT YET
    BR 20$ ; SECTOR NOT LISTED - EXIT, BECAUSE
    ; FAILING DRIVE IS NOT BEING TESTED
8$: ADD R0,R1 ; COMPUTE TABLED POINTER TO FAILING DRIVE
    MOVB (R1),R4 ; GET ACTUAL DRIVE NO.
    CMPB #377,R4 ; SEE IF DRIVE IS NOT BEING TESTED
    BEQ 20$ ; BR IF DRIVE NOT TESTED
10$: MOV #NSCOR,R0 ; GET ADRS OF NEG SCRATCH SCORES
    TSTB OFSDIR ; SEE WHICH OFST DIRECTION
    BEQ 12$ ; BR IF NEG
    MOV #PSCOR,R0 ; GET ADRS OF POS SCRATCH SCORES
12$: TSTB $REG6 ; SEE IF TRACK = 0
    
```

```

5194 022746 001410
5195 022750 062700 000020
5196 022754 123727 001176 000001
5197 022762 001402
5198 022764 062700 000020
5199 022770 060400 14$:
5200 022772 121037 003147
5201 022776 001001
5202 023000 105310
5203 023002 104410
5204 023004 000207

```

```

BEQ 14$ ;BR IF YES
ADD #16, R0 ;POINT TO SCORES FOR TRACK 1
CMPB $REG6, #1 ;SEE IF TRACK = 1
BEQ 14$ ;BR IF YES
ADD #16, R0 ;POINT TO SCORES FOR TRACK 2
ADD R4, R0 ;COMPUTE POINTER TO SCORE FOR DRIVE
CMPB (R0), OFSCNT ;SEE IF THIS SCORE IS PERFECT SO FAR
BNE 20$ ;EXIT IF NOT
DECB (R0) ;DECREMENT SCORE FOR THIS DRIVE
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

```

5205
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5230

```

```

;*****
;SBTTL RDCHRS - READ A STRING OF KBD INPUT CHARS
;THIS SUBROUTINE READS A STRING OF UP TO EIGHTY INPUT
;CHARACTERS AT THE KBD, TERMINATED BY (CR), AND PLACES
;THEM IN BUFF0, TERMINATED BY A NULL (0) BYTE.
;RUB-OUT AND (↑) FEATURES ARE PROVIDED.
;IMMEDIATELY FOLLOWING THE SUBROUTINE CALL, THREE SPECIAL
;RETURN ADDRESSES MUST BE LISTED: THE FIRST RETURN IS
;TAKEN IF (↑C) IS TYPED, THE SECOND IS TAKEN IF (↑Z) IS
;TYPED, AND THE THIRD IS TAKEN IF (↑U) OR INVALID INPUT IS TYPED.
;IF (↑G) IS TYPED, THE SOFTWARE SWITCH REGISTER IS OPENED
;FOR MODIFICATION, IF SELECTED, AND THEN THE (↑G) RETURN
;IS TAKEN.
;THE NUMBER OF INPUT CHARACTERS IN THE BUFFER IS RETURNED
;IN R0.
;
; CALL - JSR PC, RDCHRS
; <CONTROL-C RETURN ADDRESS>
; <CONTROL-Z RETURN ADDRESS>
; <CONTROL-U OR ERROR RETURN ADDRESS>
; RETURN
;*****

```

```

5231 023006
5232 023006 010146
5233 023010 010246
5234 023012 005000
5235 023014 005001
5236
5237 023016 104406
5238 023020 112602
5239
5240 023022 122702 000003
5241 023026 001006
5242 023030 104401 011652
5243 023034 017666 000004 000004
5244 023042 000523
5245
5246 023044 122702 000032
5247 023050 001006
5248 023052 104401 011657
5249 023056 062766 000002 000004

```

```

RDCHRS:
MOV R1, -(SP) ;SAVE R1
MOV R2, -(SP) ;SAVE R2
CLR R0 ;INITIALIZE CHARACTER COUNT
CLR R1 ;INITIALIZE RUB-OUT INDICATOR
;READ A CHARACTER
2$: RDCHR ;READ A CHARACTER
MOVB (SP)+, R2 ;GET CHARACTER INTO R2
;CHECK FOR (↑C)
CMPB #003, R2 ;SEE IF (↑C) TYPED
BNE 4$ ;BR IF NOT (↑C)
TYPE CNTRLC ;ECHO (↑C)
3$: MOV #4(SP), 4(SP) ;PUT RETURN ADDRESS ON STACK
BR 24$ ;BR TO TAKE EXIT
;CHECK FOR (↑Z)
4$: CMPB #032, R2 ;SEE IF (↑Z) TYPED
BNE 6$ ;BR IF NOT (↑Z)
TYPE CNTRLZ ;ECHO (↑Z)
ADD #2, 4(SP) ;MAKE OLD PC POINT TO NEXT RETURN ADR.

```

```

5250 023064 000763          BR      3$          ;BR TO TAKE (IZ) EXIT
5251          .CHECK FOR (II)
5252 023066 122702 000025 6$:  CMPB  #025,R2      ;SEE IF (IU) TYPED
5253 023072 001006          BNE  8$          ;BR IF NOT (IU)
5254 023074 104401 011671          TYPE  CNTRLU      ;ECHO (IU)
5255 023100 062766 000004 000004 7$:  ADD   #4,4(SP)      ;MAKE OLD PC POINT TO NEXT RETURN ADDR.
5256 023106 000752          BR      3$          ;BR TO TAKE (IU) EXIT
5257          .CHECK FOR (IG)
5258 023110 122702 000007 8$:  CMPB  #007,R2      ;SEE IF (IG) TYPED
5259 023114 001005          BNE  9$          ;BR IF NOT (IG)
5260 023116 104401 011676          TYPE  CNTRLG      ;ECHO (IG)
5261 023122 004737 020564          JSR   PC,GTSWRG    ;OPEN SOFTWARE SWITCH REG. FOR CHANGE
5262 023126 000764          BR      7$          ;TAKE (IU) RETURN
5263          .CHECK FOR RUB-OUT (DELETE)
5264 023130 122702 000177 9$:  CMPB  #177,R2      ;SEE IF RUB-OUT (DEL) TYPED
5265 023134 001020          BNE  14$         ;BR IF NOT RUB-OUT
5266 023136 005700          TST   R0          ;CHECK THE CHARACTER COUNT
5267 023140 001726          BEQ  2$          ;BR IF COUNT = 0
5268 023142 005701          TST   R1          ;CHECK THE RUB-OUT INDICATOR
5269 023144 001003          BNE  11$         ;BR IF WE HAD A PREVIOUS RUB-OUT
5270 023146 005201          INC   R1          ;SET RUB-OUT INDICATOR
5271 023150 104401 011707          TYPE  BKSLSH      ;TYPE A BACK-SLASH (\)
5272 023154 005037 005434 11$:  CLR   SCRACH      ;USE SCRATCH WORD FOR TEMP. BUFFER
5273 023160 005300          DEC   R0          ;DECREMENT COUNT
5274 023162 116037 005202 005434  MOVB  BUFFO(R0),SCRACH ;GET LAST CHAR. INTO BUFFER
5275 023170 104401 005434          TYPE  SCRACH      ;ECHO CHARACTER TO BE DELETED
5276 023174 000710          BR      2$          ;GO READ ANOTHER CHARACTER
5277 023176 005701          TST   R1          ;CHECK THE RUB-OUT INDICATOR
5278 023200 001403          BEQ  16$         ;BR IF INDICATOR IS NOT SET
5279 023202 104401 011707          TYPE  BKSLSH      ;TYPE A BACK-SLASH
5280 023206 005001          CLR   R1          ;CLEAR THE RUB-OUT INDICATOR
5281          .CHECK FOR CARRIAGE RETURN
5282 023210 122702 000015 16$:  CMPB  #015,R2      ;SEE IF (CR) TYPED
5283 023214 001426          BEQ  19$         ;BR IF (CR)
5284          ;HANDLE POSSIBLE DIGIT
5285 023216 005037 005434          CLR   SCRACH      ;USE SCRATCH WORD FOR TEMP. BUFFER
5286 023222 110237 005434          MOVB  R2,SCRACH   ;GET THIS CHARACTER INTO BUFFER
5287 023226 104401 005434          TYPE  SCRACH      ;ECHO THE CHARACTER TYPED
5288 023232 110260 005202          MOVB  R2,BUFFO(R0) ;PUT CHARACTER INTO BUFFER
5289 023236 005200          INC   R0          ;INCREMENT CHARACTER COUNTER
5290 023240 022700 000120          CMP   #80.,R0     ;SEE IF TOO MANY CHARACTERS TYPED
5291 023244 001264          BNE  2$          ;IF NOT TOO MANY
5292 023246 104401 001325          TYPE  SCRLF      ;TYPE (CR) AND (LF)
5293 023252 112760 000000 005202  MOVB  #0,BUFFO(R0) ;PUT TERMINATING NULL INTO BUFFER
5294 023260 104401 005202          TYPE  'BUFFO      ;ECHO INPUT STRING
5295 023264 104401 001324          TYPE  'QUES      ;TYPE ( ? ), (CR) (LF)
5296 023270 000703          BR      7$          ;TAKE ERROR EXIT FROM ROCHRS
5297 023272 104401 001325 19$:  TYPE  SCRLF      ;TYPE (CR) (LF)
5298 023276 112760 000000 005202  MOVB  #0,BUFFO(R0) ;PUT TERMINATING NULL INTO BUFFER
5299 023304 062766 000006 000004  ADD   #6,4(SP)     ;FIX UP RETURN ADDRESS
5300 023312 012602          MOV   (SP)+,R2    ;RESTORE R2
5301 023314 012601          MOV   (SP)+,R1    ;RESTORE R1
5302 023316 000207          RTS   PC          ;SUBROUTINE EXIT
5303
5304
5305

```

```

5306
5307
5308
5309
5310
5311 023320 104407
5312 023322 004737 020616
5313 023326 112765 000141 000001
5314 023334 004737 027302
5315 023340 104401 011501
5316 023344 104401 011517
5317 023350 016501 000054
5318 023354 012704 042312
5319 023360 010446
5320 023362 012703 000003
5321 023366 006101
5322 023370 006101
5323 023372 006101
5324 023374 006101
5325 023376 006101
5326 023400 006101
5327 023402 010100
5328 023404 042700 177760
5329 023410 052700 000060
5330 023414 110024
5331 023416 005303
5332 023420 001364
5333 023422 105014
5334 023424 004737 042524
5335 023430 104401 001325
5336 023434 104410
5337 023436 000207
5338
5339
5340
5341
5342
5343
5344
5345
5346 023440 004737 020616
5347 023444 105065 000007
5348 023450 105737 003124
5349 023454 001402
5350 023456 105265 000004
5351 023462 112765 000121 000001
5352 023470 012765 000632 000002
5353 023476 112765 000002 000005
5354 023504 012765 052476 000010
5355 023512 012765 177776 000012
5356 023520 004737 027302
5357 023524 104401 011510
5358 023530 104401 011517
5359 023534 012746 052476
5360 023540 004737 042210
5361 023544 004737 042524

```

```

*****
:SBTTL DRVSR - TYPE DRIVE SERIAL NUMBER (LOW 3 DIGITS)
:*THIS SUBROUTINE TYPES "DRIVE SER. NO. XXX" (IN DECIMAL), WITH LEADING
:*ZEROS SUPPRESSED.
*****
DRVSR: SAVREG
JSR PC,INITSS ;SAVE R0-R5
MOV B,R0STAT,P.CHND(R5) ;CLEAR S.S. AND PARAMETERS
JSR PC,DRVCAL ;SET READ STATUS COMMAND
TYPE 'DRIV' ;READ STATUS OF THIS DRIVE
TYPE 'SERNM' ;TYPE "DRIVE"
MOV P,A11(R5),R1 ;TYPE "SER. NO. "
MOV #S0CTLV,R4 ;GET "R" STATUS BYTE 11
MOV R4,-(SP) ;GET ADDR OF CHAR BUFFER
MOV #3,R3 ;STORE IT ON STACK FOR $SUPRS
ROL R1 ;INIT CHAR COUNT
ROL R1 ;INITIALIZE BIT POSITIONS
45: ROL R1 ;GET NEXT 4 BITS
ROL R1
ROL R1
ROL R1
MOV R1,R0 ;GET A WORKING COPY
BIC #177760,R0 ;CLEAR ALL BUT LOW 4 BITS
BIS #0,R0 ;CONVERT A DIGIT TO ASCII
MOVB R0,(R4)+ ;PUT ASCII DIGIT INTO CHAR BUFFER
DEC R3 ;DECREMENT CHAR COUNT
BNE 45 ;BR IF NOT 3 CHARS YET
CLRB (R4) ;INSERT NULL TERMINATOR
JSR PC,$SUPRS ;TYPE DRIVE SER. NUMBER
TYPE 'CR' AND 'LF'
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

```

*****
:SBTTL CRTSR - TYPE CARTRIDGE SERIAL NUMBER
:*THIS SUBROUTINE TYPES "CART. SER. NO. XXXXXXXXXXX" (IN OCTAL),
:*WITH LEADING ZEROS SUPPRESSED.
*****
CRTSR: JSR PC,INITSS ;CLEAR S.S. AND PARAMETERS
CLRB P,CS1H(R5) ;SET 22-SECTOR FORMAT
TSTB FORMAT ;CHECK THE ACTUAL FORMAT
BZ 45 ;BR IF 22 SECTORS
INCB P,SECT(R5) ;IF 20 SECTORS, READ SECTOR 1
45: MOV B,RDATA,P.CHND(R5) ;SET READ COMMAND
MOV #LSTCYL,P.CYLN(R5) ;SET CYL = 632(OCT)
MOVB #LSTTRK,P.TRCK(R5) ;SET TRACK = 2
MOV #RWBUF,P.BALO(R5) ;SET READ BUFFER ADDRESS
MOV #2,P.WC(R5) ;SET WORD COUNT TO READ 2 WORDS
JSR PC,DRVCAL ;READ SERIAL NO. IN BSF
TYPE 'CART.' ;TYPE "CART."
TYPE 'SERNM' ;TYPE "SER. NO. "
MOV #RWBUF,-(SP) ;GET POINTER FOR $DB20
JSR PC,$DB20 ;CONVERT BINARY TO OCTAL
JSR PC,$SUPRS ;TYPE CART. SERIAL NO. IN OCTAL

```

5362 023550 104401 001325
5363 023554 000207

TYPE SCRLF ;TYPE <CR> AND <LF>
RTS PC ;RETURN

5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5375
5376
5377

:SBTTL STALL - STALL FOR ST UNIT STALL TIMES
:*IF SWR BIT 8 = 0, THIS SUBROUTINE STALLS FOR ST STALL TIMES,
:*WHERE A STALL TIME = 40 US, FOR AN "AVERAGE" CPU. IF BIT 8
:*IS EQUAL TO 1, A RANDOM STALL IS APPLIED.
:* CALL - JSR PC,STALL
:*****

5378 023556 010046
5379 023560 010146
5380 023562 032777 000400 155350
5381 023570 001407
5382 023572 004737 041406
5383 023576 013700 041506
5384 023602 006200
5385 023604 006200
5386 023606 000403
5387 023610 013700 005422
5388 023614 001406
5389 023616 012701 000016
5390 023622 005301
5391 023624 001376
5392 023626 005300
5393 023630 001372
5394 023632 012601
5395 023634 012600
5396 023636 000207

STALL: MOV RO,-(SP) ;SAVE RO
MOV R1,-(SP) ;SAVE R1
BIT #BIT08,SWR ;APPLY RANDOM STALL ?
BEQ 1\$;BR IF NOT RANDOM
JSR PC,SRAND ;GENERATE PSEUDO-RANDOM NUMBER
MOV \$LONUM,R0 ;GET IT INTO R0
ASR RO ;SCALE IT DOWN
ASR RO
BR 2\$;GO STALL WITH RANDOM NO.
1\$: MOV STALLS,R0 ;GET REQUESTED NO. OF STALLS
BEQ 6\$;RETURN IF NO STALL REQUIRED
2\$: MOV #14.,R1 ;SET CONSTANT FOR 40 US
4\$: DEC R1 ;INNER LOOP COUNTER
BNE 4\$;INNER LOOP BR
DEC R0 ;OUTER LOOP COUNTER
BNE 2\$;OUTER LOOP BR
6\$: MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R0 ;RESTORE R0
RTS PC ;RETURN

5397
5398
5399

5400
5401
5402
5403

:LODSEC - THIS SUBROUTINE LOADS THE CONTENTS OF THE DATA PATTERN
:IN TABLEB INTO ALL 256(DEC) WORDS OF THE DATA BUFFER (RMBUF).
:*****

5404 023640 104407
5405 023642 012701 052476
5406 023646 012700 005656
5407 023652 012703 000020
5408 023656 012021
5409 023660 020127 053476
5410 023664 103003
5411 023666 005303
5412 023670 001372
5413 023672 000765
5414 023674 104410
5415 023676 000207

LODSEC: SAVREG ;SAVE RO-R5
MOV #RMBUF,R1 ;GET ADDS OF R/W BUF INTO R1
2\$: MOV #TABLEB,R0 ;GET DATA PATTERN ADDRESS
MOV #16.,R3 ;INIT PATTERN COUNTER
4\$: MOV (R0)+,(R1)+ ;MOVE A PATTERN WORD
CMP R1,#RMBUF+512. ;SEE IF DONE YET
BHS 9\$;BR IF DONE
DEC R3 ;DECREMENT PATTERN COUNTER
BNE 4\$;BR IF NOT RESTARTING PATTERN YET
BR 2\$;BR TO RESTART PATTERN
9\$: RESREG ;RESTORE RO-R5
RTS PC ;RETURN

5416
5417

5418
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5470
5471
5472
5473

023700 104407
023702 013746 005464
023706 005416
023710 005046
023712 116616 000003
023716 005066 000002
023722 116566 000004 000002
023730 066616 000002
023734 005066 000002
023740 012700 000026
023744 105737 003124
023750 001402
023752 012700 000024
023756 020016
023760 101004
023762 160016
023764 005266 000002
023770 000772
023772 112637 005463
023776 005046
024000 116516 000005
024004 066616 000002
024010 005066 000002
024014 122716 000003
024020 101005
024022 162716 000003
024026 005266 000002
024032 000770
024034 112637 005462
024040 066516 000002
024044 011637 005460
024050 005726
024052 104410
024054 000207

```
*****  
: SBTTL FINADR - COMPUTE FINAL PACK ADDRESS  
: *THIS SUBROUTINE IS USED AFTER A DATA TRANSFER HAS COMPLETED, TO  
: *COMPUTE THE FINAL PACK ADDRESS AT TERMINATION. THE 2'S COMP. OF THE  
: *WORD NO. AT THE POINT OF INTEREST IS PASSED IN LASTWC. THIS IS USED WITH  
: *P.CYLN(R5), P.TRCK(R5), AND P.SECT(R5) TO COMPUTE THE CORRESPONDING  
: *PACK ADDRESS, WHICH IS RETURNED IN FINCYL, FINTRK, AND FINSEC.  
: *THE SUBROUTINE IS USED TO DETERMINE THE PACK ADDRESS OF A SOFTWARE  
: *DATA MISCOMPARE.  
*****  
FINADR: SAVREG  
MOV LASTWC, -(SP) ; SAVE R0-R5  
NEG (SP) ; STORE WORD COUNT  
18$: CLR -(SP) ; MAKE IT POSITIVE  
MOV 3(SP), (SP) ; MAKE ROOM ON STACK  
CLR 2(SP) ; STORE NO. OF SECTORS TRANSFERRED  
MOV P.SECT(R5), 2(SP) ; CLEAR LOCATION ON STACK  
ADD 2(SP), (SP) ; STORE STARTING SECTOR  
CLR 2(SP) ; DETERMINE FINAL SECTOR ADDRESS  
MOV #22, R0 ; CLEAR NO. OF TRACKS TRANSFERRED  
TSTB FORMAT ; SET FOR 22 SECTORS  
BEQ 19$ ; DETERMINE THE FORMAT  
MOV #20, R0 ; BR IF 22 SECTORS  
19$: CMP R0, (SP) ; SET FOR 20 SECTORS  
BHI 20$ ; CHECK FOR SECTOR OVERFLOW  
SUB R0, (SP) ; NO. CHECK IF SECTOR CORRECT  
INC 2(SP) ; DECREMENT SECTOR COUNT BY 20 OR 22  
BR 19$ ; INCREMENT TRACKS TRANSFERRED  
20$: MOV (SP)+, FINSEC ; CHECK FOR SECTOR OVERFLOW  
CLR -(SP) ; STORE FINAL SECTOR  
MOV P.TRCK(R5), (SP) ; MAKE ROOM FOR TRACKS TRANSFERRED  
ADD 2(SP), (SP) ; STORE STARTING TRACK  
CLR 2(SP) ; DETERMINE FINAL TRACK ADDRESS  
21$: CMPB #3, (SP) ; CLEAR FINAL CYLINDER  
BHI 22$ ; CHECK FOR TRACK OVERFLOW  
SUB #3, (SP) ; NO. CHECK FINAL TRACK  
INC 2(SP) ; DECREMENT TRACK COUNT BY 3  
BR 21$ ; INCR CYL COUNT  
22$: MOV (SP)+, FINTRK ; CHECK FOR TRACK OVERFLOW  
ADD P.CYLN(R5), (SP) ; STORE FINAL TRACK  
MOV (SP), FINCYL ; CALCULATE FINAL CYLINDER  
TST (SP)+ ; STORE FINAL CYLINDER  
RESREG ; CLEAN OFF STACK  
RTS PC ; RESTORE R0-R5  
 ; RETURN
```

```
*****  
: SBTTL LOOBUF - LOAD THE READ/WRITE DATA BUFFER  
: *THIS SUBROUTINE LOADS THE READ/WRITE DATA BUFFER (POINTED TO BY  
: * PMA AND PMA+2) WITH THE APPROPRIATE REPEATING DATA PATTERN. IF  
: *PARAMETER PATRN = 0 ON ENTRY, THE DATA WHICH IS LOADED IS COMPRISED
```

```

5474
5475
5476
5477
5478
5479
5480 024056 104407
5481 024060 013702 005456
5482 024064 005737 005436
5483 024070 001007
5484 024072 012700 005556
5485 024076 012746 000400
5486 024102 012701 052476
5487 024106 000422
5488 024110 005000
5489 024112 032701 000001
5490 024116 001003
5491 024120 005200
5492 024122 006201
5493 024124 000772
5494 024126 006300
5495 024130 006300
5496 024132 006300
5497 024134 006300
5498 024136 006300
5499 024140 062700 005556
5500 024144 012746 000020
5501 024150 013701 005474
5502 024154 010003
5503 024156 011604
5504 024160 012321
5505 024162 005302
5506 024164 001403
5507 024166 005304
5508 024170 001373
5509 024172 000770
5510 024174 005726
5511 024176 104410
5512 024200 000207
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525 024202 104407
5526 024204 112737 000040 051021
5527 024212 012737 000007 052370
5528 024220 013702 005456
5529 024224 005037 005434

```

```

; *OF ALL THE PATRNS 00-15 (QUICK VERIFY DEFAULT DATA TEST).
; *IF PATRN IS NOT 0, THE NO. OF WORDS IN THE WORD WDSXFR
; *OF THE PATTERN IDENTIFIED BY THE NO. OF THE BIT SET IN R1 ON
; *ENTRY, ARE LOADED INTO THE BUFFER.
; *****

```

```

LOOBUF: SAVREG                ; SAVE R0-R5
MOV    WDSXFR,R2             ; GET NO. OF WORDS
TST    PATRN                 ; SEE IF QUICK VERIFY DATA TEST DESIRED
BNE    3$                   ; BR IF NOT QUICK VERIFY
MOV    #PAT00,R0             ; SET DATA PATTERN STARTING ADDRESS
MOV    #256,-(SP)           ; SET PATTERN WORD COUNT
MOV    #RBUF,R1              ; SET BUFFER ADDRESS
JR     30$                   ; PROCEED
3$:   CLR    R0               ; INIT PATTERN NUMBER
4$:   BIT    #BIT0,R1         ; SEE IF THIS BIT IS SET
BNE    6$                   ; BR IF THIS BIT IS SET
INC    R0                   ; INCREMENT PATTERN NO.
ASR    R1                   ; SHIFT TO EXAMINE NEXT BIT
BR     4$                   ; BR TO CHECK NEXT BIT
6$:   ASL    R0               ; MULTIPLY PATTERN NO. BY 32(DEC)
ASL    R0
ASL    R0
ASL    R0
ADD    #PAT00,R0             ; GET ADDRESS OF DESIRED PATTERN
MOV    #16,-(SP)            ; SET PATTERN WORD COUNT
MOV    #PMA,R1              ; SET BUFFER ADDRESS
30$:  MOV    R0,R3            ; GET A COPY OF PATTERN ADDRESS
MOV    (SP),R4              ; INIT PATTERN WORD COUNT
34$:  MOV    (R3)+,(R1)+     ; LOAD A DATA WORD INTO BUFFER
DEC    R2                   ; DECREMENT WORD COUNTER
BEQ    44$                 ; BR IF ALL DONE
DEC    R4                   ; DECREMENT PATTERN WORD COUNT
BNE    34$                 ; BR IF NOT DONE WITH PATTERN YET
BR     30$                 ; BR TO REPEAT THE PATTERN
44$:  TST    (SP)+          ; POP THE STACK
RESREG                       ; RESTORE R0-R5
RTS    PC                   ; RETURN

```

```

; *****
; SBTTL CMPBUF - SOFTWARE COMPARE DATA
; *THIS SUBROUTINE PERFORMS A SOFTWARE COMPARE OF THE R/W DATA BUFFER (POINTED
; *TO BY PMA AND PMA+2) TO THE APPROPRIATE REPEATING DATA PATTERN. IF
; *PARAMETER PATRN = 0 ON ENTRY, THE DATA IS COMPRISED OF ALL THE PATRNS
; *00-15 (QUICK VERIFY DEFAULT DATA TEST).
; *IF PATRN IS NOT 0, THE NO. OF THE BIT SET IN R1 ON ENTRY IDENTIFIES THE
; *REPEATING DATA PATTERN TO COMPARE AGAINST.
; *****

```

```

CMPBUF: SAVREG                ; SAVE R0-R5
MOVB   #40,DH701+38.        ; RESTORE ERROR MSG PARAMS
MOV    #7,DF25+2
MOV    WDSXFR,R2             ; SET NO. OF WORDS
CLR    SCRACH                ; CLEAR COMPARE ERROR COUNT

```

5530	024230	004737	030360		JSR	PC, REPSUP	; STORE PREV CMND FOR POSS. PRINT	
5531	024234	005737	005436		TST	PATRN	; SEE IF THIS IS QUICK VERIFY DEFAULT DATA TEST	
5532	024240	001007			BNE	3\$; BR IF NOT	
5533	024242	012700	005556		MOV	#PAT00, R0	; GET DATA PATTERN STARTING ADDR	
5534	024246	012746	000400		MOV	#256, -(SP)	; SET PATTERN WORD COUNT	
5535	024252	012701	052476		MOV	#RWBUFF, R1	; SET BUFFER ADDRESS	
5536	024256	000422			BR	30\$; PROCEED	
5537	024260	005000		3\$:	CLR	R0	; INIT PATTERN NO.	
5538	024262	032701	000001	4\$:	BIT	#BIT0, R1	; SEE IF THIS BIT IS SET	
5539	024266	001003			BNE	6\$; BR IF THIS BIT IS SET	
5540	024270	005200			INC	R0	; INCREMENT PATTERN NUMBER	
5541	024272	006201			ASR	R1	; SHIFT TO EXAMINE NEXT BIT	
5542	024274	000772			BR	4\$; BR TO CHECK NEXT BIT	
5543	024276	006300		6\$:	ASL	R0	; MULTIPLY PATTERN NO. BY 32(DEC)	
5544	024300	006300			ASL	R0		
5545	024302	006300			ASL	R0		
5546	024304	006300			ASL	R0		
5547	024306	006300			ASL	R0		
5548	024310	062700	005556		ADD	#PAT00, R0	; GET ADDRESS OF DESIRED PATTERN	
5549	024314	012746	000020		MOV	#16, -(SP)	; PATTERN WORD COUNT	
5550	024320	013701	005474		MOV	PMA, R1	; SET BUFFER ADDRESS	
5551	024324	010003		30\$:	MOV	R0, R3	; GET COPY OF PATTERN ADDRESS	
5552	024326	011604			MOV	(SP), R4	; INIT PATTERN WORD COUNT	
5553	024330	022321		34\$:	CMP	(R3)+, (R1)+	; COMPARE DATA WORD TO PATTERN WORD	
5554	024332	001002			BNE	31\$; BR IF COMPARE ERROR	
5555	024334	000137	024674		JMP	40\$; JUMP IF DATA COMPARES OK	
5556	024340	105037	051021	31\$:	CLRB	DH701+38.	; ADJUST DATA HEADER FOR MSG	
5557	024344	012737	000005	052370	MOV	#5, DF25+2	; ADJUST ERROR DATA WORD COUNT	
5558					:COMMON	COMPARE	ERROR HANDLER	
5559	024352	010237	001202		35\$:	MOV	R2, \$REG10	; GET WORD NO.
5560	024356	163737	005456	001202	SUB	WD5XFR, \$REG10		
5561	024364	013737	001202	005464	MOV	\$REG10, LASTWC	; GET 2'S COMP OF WORD NO.	
5562	024372	004737	023700		JSR	PC, FINADR	; COMPUTE ACTUAL PACK ADRS	
5563	024376	104407			SAVREG		; SAVE R0-R5	
5564	024400	013700	005460		MOV	FINCYL, R0	; GET CYL	
5565	024404	113701	005462		MOVB	FINTRK, R1	; GET TRACK	
5566	024410	113702	005463		MOVB	FINSEC, R2	; GET SECTOR	
5567	024414	004737	027176		JSR	PC, BDSACK	; SEE IF THIS SECTOR LISTED BAD	
5568	024420	104410			RESREG		; RESTORE R0-R5	
5569	024422	032737	001000	005414	BIT	#BADSEC, RECODE		
5570	024430	001116			BNE	50\$; BR IF LISTED- DON'T REPORT ERROR	
5571	024432	005737	005434		TST	SCRACH	; CHECK THE ERROR COUNT	
5572	024436	001010			BNE	36\$; BR IF THIS IS NOT FIRST ERROR	
5573	024440	104034		46\$:	ERROR	34	; TYPE HEADING FOR ERROR MSG	
5574	024442	012737	177777	001174	48\$:	MOV	#-1, \$REG5	; INIT CYL NO.
5575	024446	005037	001176		CLR	\$REG6		
5576	024454	005037	001200		CLR	\$REG7		
5577	024460	005237	005434	36\$:	INC	SCRACH	; INCREMENT THE ERROR COUNT	
5578	024464	032777	000001	154446	BIT	#BIT0, 2SWR	; SEE IF ALL ERRORS SHOULD BE REPORTED	
5579	024472	001004			BNE	38\$; BR TO REPORT ALL ERRORS	
5580	024474	022737	000012	005434	CMP	#10, SCRACH	; SEE IF 10(DEC) ERRORS YET	
5581	024502	002504			BLT	54\$; BR IF ERROR LIMIT EXCEEDED	
5582	024504	023737	001174	005460	38\$:	CMP	\$REG5, FINCYL	; SEE IF DIFFERENT CYL
5583	024512	001010			BNE	42\$; BR IF YES	
5584	024514	123737	001176	005462	CMPB	\$REG6, FINTRK	; SEE IF DIFFERENT TRACK	
5585	024522	001004			BNE	42\$; BR IF YES	


```

5586 024524 123737 001200 005463      CMPB   $REG7,FINSEC      ;SEE IF DIFFERENT SECTOR
5587 024532 001412                BEQ    44$                ;BR IF SAME PACK ADDRESS
5588 024534 013737 005460 001174      MOV    FINCYL,$REG5      ;SET NEW PACK ADRS FOR PRINTOUT
5589 024542 113737 005462 001176      MOVB   FINTRK,$REG6
5590 024550 113737 005463 001200      MOVB   FINSEC,$REG7
5591 024556 104115                ERROR  11$                ;TYPE NEW PACK ADDRESS
5592 024560 005437 001202                NEG    $REG10            ;GET WORD NO.
5593 024564 016337 177776 001204      MOV    -2(R3),$REG11     ;GET GOOD DATA
5594 024572 016137 177776 001206      MOV    -2(R1),$REG12     ;GET BAD DATA
5595 024600 013737 001202 001212      MOV    $REG10,$REG14     ;COMPUTE PHYSICAL ADDRESS
5596 024606 005037 001210      CLR    $REG13
5597 024612 006137 001212      ROL    $REG14
5598 024616 006137 001210      ROL    $REG13
5599 024622 063737 005474 001212      ADD    PMA,$REG14
5600 024630 005537 001210      ADC    $REG13
5601 024634 063737 005476 001210      ADD    PMA+2,$REG13
5602 024642 010137 001214                MOV    R1,$REG15        ;GET VIRT. ADRS FOR PRINTOUT
5603 024646 162737 000002 001214      SUB    #2,$REG15
5604 024654 104063                ERROR  63                ;TYPE GOOD AND BAD DATA, MEM. ADRS.
5605 024656 032777 000100 154254      BIT    #BIT6,$SMR        ;SEE IF JUST 1 ERROR SHOULD BE REPORTED
5606 024664 001013                BNE   54$                ;BR IF JUST 1 ERROR SHOULD BE REPORTED
5607 024666 005737 005436      50$:  TST    PATRN            ;SEE IF DEFAULT DATA TEST
5608 024672 001400                BEQ   40$                ;BR IF YES
5609 024674 005302      40$:  DEC    R2                ;DECREMENT WORD COUNTER
5610 024676 001406                BEQ   54$                ;BR IF ALL DONE
5611 024700 005304                DEC    R4                ;DECREMENT PATTERN WORD COUNT
5612 024702 001002                BNE   53$                ;BR IF NOT DONE WITH PATTERN YET
5613 024704 000137 024324      JMP    30$                ;JUMP TO REPEAT THE PATTERN
5614 024710 000137 024330      53$:  JMP    34$
5615 024714                54$:
5616 024714 005726      56$:  TST    (SP)+            ;POP THE STACK
5617 024716 104410                RESREG ;RESTORE R0-R5
5618 024720 000207                RTS    PC                ;RETURN
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5630
5631
5632 024722 122737 000001 003117  CTLOUT: CMPB   #1,TSTING      ;SEE IF CURRENTLY RUNNING TESTS
5633 024730 001075                BNE   10$                ;BR IF NOT RUNNING TESTS
5634 024732 005737 005432      TST    INTCHR            ;SEE IF ANY TTY INPUT
5635 024736 001472                BEQ   10$                ;BR IF NO INPUT
5636 024740 105737 003116      TSTB   MOFLAG            ;SEE IF DEFAULT MODE RUN
5637 024744 001441                BEQ   12$                ;BR IF YES
5638 024746 122737 000003 005432      CMPB   #003,INTCHR       ;SEE IF (↑C) TYPED
5639 024754 001026                BNE   4$                 ;BR IF NOT (↑C)
5640 024756 012700 012546      MOV    #ALLSYS,R0        ;SET RETURN ADDR = ALLSYS
5641 024762
2$:

```

```

*****
* CTLOUT - THIS SUBROUTINE CHECKS FOR (↑C) OR (↑Z) TTY INPUT.
* IF (↑C) WAS TYPED, THE SUBROUTINE RETURNS TO DAVIST. IF (↑Z)
* WAS TYPED, THE SUBROUTINE RETURNS TO INPUT. IN EITHER CASE,
* A RESET INSTRUCTION IS EXECUTED, AND THE STACK IS RE-INIT-
* IALIZED. IF THERE IS NO INPUT, OR INPUT OTHER THAN (↑C) OR (↑Z),
* NO ACTION IS TAKEN.
* CALL - JSR PC,CTLOUT
* OR - CKEXIT
*****

```

5648	024762	000005			6\$:	RESET		; RESET ALL DEVICES
5649	024764	005037	005432			CLR INTCHR		; CLEAR TTY CHAR BUFFER WORD
5644	024770	105037	003117			CLRB TSTING		; CLEAR TTY ESCAPE ALLOW FLAG
5645	024774	012737	030640	003046		MOV #ERRHDL, A.ABNL		; RESTORE ERROR HANDLER ADDRESS
5646	025002	012706	001100			MOV #STACK, SP		; RESET THE STACK
5647	025006	112765	000113	000001		MOV#B #RECAL, P.CMND(R5)		; SET RECAL COMMAND
5648	025014	004737	027302			JSR PC, DRVCAL		; DO CLEANUP RECALIBRATE
5649	025020	105037	003133			CLRB TSTTYP		; CLEAR OFFSET TEST IDENTIFIER
5650	025024	005037	001102			CLR \$TSTNM		; CLEAR THE TEST NO.
5651	025030	000110				JMP \$RO		; EXIT FROM TESTS
5652	025032	122737	000032	005432	4\$:	CMPB #032, INTCHR		; SEE IF (1Z) TYPED
5653	025040	001021				BNE 7\$; BR IF NOT (1Z)
5654	025042	012700	013212			MOV #ASKSYS, RO		; SET RETURN ADDR = ASKSYS
5655	025046	000745				BR 2\$; TAKE EXIT
5656	025050	122737	000003	005432	12\$:	CMPB #003, INTCHR		; SEE IF (1C) TYPED
5657	025056	001012				BNE 7\$; BR IF NOT
5658	025060	104401	011337			TYPE ,HLTRQD		; TYPE "HALT REQUESTED"
5659	025064	104401	011307			TYPE ,CNTROY		; TYPE "PRESS CONT WHEN RDY"
5660	025070	012700	032676			MOV #HALTPRG, RO		; SET HALT ADDRESS
5661	025074	112737	000061	003145		MOV#B #1, PASSNO		; RESTORE PASS NO. TO 1
5662	025102	000727				BR 2\$; TAKE EXIT
5663	025104	122737	000007	005432	7\$:	CMPB #007, INTCHR		; SEE IF (1G) TYPED
5664	025112	001002				BNE 8\$; BR IF NOT (1G)
5665	025114	004737	020564			JSR PC, GTSWRG		; OPEN SOFTWARE SWR FOR MODIFICATION
5666	025120	004737	020422		8\$:	JSR PC, PREPKB		; ENABLE KBD INPUT AGAIN
5667	025124	000207			10\$:	RTS PC		; RETURN

5668
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697

```

*****
;WRITE A BLOCK OF 400(OCT) WORDS ONTO THE DISK.
;THE PARAMETER BLOCK MUST BE PRE-LOADED WITH THE PACK
;ADDRESS. DATA WORD IS PASSED IN R3 ON ENTRY. RWBUF IS THE BUFFER
;USED.
*****
;WRITE THE DATA
;LOAD THE R/W BUFFER WITH DATA
;SET UP PARAMETERS
;WRITE THE DATA
;RESTORE RO-R5
;RETURN
*****
;LDMEM1 - LOAD MEMORY WITH INCREASING NUMBERS, STARTING WITH 1.
;LDMEM2 - LOAD MEMORY WITH REPEATED WORD, IN R3 ON ENTRY.

```

```

5698
5699
5700
5701 025200 104407
5702 025202 005004
5703 025204 012703 000001
5704 025210 000403
5705 025212 104407
5706 025214 012704 177777
5707 025220 013702 005456
5708 025224 013701 005474
5709 025230 010321
5710 025232 005704
5711 025234 001001
5712 025236 005203
5713 025240
5714 025240 005302
5715 025242 001372
5716 025244 104410
5717 025246 000207
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729 025250 104407
5730 025252 005004
5731 025254 012703 000001
5732 025260 000403
5733 025262 104407
5734 025264 012704 177777
5735 025270 013702 005456
5736 025274 013701 005474
5737 025300 005037 005434
5738 025304 112737 000040 051021
5739 025312 012737 000007 052370
5740 025320 004737 030360
5741 025324 020321
5742 025326 001552
5743
5744 025330 010237 001202
5745 025334 163737 005456 001202
5746 025342 013737 001202 005464
5747 025350 004737 023700
5748 025354 104407
5749 025356 013700 005460
5750 025362 113701 005462
5751 025366 113702 005463
5752 025372 004737 027176
5753 025376 104410

```

```

; *BOTH SUBROUTINES REQUIRE MEMORY ADDRESS IN PMA, PMA+2, AND WORD COUNT
; *MUST BE IN WDSXFR.
; *****
LDMEM1: SAVREG ; SAVE R0-R5
CLR R4 ; R4=0 INDICATES LDMEM1
MOV #1,R3 ; INIT DATA WORD TO 1
BR LDNO ; PROCEED
LDMEM2: SAVREG ; SAVE R0-R5
MOV #-1,R4 ; R4=177777 INDICATES LDMEM2
LDNO: MOV WDSXFR,R2 ; GET NO. OF WORDS
MOV PMA,R1 ; GET MEM ADRS
BS: MOV R3,(R1)+ ; LOAD A WORD INTO BUFFER
TST R4 ; SEE WHICH DATA DESIRED
BNE BS ; BR IF NOT INCREMENTING DATA
INC R3 ; INCREMENT THE DATA

BS:
10$: DEC R2 ; DECREMENT COUNTER
BNE BS ; BR IF NOT DONE LOADING YET
12$: RESREG ; RESTORE R0-R5
RTS PC ; RETURN

```

```

; *****
; *CKMEM1 - PERFORM SOFTWARE COMPARE OF MEMORY, WITH INCREASING
; *NUMBERS, STARTING WITH 1.
; *CKMEM2 - PERFORM SOFTWARE COMPARE OF MEMORY, WITH REPEATED DATA WORD
; *IN R3 ON ENTRY.
; *BOTH SUBROUTINES REQUIRE MEM ADRS IN PMA, PMA+2, AND WORD COUNT MUST BE
; *IN WDSXFR.
; *****
CKMEM1: SAVREG ; SAVE R0-R5
CLR R4 ; R4=0 INDICATES CKMEM1
MOV #1,R3 ; INIT DATA TO 1
BR CKNO ; PROCEED
CKMEM2: SAVREG ; SAVE R0-R5
MOV #-1,R4 ; R4=177777 INDICATES CKMEM2
CKNO: MOV WDSXFR,R2 ; GET NO. OF WORDS
MOV PMA,R1 ; GET MEM ADRS
CLR SCRACH
MOV #40,DH701+38. ; RESTORE ERROR MSG PARAMS
MOV #7,DF25+2
JSR PC,REPSUP ; STORE PREV CMND FOR POSS. PRINT
BS: CMP R3,(R1)+ ; COMPARE A DATA WORD
BEQ BS ; BR IF DATA COMPARES OK

; COMPARE ERROR HANDLER
MOV R2,$REG10 ; GET WORD NO.
SUB WDSXFR,$REG10
MOV $REG10,LASTWC ; GET 2'S COMP OF WORD NO.
JSR PC,FINADR ; COMPUTE ACTUAL PACK ADRS
SAVREG ; SAVE R0-R5
MOV FINCYL,R0 ; GET CYL
MOV FINTRK,R1 ; GET TRACK
MOV FINSEC,R2 ; GET SECTOR
JSR PC,BDSACK ; SEE IF THIS SECTOR LISTED BAD
RESREG ; RESTORE R0-R5

```

5754	025400	032737	001000	005414		BIT	#BADSEC, RECODE	
5755	025406	001122				BNE	16\$: BR IF LISTED- DON'T REPORT ERROR
5756	025410	105037	051021			CLRB	DH701+38.	: ADJUST DATA HEADER FOR MSG
5757	025414	012737	000005	052370		MOV	#5, DF25+2	: ADJ. ERROR DATA WORD COUNT
5758	025422	005737	005434		9\$:	TST	SCRACH	: SEE IF FIRST ERROR IN THIS BLOCK
5759	025426	001010				BNE	10\$: BR IF NOT FIRST ERROR
5760	025430	104034			17\$:	ERROR	34	: TYPE HEADING FOR ERROR MSG
5761	025432	012737	177777	001174	11\$:	MOV	#-1, \$REG5	: INIT CYL NO.
5762	025440	005037	001176			CLR	\$REG6	
5763	025444	005037	001200			CLR	\$REG7	
5764	025450	005237	005434		10\$:	INC	SCRACH	: INCREMENT THE ERROR COUNT
5765	025454	032777	000001	153456		BIT	#BIT0, #SWR	: SEE IF ALL ERRORS SHOULD BE REPORTED
5766	025462	001004				BNE	12\$: BR TO REPORT ALL ERRORS
5767	025464	022737	000012	005434		CMP	#10., SCRACH	: SEE IF 10(DEC) ERRORS YET
5768	025472	002477				BLT	21\$: BR IF ERROR LIMIT EXCEEDED
5769	025474	023737	001174	005460	12\$:	CMP	\$REG5, FINCYL	: SEE IF DIFFERENT CYL
5770	025502	001010				BNE	14\$: BR IF YES
5771	025504	123737	001176	005462		CMP#	\$REG6, FINTRK	: SEE IF DIFFERENT TRACK
5772	025512	001004				BNE	14\$: BR IF YES
5773	025514	123737	001200	005463		CMP#	\$REG7, FINSEC	: SEE IF DIFFERENT SECTOR
5774	025522	001412				BEQ	15\$: BR IF SAME PACK ADDRESS
5775	025524	013737	005460	001174	14\$:	MOV	FINCYL, \$REG5	: SET NEW PACK ADR FOR PRINTOUT
5776	025532	113737	005462	001176		MOVB	FINTRK, \$REG6	
5777	025540	113737	005463	001200		MOVB	FINSEC, \$REG7	
5778	025546	104115				ERROR	11\$: TYPE NEW PACK ADRS
5779	025550	005437	001202		15\$:	NEG	\$REG10	: GET WORD NO.
5780	025554	010337	001204			MOV	R3, \$REG11	: GOOD DATA
5781	025560	016137	177776	001206		MOV	-2(R1), \$REG12	: BAD DATA
5782	025566	013737	001202	001212		MOV	\$REG10, \$REG14	: COMPUTE PHYSICAL ADDRESS
5783	025574	005037	001210			CLR	\$REG13	
5784	025600	006137	001212			ROL	\$REG14	
5785	025604	006137	001210			ROL	\$REG13	
5786	025610	063737	005474	001212		ADD	PMA, \$REG14	
5787	025616	005537	001210			ADC	\$REG13	
5788	025622	063737	005476	001210		ADD	PMA+2, \$REG13	
5789	025630	010137	001214			MOV	R1, \$REG15	: GET VIRT ADRS FOR PRINTOUT
5790	025634	162737	000002	001214		SUB	#2, \$REG15	
5791	025642	104063				ERROR	63	: TYPE GOOD AND BAD DATA
5792	025644	032777	000100	153266		BIT	#BIT6, #SWR	: SEE IF JUST 1 ERROR SHOULD BE REPORTED
5793	025652	001007				BNE	21\$: BR IF JUST 1
5794						: PROCEED WITH COMPARISONS		
5795	025654	005704			16\$:	TST	R4	: SEE WHICH DATA DESIRED
5796	025656	001001				BNE	18\$: BR IF NOT INCREASING DATA
5797	025660	005203				INC	R3	: INCREMENT THE DATA WORD
5798	025662				18\$:			
5799	025662	005302			20\$:	DEC	R2	: DECR COUNTER
5800	025664	001402				BEQ	21\$: BR IF DONE COMPARING
5801	025666	000137	025324			JMP	6\$: CONTINUE COMPARING WORDS
5802	025672				21\$:			
5803	025672	104410			22\$:	RESREG		: RESTORE R0-R5
5804	025674	000207				RTS	PC	: RETURN
5805								
5806								
5807								
5808								
5809								

```

5810 ;*****
5811 ;*REDBSF - READ FACTORY BSF INTO BSFACT, AND READ SOFTWARE BSF
5812 ;*INTO BSSOFT.
5813 ;*****
5814 REDBSF:
5815 JSR PC INITSS ;INIT THE S.S.
5816 CLR B P.CS1H(R5) ;SET 22-SECTOR FORMAT
5817 MOVB #RD0DATA,P.CMD(R5) ;SET READ DATA COMMAND
5818 MOV #632,P.CYLN(R5) ;SET CYL = 632
5819 MOVB #2,P.TRCK(R5) ;SET TRACK = 2
5820 MOV #R3 ;SET FACTORY BSF SECTOR LIMIT
5821 TSTB FORMAT
5822 BEQ 6$ ;BR IF 22 SECTORS
5823 INCB P.SECT(R5) ;SET STARTING FACTORY BSF SECTOR NO.
5824 INC R3
5825 MOV #400,P.WC(R5) ;SET WORD COUNT FOR 1 SECTOR
5826 MOV #BSFACT,P.BALO(R5) ;SET BA FOR FACTORY BSF
5827 MOV #BDSCHD,A.ABNL ;SET ERROR HANDLER FOR BSF READ
5828 CLR B WCEFLG ;INIT THE ERROR FLAG
5829 JSR PC DRVCL ;READ THE FACTORY BSF
5830 TSTB WCEFLG ;SEE IF ANY ERRORS
5831 BEQ 12$ ;BR IF NOT
5832 ADD #2,P.SECT(R5) ;CHECK NEXT SECTOR
5833 CMPB P.SECT(R5),R3 ;SEE IF LIMIT EXCEEDED YET
5834 BNE 8$ ;BR IF NOT YET
5835 JSR PC REPSUP ;GATHER STATUS FOR PRINTOUT
5836 ERROR 120 ;ABORTING- BAD BSF READ
5837 JMP HLT PRG ;HALT- CAN'T PROCEED
5838 MOV #BSSOFT,P.BALO(R5) ;SET BA FOR SOFTWARE RECORD
5839 MOVB R3,P.SECT(R5) ;SET STARTING SECTOR NO.
5840 MOV #22,R3 ;SET 22 SECTOR LIMIT
5841 TSTB FORMAT ;SEE IF 22 SECTOR FORMAT
5842 BEQ 14$ ;BR IF YES
5843 MOV #19,R3 ;SET 20 SECTOR LIMIT
5844 MOV #BDSCHD,A.ABNL ;SET ERROR HANDLER FOR BSF READ
5845 CLR B WCEFLG ;INIT THE ERROR FLAG
5846 JSR PC DRVCL ;READ THE SOFTWARE BSF
5847 TSTB WCEFLG ;SEE IF ANY ERRORS
5848 BEQ 16$ ;BR IF NOT
5849 ADD #2,P.SECT(R5) ;CHECK NEXT SECTOR
5850 CMPB P.SECT(R5),R3 ;SEE IF LIMIT EXCEEDED YET
5851 BLE 14$ ;BR IF NOT YET
5852 BR 10$ ;REPORT ERROR AND ABORT
5853 MOV #ERRHDL,A.ABNL ;RESTORE ERROR HANDLER ADRS
5854 RTS PC ;RETURN
5855
5856 ;*THIS IS THE ABNORMAL DRIVER RETURN WHICH IS USED TO HANDLE
5857 ;*POSSIBLE ERRORS IN READING BAD SECTORS.
5858 BOSCHD: BIT #BSE!HVRC!DTE!OPI!DCK,P.ER(R5) ;SEE IF ANY READ ERRORS
5859 BNE 4$ ;BR IF A READ ERROR OCCURRED
5860 JMP ERRHDL ;GO HANDLE OTHER ERROR
5861 INC B WCEFLG ;SET ERROR FLAG
5862 JMP RETNML ;TAKE NORMAL DRIVER RETURN
5863
5864
5865

```

```

5866
5867
5868
5869
5870
5871
5872 026164 104407
5873 026166 005004
5874 026170 012700 004162
5875 026174 104401 011532
5876 026200 104401 011557
5877 026204 104401 050566
5878 026210 104401 001325
5879 026214 005001
5880 026216 005710
5881 026220 100007
5882 026222 005701
5883 026224 001032
5884 026226 104401 011577
5885 026232 104401 001325
5886 026236 000425
5887 026240 012046
5888 026242 104402
5889 026244 104401 011717
5890 026250 011003
5891 026252 105003
5892 026254 000303
5893 026256 010346
5894 026260 104402
5895 026262 104401 011717
5896 026266 111003
5897 026270 010346
5898 026272 104402
5899 026274 104401 001325
5900 026300 005720
5901 026302 005201
5902 026304 020127 000176
5903 026310 002742
5904 026312 005704
5905 026314 001006
5906 026316 005204
5907 026320 012700 003162
5908 026324 104401 011545
5909 026330 000725
5910 026332 104410
5911 026334 000207
5912
5913
5914
5915
5916
5917
5918
5919 026336 104407
5920 026340 016500 000002
5921 026344 116501 000005

```

```

*****
;TYPBSF - TYPE FACTORY AND SOFTWARE BAD SECTOR FILES (BSF'S)
;WITH THE CYLINDER, TRACK, AND SECTOR NO. OF EACH BAD SECTOR
;LISTED SEPARATELY, IN A TABLE OF OCTAL VALUES, FOR EACH OF THE
;2 BSF'S.
*****
TYPBSF: SAVREG ;SAVE RO-R5
CLR R4 ;INIT B,F FILE INDICATOR
MOV #BSFACT+10,R0 ;ADRS OF DATA IN FACTORY BSF
TYPE ,FACTBS ;TYPE "FACTORY"
TYPE ,BSECT ;TYPE "BAD SECTORS"
3$: TYPE ,DH6041 ;TYPE "CYLNDR TRACK SECTOR"
TYPE ,SCRLF
CLR R1 ;INIT LIMIT COUNTER
4$: TST (R0) ;SEE IF A SECTOR LISTED HERE
BPL BS ;BR IF YES
TST R1 ;SEE IF FILE IS EMPTY
BNE 14$ ;BR IF NOT EMPTY
TYPE ,NOFALS ;TYPE "NONE"
TYPE ,SCRLF ;TYPE <CR>,<LF>
BR 14$
8$: MOV (R0)+,-(SP) ;GET A CYL NO.
TPOC ;TYPE IT
TYPE SPACE2 ;TYPE SEPARATORS
MOV (R0),R3 ;GET TRACK AND SECTOR
CLRB R3
SWAB R3 ;GET THE TRACK NO.
MOV R3,-(SP)
TPOC ;TYPE IT
TYPE SPACE2 ;TYPE SEPARATORS
MOVB (R0),R3 ;GET SECTOR NO.
MOV R3,-(SP)
TPOC ;TYPE IT
TYPE ,SCRLF
TST (R0)+ ;INCR THE POINTER
INC R1 ;INCR THE COUNTER
CMP R1,#126. ;SEE IF LIMIT REACHED IN THIS FILE
BLT 4$ ;BR IF NOT YET
14$: TST R4 ;SEE IF BOTH FILES TYPED YET
BNE 18$ ;BR IF YES
INC R4 ;SET INDICATOR FOR SOFT. FILE
MOV #BSSOFT+10,R0 ;ADRS OF DATA IN SOFTWARE BSF
TYPE ,SOFTBS ;TYPE "SOFTWARE BAD SECTORS:"
BR 3$ ;GO TYPE SOFT. BSF
18$: RESREG ;RESTORE RO-R5
RTS PC ;RETURN
*****
;FINMEM - COMPUTE NO. OF WORDS XFERRED ON PARTIAL XFER
;(TERMINATED BY BSE ERROR), AND STORE IT IN WDSXFR.
*****
FINMEM: SAVREG ;SAVE RO-R5
MOV P.CYLN(R5),R0 ;GET ORIG. CYL NO.
MOVB P.TRCK(R5),R1 ;GET TRACK NO.

```

5922	026350	116502	000004
5923	026354	005003	
5924	026356	026500	000030
5925	026362	001006	
5926	026364	126501	000027
5927	026370	001003	
5928	026372	126502	000026
5929	026376	001404	
5930	026400	005203	
5931	026402	004737	026422
5932	026406	000763	
5933	026410	000303	
5934	026412	010337	005456
5935	026416	104410	
5936	026420	000207	

```

MOV B P,SECT(R5),R2 ;SECTOR NO.
CLR R3 ;INIT SECTOR COUNT TO 0
65: CMP P,DCYL(R5),R0 ;COMPARE ORIG. AND CURRENT CYLS
BNE B$ ;BR IF NOT EQUAL
CMP B P,DTS+1(R5),R1 ;COMPARE TRACKS
BNE B$ ;BR IF NOT EQUAL
CMP B P,DTS(R5),R2 ;COMPARE SECTORS
BEQ 12$ ;BR IF ADRS ARE EQUAL
85: INC R3 ;INCR SECTOR COUNT
JSR PC,INCRSC ;INCR PACK ADRS BY 1 SECTOR
BR 6$ ;GO COMPARE ADDRESSES
12$: SWAB R3 ;COMPUTE NO. OF WORDS XFERRED
MOV R3,WDSXFR ;STORE IT
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

```

*****
*INCRSC - INCREMENT PACK ADDRESS TO NEXT SECTOR
*THE CYLINDER IS PASSED AND RETURNED IN R0, TRACK IN R1, AND SECTOR
*IN R2.
*****

```

5945	026422	012704	000025
5946	026426	105737	003124
5947	026432	001402	
5948	026434	012704	000023
5949	026440	005202	
5950	026442	020204	
5951	026444	003407	
5952	026446	005002	
5953	026450	005201	
5954	026452	020127	000002
5955	026456	003402	
5956	026460	005001	
5957	026462	005200	
5958	026464	000207	

```

*****
INCRSC: MOV #21,R4 ;SET SECTOR LIMIT
TSTB FORMAT ;SEE IF 22 SECTOR FORMAT
BEQ 4$ ;BR IF YES
MOV #19,R4 ;SET SECTOR LIMIT
45: INC R2 ;INCR. SECTOR NO.
CMP R2,R4 ;SEE IF SECTOR LIMIT EXCEEDED
BLE 6$ ;BR IF NOT
CLR R2 ;SET SECTOR = 0
INC R1 ;INCR. TRACK NO.
5954: CMP R1,#2 ;SEE IF TRACK LIMIT EXCEEDED
BLE 6$ ;BR IF NOT
CLR R1 ;SET TRACK = 0
INC R0 ;INCR. CYLINDER NO.
65: RTS PC ;RETURN

```

```

*****
*MIDXFR - MID-TRANSFER PARAMETER UPDATE SUBROUTINE
*THIS SUBROUTINE UPDATES PMA, PMA+2, P.BALO(R5),
*P.BAHI(R5), AND P.WC(R5), IN PREPARATION TO RESUME A TRANSFER
*TERMINATED BY A BAD SECTOR ERROR (BSE), A DATA CHECK ERROR (DCK),
*OR A HEADER VRC ERROR (HVRC).
*****

```

5969	026466	104407	
5970	026470	004737	026336
5971	026474	016500	000030
5972	026500	116501	000027
5973	026504	116502	000026
5974	026510	013703	005456
5975	026514	032737	000020 005414
5976	026522	001004	
5977	026524	062703	000400

```

MIDXFR: SAVREG ;SAVE R0-R5
JSR PC,FINMEM ;COMPUTE NO. OF WORDS XFERRED
MOV P,DCYL(R5),R0 ;GET CYL NO.
MOV B P,DTS+1(R5),R1 ;GET TRACK NO.
MOV B P,DTS(R5),R2 ;GET SECTOR NO.
MOV WDSXFR,R3 ;GET NO. OF WORDS TRANSFERRED
BIT #DCKERR,RECODE ;SEE IF DATA CHECK ERROR OCCURRED
BNE 9$ ;BR IF YES
ADD #400,R3 ;SKIP BAD SECTOR

```

```

5978 026530 004737 026422          JSR    PC,INCRSC      ;INCREMENT PACK ADRS TO NEXT SECTOR
5979 026534 010065 000002          9$:   MOV    R0,P.CYLN(RS) ;UPDATE CYLINDER
5980 026540 110165 000005          MOVB  R1,P.TRCK(RS)  ;UPDATE TRACK
5981 026544 110265 000004          MOVB  R2,P.SECT(RS)  ;UPDATE SECTOR
5982 026550 060365 000012          ADD   R3,P.WC(RS)    ;UPDATE P.WC(RS)
5983 026554 032765 100000 000014  BIT    NOTBAII,P.PRST(RS) ;SEE IF BUS ADRS INCR INHIB SET
5984 026562 001022          BNE   16$           ;BR IF YES
5985 026564 005004          CLR   R4            ;GET BYTES XFERRD
5986 026566 006103          ROL   R3
5987 026570 006104          ROL   R4
5988 026572 060337 005474          ADD   R3,PMA        ;UPDATE PMA,PMA+2
5989 026576 005537 005476          ADC   PMA+2
5990 026602 060437 005476          ADD   R4,PMA+2
5991 026606 013765 005474 000010  MOV   PMA,P.BALO(RS) ;UPDATE P.BALO(RS)
5992 026614 013700 005476          MOV   PMA+2,R0
5993 026620 042700 177774          BIC   #177774,R0
5994 026624 150065 000007          BISB  R0,P.BAHI(RS) ;UPDATE P.BAHI(RS)
5995 026630 104410          RESREG ;RESTORE R0-R5
5996 026632 000207          RTS    PC           ;RETURN
5997
5998
5999
6000
6001
6002
6003

```

```

*****
;SVPRMS - SAVE INITIAL PARAMETERS FOR TRANSFER
;GTPRMS - RESTORE SAVED TRANSFER PARAMETERS
*****

```

```

6004 026634 104407          SVPRMS: SAVREG      ;SAVE R0-R5
6005 026636 012700 002632          MOV   #PARMO+2,R0   ;ADRS OF PARAMS
6006 026642 012701 005444          MOV   #SAVPRS,R1    ;ADRS OF SAVE AREA
6007 026646 016537 000012 005456  MOV   P.WC(RS),WDSXFR ;GET WORD COUNT
6008 026654 005437 005456          NEG   WDSXFR        ;MAKE IT POSITIVE
6009 026660 012737 052476 005474  MOV   #RMBUF,PMA    ;INIT PMA TO RMBUF
6010 026666 005037 005476          CLR   PMA+2        ;INIT PMA+2 TO 0
6011 026672 000405          BR    GTO           ;CONTINUE
6012 026674 104407          GTPRMS: SAVREG     ;SAVE R0-R5
6013 026676 012700 005444          MOV   #SAVPRS,R0    ;ADRS OF SAVE AREA
6014 026702 012701 002632          MOV   #PARMO+2,R1   ;ADRS OF PARAMS
6015 026706 012702 000005          GTO:  MOV   #5,R2    ;SET FOR 5 WORDS
6016 026712 012021          6$:   MOV   (R0)+,(R1)+ ;MOVE A WORD
6017 026714 005302          DEC   R2            ;SEE IF DONE YET
6018 026716 001375          BNE   6$           ;BR IF NOT YET
6019 026720 104410          RESREG ;RESTORE R0-R5
6020 026722 000207          RTS    PC           ;RETURN
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033 026724 010446

```

```

*****
;TRANSFR - PERFORM A DATA TRANSFER, AND HANDLE BAD SECTORS
;OR DATA CHECK ERRORS, IF ENCOUNTERED.
;THE COMMAND MUST BE SET IN THE PARAM BLK. THIS SUB-
;ROUTINE IS SUBSTITUTED FOR DRVCAL WHEN BAD SECTORS MUST BE HANDLED,
;AND IT IS ALSO USED TO HANDLE THE DATA CHECK ERRORS CAUSED
;BY READING DATA WITH OFFSETS, IN THE OVERWRITE AND DATA COMPATIBILITY
;TESTS.
*****

```

```

TRANSFR: MOV   R4,-(SP) ;SAVE R4 ON STACK

```


6034	026726	005004			CLR	R4	:CLEAR ERROR INDICATOR
6035	026730	105037	003134		CLR8	DCEFLG	:CLEAR DCK ERROR FLAG
6036	026734	005037	005414		CLR	RECODE	:CLEAR ERROR FLAGS
6037	026740	012737	020204	003046	MOV	#DCKHDL,A.ABNL	:SET DCK ERROR HANDLER ADRS
6038	026746	105037	003125		CLR8	ERRCNT	:CLEAR ERROR RETRY COUNT
6039	026752	004737	027302		JSR	PC,DRVCAL	:PERFORM THE S.S. FUNCTION
6040	026756	032737	000026	005414	BIT	#BSERR!DCKERR!HVRCER,RECODE	:SEE IF BSE OR DCK OR HVRC ERROR OCCURRED
6041	026764	001410			BEQ	6\$:BR IF NOT
6042	026766	005204			INC	R4	:INCR ERROR INDICATOR
6043	026770	004737	022602		JSR	PC,CKSCOR	:DECR. DRIVE SCORE, IF NECESSARY
6044	026774	004737	026466		JSR	PC,MIDXFR	:UPDATE PARAMS TO RESUME XFER
6045	027000	005765	000012		TST	P.AC(R5)	:SEE IF ENTIRE XFER IS COMPLETED
6046	027004	001353			BNE	4\$:BR IF NOT
6047	027006	005704			TST	R4	:SEE IF ANY ERRORS OCCURRED
6048	027010	001404			BEQ	8\$:BR IF NOT
6049	027012	004737	026674		JSR	PC,GTPRMS	:RESTORE ORIGINAL PARAMS OF XFER
6050	027016	004737	026634		JSR	PC,SVPRMS	:RESTORE WDSXFR,PMA,PMA+2
6051	027022	012604			MOV	(SP)+,R4	:RESTORE R4
6052	027024	012737	030640	003046	MOV	#ERRHDL,A.ABNL	:RESTORE NORMAL ERROR HANDLER ADRS
6053	027032	000207			RTS	PC	:RETURN

6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067

```

:*****
:SBTTL SEARCH BAD SECTOR TABLES ROUTINE
:THIS ROUTINE RETURNS A LIST OF BAD SECTORS FOUND IN
:THE SPECIFIC TABLE (BAD SECTOR SOFTWARE OR BAD SECTOR FACTORY)
:FOR THE SPECIFIC CYLINDER AND TRACK DESIRED. R0 AND R1 MUST
:CONTAIN THE CYLINDER AND TRACK, RESPECTIVELY.
:*
:*THE LIST OF BAD SECTORS IS FOUND ON THE STACK WHEN THE ROUTINE
:*RETURNS TO THE CALLER. THE FIRST ENTRY ON THE STACK WILL BE THE
:*NUMBER OF BAD SECTORS FOUND FOR THIS CYLINDER AND TRACK.
:*****

```

6068	027034	010237	001266		SRHTBS: MOV	R2,STMP2	
6069	027040	010337	001270		MOV	R3,STMP3	
6070	027044	012637	001272		MOV	(SP)+,STMP4	:STORE RETURN CONTENTS OF R4
6071	027050	011402			MOV	(R4),R2	:GET ADDRESS OF BAD SECTOR TABLE TO SEARCH
6072	027052	012437	001264		MOV	(R4)+,STMP1	:SETUP FOR LENGTH OF BAD SECTOR TABLE
6073	027056	062737	001000	001264	ADD	#1000,STMP1	
6074	027064	005003			CLR	R3	:CLEAR R3 FOR COUNT
6075	027066	062702	000010		ADD	#10,R2	:SET R2 FOR POINTER TO CYLINDER ENTRY
6076	027072	005712			TST	(R2)	:TEST IF ALL ONES
6077	027074	100430			BMI	5\$:YES-DONE
6078	027076	020012			CMP	R0,(R2)	:TEST IF BAD SECTOR IN PRESENT CYL
6079	027100	001406			BEQ	3\$:YES-GO CHECK TRACK
6080	027102	062702	000004		ADD	#4,R2	:ELSE BUMP POINTER
6081	027106	020237	001264		CMP	R2,STMP1	:TEST II' OUT OF TABLE
6082	027112	002021			BGE	5\$:YES-EXIT
6083	027114	000766			BR	1\$:LOOP
6084	027116	005722			TST	(R2)+	:TRACK CHECK - PUT POINTER AT TRK/SEC BYTE
6085	027120	011237	001302		MOV	(R2),STMP10	:GET TRK/SEC WORD
6086	027124	042737	174377	001302	BIC	#174377,STMP10	:CLEAR ALL BUT TRACK
6087	027132	123701	001303		CMPB	STMP10+1,R1	:CHECK IF BAD SECTOR IN THIS TRACK
6088	027136	001402			BEQ	4\$:YES - GO PUT SECTOR NUMBER ON STACK
6089	027140	005722			TST	(R2)+	:ELSE BUMP POINTER TO NEXT CYL WORD

```

6090 027142 000753
6091 027144 005203
6092 027146 012246
6093 027150 042716 177700
6094 027154 000746
6095 027156 010346
6096 027160 013702 001266
6097 027164 013703 001270
6098 027170 013746 001272
6099 027174 000204
6100
6101
6102
6103
6104
6105
6106
6107 027176 104407
6108 027200 012737 004152 027212
6109 027206 004437 027034
6110 027212 000000
6111 027214 012603
6112 027216 001015
6113 027220 023727 027212 003152
6114 027226 001404
6115 027230 012737 003152 027212
6116 027236 000763
6117 027240 042737 001000 005414
6118 027246 104410
6119 027250 000207
6120 027252 022602
6121 027254 001403
6122 027256 005303
6123 027260 001374
6124 027262 000756
6125 027264 052737 001000 005414
6126 027272 005303
6127 027274 001764
6128 027276 005726
6129 027300 000774
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145

```

```

4$: BR 1$ ;GO TEST NEXT CYL
INC R3 ;BUMP BAD SECTOR COUNT
MOV (R2)+, -(SP) ;PUT TRK/SEC WORD ON STACK
BIC #177700, (SP) ;CLEAR ALL BUT SECTOR NUMBER
BR 1$ ;GO CHECK REST OF FILE
5$: MOV R3, -(SP) ;EXIT - PUT NUMBER OF BAD
MOV STMP2, R2 ;SECTORS ON STACK-RESTORE
MOV STMP3, R3 ;REGISTERS
MOV STMP4, -(SP) ;PUT RETURN ON STACK
RTS R4 ;RETURN
;*****
;SBTTL BAD SECTOR CHECK ROUTINE
;THIS ROUTINE WILL SEARCH BOTH TABLES TO DETERMINE IF A
;SPECIFIC SECTOR IS LISTED AS BAD. IF THE SECTOR IS LISTED IN THE
;TABLES THE ROUTINE SETS THE "BADSEC" FLAG AND RETURNS. IF THE SECTOR
;IS NOT LISTED THE FLAG IS RESET.
;*****
BDSRCK: SAVREG
MOV #BSFACT, 1$ ;SET TABLE TO SEARCH
JSR R4, SRHTBS ;GO SEARCH IT
1$: .WORD ;TABLE ADDRESS GOES HERE
MOV (SP)+, R3 ;GET NUMBER OF BAD SECTORS, IF ANY
BNE 6$ ;IF ANY, GO TEST WHICH ONES
7$: CMP 1$, #BSSOFT ;ELSE TEST IF OTHER TABLE ALREADY SEARCHED
BEQ 3$ ;IF YES-EXIT
MOV #BSSOFT, 1$ ;SET OTHER TABLE FOR SEARCH
BR 2$ ;GO SEARCH IT
3$: BIC #BADSEC, RECODE ;CLEAR BAD SECTOR BIT
4$: RESREG
RTS PC ;RETURN
6$: CMP (SP)+, R2 ;THIS SECTOR IN TABLE?
BEQ 8$ ;YES-GO SET BIT & EXIT
DEC R3 ;DECREMENT BAD SECTOR COUNT
BNE 6$ ;IF NOT ZERO-CHECK NEXT ENTRY
BR 7$ ;ELSE GO SEARCH OTHER TABLE
8$: BIS #BADSEC, RECODE ;SET BAD SECTOR BIT
9$: DEC R3 ;CLEAR STACK OF OTHER BAD SECTOR
BEQ 4$ ;NUMBER
BR 9$ ;EXIT
;*****
;SBTTL CALL DRIVER ROUTINE
;ENTRY JSR PC, DRVCAL
; WITH R5 POINTING TO PARAMETER BLOCK
;RETURN RTS PC
;
;THIS ROUTINE IS USED TO INITIATE A SUBSYSTEM OPERATION BY
;CALLING THE DRIVER. THE PARAMETER BLOCK MUST BE SET UP
;BY THE CALLER AND R5 MUST POINT TO THE PARAMETER
;BLOCK WHEN THE ROUTINE IS CALLED.
;
;THIS ROUTINE WAITS FOR THE OPERATION TO BE COMPLETED.

```

CALL DRIVER ROUTINE

```

6146
6147
6148
6149
6150
6151 027302
6152 027302 004737 023556
6153 027306 004737 024722
6154 027312 105037 003122
6155 027316 105037 003131
6156 027322 004737 027354
6157 027326 017537 027336
6158 027332 004737 037356
6159 027336 000000
6160 027340 004737 033726
6161 027344 105737 003122
6162 027350 001773
6163 027352 000207
6164
6165
6166
6167
6168
6169
6170 027354 104407
6171 027356 012701 005152
6172 027362 012700 005166
6173
6174 027366 012021
6175 027370 012021
6176 027372 012021
6177 027374 012021
6178 027376 012021
6179 027400 012021
6180
6181 027402 012701 005166
6182 027406 012521
6183 027410 012521
6184 027412 012521
6185 027414 012521
6186 027416 012521
6187 027420 012521
6188 027422 104410
6189 027424 000207
6190
6191
6192
6193
6194
6195
6196
6197
6198
6199
6200 027426 152737 000377 003122
6201 027434 032737 100000 005414

```

```

; *WHILE WAITING THE WATCHDOG TIMER IS CALLED TO PREVENT
; *SILENT DEATH IN CASE THE SUBSYSTEM DOES NOT PROVIDE AN
; *INTERRUPT. THE TERMINATION HANDLER ROUTINES WILL SET THE DONE
; *FLAG WHICH KEYS THE ROUTINE TO RETURN TO THE CALLER.
; *****
DRVCAL:
      JSR      PC,STALL      ;PERFORM A STALL IF REQUIRED
      JSR      PC,CTLOUT    ;CHECK FOR (1C) OR (1Z) KBD INPUT
      CLRB     DONE        ;CLEAR DONE FLAG
      CLRB     DRNAFG       ;CLEAR DRIVE SEIZED BY OTHER PORT FLAG
      JSR      PC,STRCMD    ;STORE PREV AND CURRENT COMMANDS
      MOV      R5,4S        ;GET PARAM BLOCK ADDRESS
      JSR      PC,C.INIT    ;CALL DRIVER
      .WORD    P.B. ADDRESS GOES HERE
4S:   JSR      PC,W.WTCH     ;CALL WATCH DOG
6S:   TSTB     DONE        ;DONE SET?
      BEQ      6S          ;NO-LOOP
      RTS      PC          ;YES-RETURN

```

```

; *****
; *STRCMD - STORE PREVIOUS AND CURRENT SUBSYSTEM COMMANDS
; *****
STRCMD: SAVREG      ;SAVE R0-R5
      MOV      #PRVCMD,R1  ;ADDR OF PREV COMMAND STORAGE
      MOV      #COMSTR,R0  ;ADDR OF CURRENT COMMAND STORAGE
; STORE PREVIOUS COMMAND
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
; STORE CURRENT COMMAND
4S:   MOV      #COMSTR,R1
      MOV      (R5)+,(R1)+ ;R5 POINTS TO DRIVER PARAM BLK
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+
      RESREG
      RTS      PC          ;RESTORE R0-R5
;RETURN

```

```

; *****
; *SBTTL DRIVE ERROR FREE RETURN ROUTINE
; *THIS ROUTINE IS CALLED BY THE DRIVER WHEN NO ERROR
; *HAS BEEN DETECTED IN THE OPERATION. THE ROUTINE SETS THE
; *DONE FLAG THAT IS TESTED IN THE DRIVER CALLING
; *ROUTINE.
; *****
ERRFRE: BISB     #377,DONE  ;SET THE DONE FLAG
      BIT      #ANYDER,RECODE ;TEST IF ANY DATA ERROR

```

```

6202 027442 001403          BEQ      2$          ; IF NO - DO ERROR RECOVERY PRINT TEST
6203 027444 105037 003125    CLRB    ERRCNT     ; CLEAR ERROR COUNT
6204 027450 000413          BR      1$          ; EXIT
6205 027452 105737 003125    2$:    TSTB    ERRCNT ; CHECK IF ANY ERRORS HAVE OCCURRED
6206 027456 001410          BEQ      1$          ; NO - SKIP TO EXIT
6207 027460 005037 001174    CLR     $REGS      ; GET RETRY COUNT
6208 027464 113737 003125 001174    MOVB   ERRCNT,$REGS ; PRINT RETRY SUCCESSFUL MESSAGE
6209 027472 104101          ERROR   101        ; CLEAR ERROR COUNT
6210 027474 105037 003125    CLRB    ERRCNT     ; CLEAR RECOVERY FLAGS
6211 027500 005037 005414    1$:    CLR     RECODE ; RETURN
6212 027504 000207          RTS     PC
6213                                     ;*****
6214                                     ;SBTTL TYPE ERROR ROUTINE
6215                                     ;*ENTRY - JSR PC,TYPERR
6216                                     ;*RETURN - RTS PC
6217                                     ;*
6218                                     ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
6219                                     ;*ERROR IS TO BE REPORTED. IT THEN USES THE "ERROR TABLE" ($ERRTB)
6220                                     ;*ENTRY TO DEFINE WHAT INFORMATION IS TO BE REPORTED CONCERNING
6221                                     ;*THE ERROR.
6222                                     ;*****
6223 027506 104407          TYPERR: SAVREG
6224 027510 105237 003127    INCB   DRVERS     ; INCR ERROR COUNT FOR THIS DRIVE
6225 027514 032777 020000 151416 9$:    BIT    $SW13,$SWR ; INHIBIT ERROR TIMEOUTS?
6226 027522 001402          BEQ     6$          ; BR IF NO
6227 027524 000137 030130    JMP    20$
6228 027530 005000          6$:    CLR    R0        ; CLR R0 FOR ERROR NUMBER
6229 027532 005005          CLR    R5        ; INIT INDENT INDICATOR
6230 027534 005105          COM    R5
6231 027536 113700 001114    MOVB   $ITEMB,R0 ; ENTER ERROR NUMBER
6232 027542 005300          DEC    R0        ; FORM INDEX FOR ERROR TABLE
6233 027544 006300          ASL   R0
6234 027546 006300          ASL   R0
6235 027550 006300          ASL   R0
6236 027552 062700 001410 1$:    ADD    $ERRTB,R0 ; FORM ADDRESS OF ERROR ENTRY
6237 027556 012037 027576    MOV    (R0)+,2$   ; GET EM POINTER
6238 027562 001406          BEQ    3$          ; BRANCH IF THERE ISN'T ONE
6239 027564 104401 011703    TYPE   ,CR2LF
6240 027570 104401 044546    TYPE   ,AS2SP2
6241 027574 104401          TYPE   ; TYPE "## "
6242 027576 000000          TYPE   ; TYPE ERROR MESSAGE (EM)
6243 027600 012037 027754    2$:    .WORD 0         ; EM POINTER GOES HERE
6244 027604 001467          3$:    MOV    (R0)+,4$   ; GET DH POINTER
6245 027606 104401 001325    BEQ    5$          ; BR IF THERE ISN'T ONE
6246 027612 104401 044534    TYPE   ,$CRLF
6247 027616 013746 001102    TYPE   ,TSTMSG
6248 027622 104403          MOV    $TSTNM,-(SP) ; TYPE " TEST "
6249 027624 002          TYPOS ; GET TEST NO. ON STACK
6250 027625 000          .BYTE 2         ; TYPE IT
6251 027626 104401 011703    .BYTE 0         ; 2 DIGITS
6252 027632 032777 010000 151300 TYPE   ,CR2LF
6253 027640 001133          BIT    $BIT12,$SWR ; SUPPRESS LEADING ZEROS
6254 027642 104401 047670    BNE   20$        ; REPORT DESCRIPTION ONLY ?
6255 027646 104401 001325    TYPE   ,DH10S    ; BR IF YES
6256 027652 104401 050062    TYPE   ,DH101+10 ; TYPE "PREVIOUS COMMAND : "
6257 027656 104401 001325    TYPE   ,CRLF    ; TYPE PREV COMMAND HEADER

```

TYPE ERROR ROUTINE

6258	027662	012701	000006	MOV	#6,R1	;SIX COMMAND VALUES
6259	027666	012702	001236	MOV	#\$REG26,R2	;STARTING ADDR OF PREV CMND VALUES
6260	027672	012246		30\$: MOV	(R2)+,-(SP)	;PUT A WORD ON STACK
6261	027674	104402		TYPOC		;TYPE IT
6262	027676	104401	011717	TYPE	SPACE2	;TYPE SEPARATORS
6263	027702	005301		DEC	R1	;SEE IF 7 VALUES TYPED YET
6264	027704	001372		BNE	30\$;BR IF NOT
6265	027706	104401	001325	TYPE	,SCLF	
6266	027712	104401	011717	TYPE	,SPACE2	;INDENT
6267	027716	104401	050141	TYPE	,DH102	;TYPE HDR FOR BA DATA
6268	027722	104401	001325	TYPE	,SCLF	
6269	027726	104401	011717	TYPE	,SPACE2	;INDENT
6270	027732	012246		MOV	(R2)+,-(SP)	
6271	027734	104402		TYPOC		;TYPE PREV. HI BA BITS
6272	027736	104401	011717	TYPE	SPACE2	
6273	027742	011246		MOV	(R2),-(SP)	
6274	027744	104402		TYPOC		;TYPE PREV. LO BA BITS
6275	027746	104401	011703	TYPE	,CR2LF	
6276	027752	104401		TYPE		;TYPE DATA HEADER
6277	027754	000000		4\$: WORD	0	;DH POINTER GOES HERE
6278	027756	104401	001325	TYPE	,SCLF	
6279	027762	005005		CLR	R5	;INIT INDENT INDICATOR
6280	027764	032777	010000 151146	5\$: BIT	#BIT12,DSWR	;REPORT DESCRIPTION ONLY ?
6281	027772	001056		BNE	20\$;BR IF YES
6282	027774	012001		MOV	(R0)+,R1	;GET DT POINTER
6283	027776	001454		BEQ	20\$;BRANCH IF THE E ARE NONE
6284	030000	012000		MOV	(R0)+,R0	;GET DF POINTER
6285	030002	012002		MOV	(R0)+,R2	;STORE NUMBER OF DH'S
6286	030004	112003		10\$: MOV	(R0)+,R3	;GET & STORE NUMBER OF DATA WORDS
6287	030006	105720		TSTB	(R0)+	;BUMP PAST FORMAT WORD
6288	030010	005703		TST	R3	;TEST IF ANY DATA FOR THIS HEADER
6289	030012	001417		BEQ	14\$;NO - SKIP DATA PRINT
6290	030014	005705		TST	R5	;SEE IF SHOULD INDENT
6291	030016	001002		BNE	11\$;BR IF NOT
6292	030020	104401	011717	TYPE	SPACE2	;INDENT
6293	030024	013146		11\$: MOV	(R1)+,-(SP)	;PUT FIRST DATA WORD ON STACK
6294	030026	104402		TYPOC		;TYPE IT
6295	030030	005303		DEC	R3	;MORE DATA WORDS
6296	030032	001403		BEQ	12\$;NO-BRANCH
6297	030034	104401	011717	TYPE	SPACE2	;TYPE SEPARATORS
6298	030040	000771		BR	11\$;LOOP
6299	030042	005702		12\$: TST	R2	;SEE IF <CR>,<LF> NEEDED
6300	030044	001402		BEQ	14\$;BR IF NOT
6301	030046	104401	001325	TYPE	,SCLF	;TYPE IT
6302	030052	005302		14\$: DEC	R2	;MORE DH'S?
6303	030054	003425		BLE	20\$;NO-BRANCH
6304	030056	012037	030110	15\$: MOV	(R0)+,16\$;GET NEXT DH POINTER
6305	030062	105710		TSTB	(R0)	;SEE IF ANY DATA FOR HDR
6306	030064	001004		BNE	34\$;BR IF YES
6307	030066	104401	001325	TYPE	,SCLF	;SKIP EXTRA LINE
6308	030072	005005		CLR	R5	;RE-INIT INDENT INDICATOR
6309	030074	000404		BR	36\$	
6310	030076	005105		34\$: COM	R5	;COMPLEMENT INDENT INDICATOR
6311	030100	001002		BNE	36\$;BR IF NO INDENT REQUIRED
6312	030102	104401	011717	TYPE	,SPACE2	;INDENT
6313	030106	104401		36\$: TYPE		;TYPE DH

TYPE ERROR ROUTINE

```

6314 030110 000000          16$:  WORD 0          ;DH POINTER GOES HERE
6315 030112 104401 001325  TYPE  $CRLF
6316 030116 105710          TSTB (R0)          ;TYPE A DT?
6317 030120 001331          BNE  10$          ;YES-BRANCH
6318 030122 062700 000002  ADD  #2,R0        ;INCREMENT OF POINTER
6319 030126 000751          BR   14$          ;SEE IF END OF DF BLOCK
6320 030130 104410
6321 030132 000207
6322
6323 ;*****
6324 ;SBTTL CONTROLLER ERROR REPORTER ROUTINE
6325 ;*ENTRY: JSR PC, CONERR
6326 ;*RETURN: RTS PC
6327 ;*
6328 ;*THIS ROUTINE DECODES THE CONTROLLER ERROR WORD AND
6329 ;*REPORTS THE APPROPRIATE MESSAGE. THE RK611 REGISTERS ARE
6330 ;*RETRIEVED FROM THE RK611 AND PLACED IN THE PARAMETER
6331 ;*BLOCK. THIS IS DONE BECAUSE PARM 0 MAY NOT BE VALID
6332 ;*AT THIS TIME.
6333 ;*****
6333 030134 104407          CONERR: SAVREG      ;SAVE R0-R5
6334 030136 152737 000377 003122  BISB #377,DONE     ;SET DONE FLAG
6335 030144 105237 003125          INCB ERRCNT        ;INCREMENT ERROR COUNT
6336 030150 004737 033120          JSR  PC,TOPROC     ;LOAD RK REGS INTO $REGS
6337 030154 032737 000001 003052  BIT  #BIT0,E.CONT  ;ERROR 0?
6338 030162 001402          BEQ  1$           ;NO-BRANCH
6339 030164 104064          ERROR 64         ;CLEAR CONT DID NOT CLEAR ERROR
6340 030166 000470          BR   7$
6341 030170 032737 000002 003052  1$: BIT  #BIT1,E.CONT  ;ERROR 1?
6342 030176 001402          BEQ  2$           ;NO-BRANCH
6343 030200 104065          ERROR 65         ;NO ATTENTION IN ATTENTION SUM REG
6344 030202 000462          BR   7$
6345 030204 032737 000004 003052  2$: BIT  #BIT2,E.CONT  ;ERROR 2?
6346 030212 001407          BEQ  3$           ;NO-BRANCH
6347 030214 105737 003131          TSTB DRNAFG       ;SEE IF DRIVE WAS SIEZED BY OTHER PORT
6348 030220 001402          BEQ  15$          ;BR IF NOT
6349 030222 105237 003132          INCB REISSU        ;SET FLAG TO RE-ISSUE COMMAND
6350 030226 104066          15$: ERROR 66      ;UNSOLICITED ATTENTION
6351 030230 000447          BR   7$
6352 030232 032737 000010 003052  3$: BIT  #BIT3,E.CONT  ;ERROR 3?
6353 030240 001402          BEQ  4$           ;NO-BRANCH
6354 030242 104067          ERROR 67         ;UNEXPECTED DATA TYPE ERROR
6355 030244 000441          BR   7$
6356 030246 032737 000020 003052  4$: BIT  #BIT4,E.CONT  ;ERROR 4?
6357 030254 001402          BEQ  5$           ;NO-BRANCH
6358 030256 104070          ERROR 70         ;ATTENTION DID NOT RESET WITH CLEAR
6359 030260 000433          BR   7$
6360 030262 032737 000040 003052  5$: BIT  #BIT5,E.CONT  ;ERROR 5?
6361 030270 001402          BEQ  6$           ;NO-BRANCH
6362 030272 104071          ERROR 71         ;SUBSYS CLEAR DIDN'T CLEAR DRIVE ATTENTION
6363 030274 000425          BR   7$
6364 030276 032737 000400 003052  6$: BIT  #BIT8,E.CONT
6365 030304 001401          BEQ  8$
6366 030306 104072          ERROR 72         ;DATA LATE WHEN UNLOADING HEADER
6367 030310 032737 001000 003052  8$: BIT  #BIT9,E.CONT
6368 030316 001401          BEQ  9$
6369 030320 104073          ERROR 73         ;CONTROLLER ERROR DURING DRIVER SERVICE

```

CONTROLLER ERROR REPORTER ROUTINE

```

6370 030322 032737 002000 003052 9$: BIT #BIT10,E.CONT
6371 030330 001401 BEQ 10$
6372 030332 104074 ERROR 74 ;DRIVE DETECTED PARITY ERROR
6373 030334 032737 100000 003052 10$: BIT #BIT15,E.CONT
6374 030342 001401 BEQ 11$
6375 030344 104052 ERROR 52 ;MULTIPLE DRIVE SELECT
6376 030346 104075 11$: ERROR 75 ;UNDEFINED ERROR
6377 030350 005037 003052 7$: CLR E.CONT ;CLEAR CONTROLLER ERROR WORD
6378 030354 000137 032714 JMP BGNRTY ;GO DO RETRY
6379 ;*****
6380 ;S8TTL REPORT SUPPORT ROUTINE
6381 ;*THIS ROUTINE LOADS ALL THE PARAMETER BLOCK DATA TO BE REPORTED
6382 ;*INTO THE PROPER TEMPORARY REGISTERS FOR REPORTING. ALL THESE MAY
6383 ;*NOT BE INCLUDED IN THE REPORT, BUT THEY ARE LOADED ANYWAY.
6384 REPSUP:
6385 030360 104407 SAVREG
6386 030362 005037 005416 CLR ERRCOM ;CLEAR ERROR COMMAND STORE
6387 030366 116537 000001 005416 MOVB P.CMND(R5),ERRCOM ;STORE COMMAND START VALUES
6388 030374 012700 001162 MOV #SREG0,R0 ;FOR REPORTING
6389 030400 016520 000002 MOV P.CYLN(R5),(R0)+
6390 030404 116520 000005 MOV P.TRCK(R5),(R0)+
6391 030410 105020 CLR R3 (R0)+
6392 030412 116520 000004 MOV P.SECT(R5),(R0)+
6393 030416 105020 CLRB (R0)+
6394 030420 016520 000012 MOV P.WC(R5),(R0)+
6395 030424 012700 001174 MOV #SREG5,R0
6396 030430 116503 000007 MOVB P.BAHI(R5),R3
6397 030434 042703 177774 BIC #177774,R3
6398 030440 010337 001256 MOV R3,SREG36 ;HI BA BITS
6399 030444 016537 000010 001260 MOV P.BALO(R5),SREG37 ;LO BA BITS
6400 030452 016520 000016 MOV P.CS1(R5),(R0)+ ;GET ALL THE VALUES FROM THE
6401 030456 016520 000020 MOV P.CS2(R5),(R0)+ ;PARAMETER BLOCK AND LOAD
6402 030462 016520 000030 MOV P.DCYL(R5),(R0)+ ;THE TEMPORARY REGISTERS
6403 030466 016520 000026 MOV P.DTS(R5),(R0)+ ;FOR REPORTING. ALL THIS
6404 030472 016520 000022 MOV P.WCR(R5),(R0)+ ;DATA MAY NOT BE VALID
6405 ;FOR ALL REPORTS (TO BE
6406 030476 016520 000024 MOV P.BAR(R5),(R0)+ ;DETERMINED LATER) BUT IT IS
6407 030502 016520 000032 MOV P.ASOF(R5),(R0)+ ;STORED ANY WAY.
6408 030506 016520 000036 MOV P.DS(R5),(R0)+
6409 030512 016520 000034 MOV P.ER(R5),(R0)+
6410 030516 016520 000040 MOV P.A00(R5),(R0)+
6411 030522 016520 000042 MOV P.B00(R5),(R0)+
6412 030526 016520 000044 MOV P.A01(R5),(R0)+
6413 030532 016520 000046 MOV P.B01(R5),(R0)+
6414 030536 016520 000050 MOV P.A10(R5),(R0)+
6415 030542 016520 000052 MOV P.B10(R5),(R0)+
6416 030546 016520 000054 MOV P.A11(R5),(R0)+
6417 030552 016520 000056 MOV P.B11(R5),(R0)+
6418 ;STORE PREVIOUS COMMAND FOR PRINTOUT
6419 030556 012701 001236 MOV #SREG26,R1 ;ADRS OF PRINT BUF AREA
6420 030562 012700 005152 MOV #PRVCMO,R0 ;ADRS OF PREV CMND STORAGE
6421 030566 112021 MOVB (R0)+,(R1)+ ;DRIVE NO.
6422 030570 105021 CLRB (R1)+
6423 030572 112021 MOVB (R0)+,(R1)+ ;COMMAND
6424 030574 105021 CLRB (R1)+
6425 030576 012021 MOV (R0)+,(R1)+ ;CYL ADDRESS

```

6426	030600	116021	000001
6427	030604	105021	
6428	030606	111021	
6429	030610	105021	
6430	030612	016021	000006
6431	030616	116003	000003
6432	030622	042703	177774
6433	030626	010321	
6434	030630	016011	000004
6435	030634	104410	
6436	030636	000207	

```

MOV 1(R0),(R1)+ ;TRACK
CLRB (R1)+
MOV (R0),(R1)+ ;SECTOR
CLRB (R1)+
MOV 6(R0),(R1)+ ;WORD COUNT
MOV 3(R0),R3 ;HI BA BITS
BIC #177774,R3
MOV R3,(R1)+
MOV 4(R0),(R1) ;LO BA BITS
RESREG
RTS PC

```

```

*****
:SBTTL REPORT ERROR ROUTINE
:* ENTRY JSR PC,ERRHDL
:*RETURN RTS PC
:*
:*THIS ROUTINE IS CALLED BY THE DRIVER WHEN AN ERROR IS DETECTED
:*IN THE OPERATION. THE ROUTINE DETERMINES WHICH COMMAND WAS
:*BEING EXECUTED AND GENERATES THE PROPER ERROR MESSAGE.
:*****

```

6447	030640	104407		
6448	030642	152737	000377	003122
6449	030650	105237	003125	
6450	030654	005037	005414	
6451	030660	032737	000400	005414
6452	030666	001402		
6453	030670	012705	002714	
6454	030674	012737	033072	003046
6455	030702	012737	033110	003044
6456	030710	004737	030360	

```

ERRHDL: SAVREG
BISB #377,DONE
INCB ERRCNT
CLR RECODE
ER2ENT: BIT #LEVZER,RECODE
BEQ 52$
MOV #PARM1,R5
52$: MOV #RETANL,A.ABNL
MOV #RETANL,A.NORM
JSR PC,REPSUP

```

```

;SET DONE FLAG
;INCREMENT ERROR COUNT
;CLEAR RECOVERY CODE WORD
;TEST IF 2ND LEVEL ERROR
;NO - SKIP PARAM BLOCK CHANGE
;ELSE SET R5 TO PARAMETER BLOCK 1
;SET NOW A NORMAL AND NORMAL RETURN FOR
;DRIVER OPERATIONS IN ERROR PROCESSING
;GO SET UP REGISTERS FOR REPORT
;NOW BEGIN TESTING THE ERROR
;BITS. THE SEQUENCE IN
;WHICH THEY ARE TESTED IS
;CONSIDERED SIGNIFICANT IN
;THAT ERRORS OF A MORE
;CATASTROPHIC NATURE ARE FIRST
;TESTED.

```

6474	030714	016504	000034	
6475	030720	032765	020000	000020
6476	030726	001406		
6477	030730	104001		
6478	030732	052737	000200	005414
6479	030740	000137	032372	
6480	030744	032765	004000	000020
6481	030752	001406		

```

MOV P.ER(R5),R4
BIT #UPE,P.CS2(R5)
BEQ 1$
ERROR 1
BIS #ABORT,RECODE
JMP 37$
1$: BIT #NEM,P.CS2(R5) ;TEST NON-EXISTANT MEMORY
BEQ 2$

```

```

;IF AN ERROR IS FOUND SET
;THAT ERROR IS REPORTED AND
;THE REPORTING IS TERMINATED.
;IF ADDITIONAL ERRORS ARE SET,
;THE RK611 REGISTER PRINTOUTS
;WILL SHOW THIS BUT THE
;REGISTER CONTENTS MUST BE
;MANUALLY DECODED TO LOCATE THE
;SECOND ERROR
;SET R4 TO ERROR REGISTER
;TEST UPE. IF UES-SET
;REPORT ERROR

```


6482	030754	104002			ERROR	2	
6483	030756	052737	000200	005414	BIS	#ABORT, RECODE	
6484	030764	000137	032372		JMP	37\$	
6485	030770	032765	010000	000020	2\$: BIT	#NED, P. CS2(R5)	; TEST NON-EXISTANT DRIVE
6486	030776	001412			BEQ	3\$	
6487	031000	032765	000400	000020	BIT	#UFE, P. CS2(R5)	; TEST IF NED & UFE BOTH SET
6488	031006	001403			BEQ	38\$	
6489	031010	104035			ERROR	35	; NED & UFE BOTH SET ERROR
6490	031012	000137	032372		JMP	37\$	
6491	031016	104003			38\$: ERROR	3	; NED ONLY
6492	031020	000137	032372		JMP	37\$	
6493	031024	032765	000400	000020	3\$: BIT	#UFE, P. CS2(R5)	; TEST UNIT FIELD ERROR
6494	031032	001412			BEQ	4\$	
6495	031034	032765	010000	000020	BIT	#NED, P. CS2(R5)	; TEST IF UFE & NED BOTH SET
6496	031042	001403			BEQ	39\$; NO-SKIP
6497	031044	104035			ERROR	35	; REPORT NED & UFE BOTH SET
6498	031046	000137	032372		JMP	37\$	
6499	031052	104004			39\$: ERROR	4	; REPORT UFE ONLY
6500	031054	000137	032372		JMP	37\$	
6501	031060	032765	000100	000014	4\$: BIT	#CMOTO, P. PRST(R5)	
6502	031066	001423			BEQ	5\$	
6503	031070	004737	033120		JSR	PC, TOPROC	; GO PROCESS TIMEOUT
6504	031074	032737	002000	005414	BIT	#TWOTOS, RECODE	; 2ND TIMEOUT IN TIMEOUT PROC?
6505	031102	001007			BNE	40\$; YES - SKIP TO ERROR 56
6506	031104	032737	000400	005414	BIT	#LEVZER, RECODE	; TEST IF LEVEL 2 ERROR
6507	031112	001006			BNE	41\$; YES - SKIP TO ERROR 57
6508	031114	104005			ERROR	5	; ELSE MAKE FULL TIMEOUT REPORT
6509	031116	000137	032372		JMP	37\$	
6510	031122	104056			40\$: ERROR	56	; TWO TIMEOUTS ERROR REPORT
6511	031124	000137	032372		JMP	37\$	
6512	031130	104057			41\$: ERROR	57	; 2ND LEVEL ERROR REPORT
6513	031132	000137	030660		JMP	ER2ENT	; GO BUILD AND MAKE 2ND REPORT
6514	031136	032765	010000	000014	5\$: BIT	#DRVSZD, P. PRST(R5)	; SEE IF DRIVE SIEZED BY OTHER PORT
6515	031144	001403			BEQ	65\$; BR IF DRIVE IS AVAILABLE
6516	031146	104107			ERROR	107	; DRIVE SIEZED BY OTHER PORT
6517	031150	000137	032372		JMP	37\$	
6518	031154	032765	020000	000016	65\$: BIT	#SPAR, P. CS1(R5)	; TEST D TO C PARITY ERROR
6519	031162	001406			BEQ	6\$	
6520	031164	104006			ERROR	6	
6521	031166	052737	004000	005414	BIS	#RCLREQ, RECODE	
6522	031174	000137	032372		JMP	37\$	
6523	031200	032704	000010		6\$: BIT	#DRPAR, R4	; TEST DRIVE DETECTED PARITY ERROR
6524	031204	001406			BEQ	7\$	
6525	031206	104007			ERROR	7	
6526	031210	052737	004000	005414	BIS	#RCLREQ, RECODE	
6527	031216	000137	032372		JMP	37\$	
6528	031222	032765	000010	000036	7\$: BIT	#ACLO, P. DS(R5)	; TEST AC LOW
6529	031230	001403			BEQ	8\$	
6530	031232	104010			ERROR	10	
6531	031234	000137	032372		JMP	37\$	
6532	031240	032765	000020	000036	8\$: BIT	#SPDLSS, P. DS(R5)	; TEST SPEED LOSS
6533	031246	001403			BEQ	24\$	
6534	031250	104011			ERROR	11	
6535	031252	000137	032372		JMP	37\$	
6536	031256	032765	004000	000016	24\$: BIT	#CTO, P. CS1(R5)	; TEST FOR CONTROLLER TIMEOUT
6537	031264	001401			BEQ	25\$	

6538	031266	104027				ERROR	27		
6539	031270	032704	000001		25\$:	BIT	#ILC,R4		;TEST ILLEGAL FUNCTION CODE
6540	031274	001403				BEQ	10\$		
6541	031276	104012				ERROR	12		
6542	031300	000137	032372			JMP	37\$		
6543	031304	032765	002000	000020	10\$:	BIT	#PGE,P.CS2(R5)		;TEST PROGRAMMING ERROR
6544	031312	001403				BEQ	11\$		
6545	031314	104013				ERROR	13		
6546	031316	000137	032372			JMP	37\$		
6547	031322	032704	000004		11\$:	BIT	#ILF,R4		;TEST ILLEGAL DRIVE FUNCTION
6548	031326	001403				BEQ	12\$		
6549	031330	104014				ERROR	14		
6550	031332	000137	032372			JMP	37\$		
6551	031336	032704	000040		12\$:	BIT	#DIYE,R4		;TEST DRIVE TYPE ERROR
6552	031342	001403				BEQ	13\$		
6553	031344	104015				ERROR	15		
6554	031346	000137	032372			JMP	37\$		
6555	031352	032704	000020		13\$:	BIT	#FMTE,R4		;TEST FORMAT ERROR
6556	031356	001403				BEQ	14\$		
6557	031360	104016				ERROR	16		
6558	031362	000137	032372			JMP	37\$		
6559	031366	032704	004000		14\$:	BIT	#WLE,R4		;TEST WRITE LOCK ERROR
6560	031372	001403				BEQ	15\$		
6561	031374	104017				ERROR	17		
6562	031376	000137	032372			JMP	37\$		
6563	031402	032704	040000		15\$:	BIT	#UNS,R4		;TEST DRIVE UNSAFE
6564	031406	001406				BEQ	16\$		
6565	031410	104020				ERROR	20		
6566	031412	052737	000200	005414		BIS	#ABORT,RECODE		
6567	031420	000137	032472			JMP	ALLTRM		
6568	031424	032704	000002		16\$:	BIT	#SKI,R4		;TEST SEEK INCOMPLETE
6569	031430	001406				BEQ	17\$		
6570	031432	104021				ERROR	21		
6571	031434	052737	004000	005414		BIS	#ACLREQ,RECODE		
6572	031442	000137	032372			JMP	37\$		
6573	031446	032704	001000		17\$:	BIT	#COE,R4		;TEST CYLINDER OVERFLOW
6574	031452	001406				BEQ	18\$		
6575	031454	104022				ERROR	22		
6576	031456	052737	004000	005414		BIS	#ACLREQ,RECODE		
6577	031464	000137	032372			JMP	37\$		
6578	031470	032704	002000		18\$:	BIT	#IDAE,R4		;TEST ILLEGAL CYLINDER
6579	031474	001406				BEQ	19\$		
6580	031476	104023				ERROR	23		
6581	031500	052737	004000	005414		BIS	#ACLREQ,RECODE		
6582	031506	000137	032372			JMP	37\$		
6583	031512	032765	000040	000036	19\$:	BIT	#DROT,P.DS(R5)		;TEST DRIVE OFF TRACK
6584	031520	001406				BEQ	20\$		
6585	031522	104024				ERROR	24		
6586	031524	052737	004000	005414		BIS	#ACLREQ,RECODE		
6587	031532	000137	032372			JMP	37\$		
6588	031536	032704	010000		20\$:	BIT	#DTE,R4		;TEST DRIVE TIMING ERROR
6589	031542	001406				BEQ	21\$		
6590	031544	104025				ERROR	25		
6591	031546	052737	000200	005414		BIS	#ABORT,RECODE		
6592	031554	000137	032372			JMP	37\$		
6593	031560	032765	100000	000020	21\$:	BIT	#DLT,P.CS2(R5)		;TEST DATA LATE

```

6594 031566 001403      BEQ      22$
6595 031570 104026      ERROR   26
6596 031572 000137 032372      JMP      37$
6597 031576 032704 020000      22$: BIT      #OPI,R4      ;TEST IF OPI ERROR
6598 031602 001470      BEQ      29$
6599 031604 052737 000010 005414      BIS      #OPIERR,RECODE
6600 031612 105737 003120      TSTB    DERCNT      ;TEST IF FIRST ERROR
6601 031616 001402      BEQ      50$
6602 031620 000137 032372      JMP      37$      ;NO - SKIP REPORT
6603 031624 032737 000400 005414 50$: BIT      #LEV2ER,RECODE ;TEST IF A SECOND LEVEL 2 ERROR
6604 031632 001403      BEQ      26$      ;HAS ALREADY OCCURRED
6605 031634 104054      27$: ERROR   54
6606 031636 000137 032372      JMP      37$      ;GET OUT OF ERROR REPORT
6607 031642 004737 033600      26$: JSR     PC,BLDEXH ;GO BUILD EXPECTED HEADER
6608 031646 004737 033270      JSR     PC,ROH00    ;GET HEADER OF SECTOR 0
6609 031652 032737 000400 005414      BIT      #LEV2ER,RECODE ;TEST IF ERROR GETTING HDR
6610 031660 001036      BNE      28$      ;IF YES-GO MAKE ABBREVIATED REPORT
6611 031662 013702 003036      MOV     RKBAS,R2   ;STORE HEADER 0 INTO REGISTERS
6612 031666 042762 000100 000000      BIC     #IE,RKCS1(R2) ;RESET INTERRUPT ENABLE
6613 031674 016237 000024 001202      MOV     RKDB(R2),R10 ;FOR REPORTING
6614 031702 016237 000024 001204      MOV     RKDB(R2),R11
6615 031710 016237 000024 001206      MOV     RKDB(R2),R12
6616 031716 032762 100000 000000      BIT      #CERR,RKCS1(R2) ;TEST IF ERROR DURING STORAGE
6617 031724 001343      BNE      27$
6618 031726 105737 003133      TSTB    TSTYP      ;SEE IF READING WITH OFFSET
6619 031732 001403      BEQ      54$      ;BR IF NOT
6620 031734 105037 003125      CLRB    ERRCNT     ;CLEAR ERROR COUNT
6621 031740 000401      BR      55$      ;CONTINUE
6622 031742 104030      54$: ERROR   30      ;MAKE OPI REPORT
6623 031744 052737 004000 005414 55$: BIS      #ACLREQ,RECODE ;SET RECALIBRATE REQUEST BIT
6624 031752 000137 032372      JMP      37$
6625 031756 104054      28$: ERROR   54
6626 031750 000137 030660      JMP     ER2ENT     ;GO MAKE 2ND LEVEL REPORT
6627 031764 032704 000400      29$: BIT      #HVRC,R4      ;TEST IF HVRC ERROR
6628 031770 001457      BEQ      23$
6629 031772 052737 000004 005414      BIS      #HVRCER,RECODE
6630 032000 105737 003120      TSTB    DERCNT      ;TEST IF FIRST ERROR
6631 032004 001402      BEQ      30$      ;YES - REPORT
6632 032006 000137 032372      JMP     37$      ;JUMP TO RETURN
6633 032012 032737 000400 005414 30$: BIT      #LEV2ER,RECODE ;TEST IF A 2ND LEVEL ERROR HAS ALREADY
6634 032020 001403      BEQ      31$      ;OCCURRED. NO-SKIP EXIT
6635 032022 104055      51$: ERROR   55
6636 032024 000137 032372      JMP     37$      ;GET OUT OF ERROR REPORT
6637 032030 004737 033600      31$: JSR     PC,BLDEXH ;GO BUILD EXPECTED HEADER
6638 032034 004737 033270      JSR     PC,ROH00    ;GO GET HDR 0
6639 032040 032737 000400 005414      BIT      #LEV2ER,RECODE ;TEST IF ERROR IN GETTING HDR
6640 032046 001025      BNE      32$      ;IF YES-GO MAKE ABBREVIATED REPORT
6641 032050 013702 003036      MOV     RKBAS,R2   ;GET RK611 BASE ADDRESS
6642 032054 042762 000100 000000      BIC     #IE,RKCS1(R2) ;CLEAR INTERRUPT ENABLE
6643 032062 016237 000024 001202      MOV     RKDB(R2),R10 ;STORE OFF HEADER
6644 032070 016237 000024 001204      MOV     RKDB(R2),R11
6645 032076 016237 000024 001206      MOV     RKDB(R2),R12
6646 032104 032762 100000 000000      BIT      #CERR,RKCS1(R2) ;TEST IN ANY ERROR IN UNLOAD
6647 032112 001343      BNE      51$      ;IF YES - GO MAKE SHORT REPORT
6648 032114 104031      ERROR   31      ;MAKE FULL REPORT
6649 032116 000137 032372      JMP     37$

```

REPORT ERROR ROUTINE

6650	032122	104055			32%:	ERROR	55	ABREVIATED HVRC ERROR RPORT
6651	032124	000137	030660			JMP	ER2ENT	GO REPORT 2ND LEVEL ERROR
6652	032130	032704	000200		23%:	BIT	#BSE,R4	TEST FOR BAD SECTOR ERROR
6653	032134	001436				BEQ	33%	NO - SKIP
6654	032136	052737	000002	005414		BIS	#BSERR,RECODE	SET ERROR FLAG
6655	032144	016500	000030			MOV	P.DCYL(R5),R0	GET CYL NO.
6656	032150	116501	000027			MOV	P.DTS+1(R5),R1	GET TRACK NO.
6657	032154	042701	177774			BIC	#177774,R1	CLEAR ALL BITS EXCEPT TRACK
6658	032160	016502	000026			MOV	P.DTS(R5),R2	GET SECTOR IN ERROR
6659	032164	042702	177740			BIC	#177740,R2	CLEAR ALL BITS EXCEPT SECTOR
6660	032170	004737	027176			JSR	PC,BOSACK	GO SEE IF THIS SECTOR LISTED
6661	032174	032737	001000	005414		BIT	#BADSEC,RECODE	TEST RESULT
6662	032202	001073				BNE	37%	YES - EXIT, NO ERROR OR REPORT
6663	032204	105737	003133			TSTB	TSITYP	SEE IF READING WITH OFFSET
6664	032210	001403				BEQ	56%	BR IF NOT
6665	032212	105037	003125			CLRB	ERRCNT	CLEAR ERROR COUNT
6666	032216	000465				BR	37%	CONTINUE
6667	032220	104104			56%:	ERROR	104	ELSE REPORT BAD BSE
6668	032222	042737	000002	005414		BIC	#BSERR,RECODE	RESET BSE ERROR FLAG
6669	032230	000460				BR	37%	GO EXIT
6670	032232	032704	100000		33%:	BIT	#DCK,R4	TEST IF DATA CHECK
6671	032236	001435				BEQ	36%	
6672	032240	052737	000020	005414		BIS	#DCKERR,RECODE	SET DATA CHECK ERROR IN RECOVERY CODE
6673	032246	032704	000100			BIT	#ECH,R4	TEST IF ECC IS HARD. IF
6674	032252	001406				BEQ	34%	YES SET UNCORRECTABLE IN
6675	032254	052737	000040	005414		BIS	#ECCNC,RECODE	RECOVERY FLAG AND A 0 IN
6676	032262	005037	001206			CLR	\$REG12	REG12 TO INDICATE UNCORRECTABLE,
6677	032266	000403				BR	35%	A 1 IN REG 12 FOR CORRECTABLE
6678	032270	012737	000001	001206	34%:	MOV	#1,\$REG12	
6679	032276	105737	003120		35%:	TSTB	DERCNT	TEST IF FIRST ERROR
6680	032302	001033				BNE	37%	NO SKIP REPORT
6681	032304	004737	033444			JSR	PC,GTPKAD	GO GET PACK ADDRESS OF ERROR
6682	032310	016537	000060	001202		MOV	P.EPOS(R5),\$REG10	STORE ECC POSITION &
6683	032316	016537	000062	001204		MOV	P.EPAT(R5),\$REG11	PATTERN
6684	032324	104032				ERROR	32	REPORT DCK ERROR
6685	032326	000137	032372			JMP	37%	
6686	032332	032765	040000	000020	36%:	BIT	#WCE,P.CS2(R5)	TEST WRITE CHECK ERROR
6687	032340	001414				BEQ	37%	
6688	032342	042737	000200	005414		BIC	#ABORT,RECODE	CLEAR ABORT & SET WRITE
6689	032350	052737	000100	005414		BIS	#WCERR,RECODE	CHECK ERROR IN RECODE
6690	032356	105737	003120			TSTB	DERCNT	TEST IF FIRST ERROR
6691	032362	001003				BNE	37%	NO - SKIP
6692	032364	004737	033444			JSR	PC,GTPKAD	GO GET ADDRESS OF ERROR
6693	032370	104033				ERROR	33	REPORT WCE
6694								
6695	032372	032765	000020	000014	37%:	BIT	#DRVHRD,P.PRST(R5)	TEST HARD ERROR
6696	032400	001404				BEQ	43%	
6697	032402	104036				ERROR	36	
6698	032404	052737	000200	005414		BIS	#ABORT,RECODE	
6699								
6700	032412	032765	000040	000014	43%:	BIT	#DRVDSC,P.PRST(R5)	TEST STATUS CHANGE NOT CLEARED
6701	032420	001404				BEQ	44%	
6702	032422	104037				ERROR	37	
6703	032424	052737	000200	005414		BIS	#ABORT,RECODE	
6704								
6705	032432	032765	004000	000014	44%:	BIT	#NODSC,P.PRST(R5)	IFST ATTENTION BUT NO FAULT OR DSC

REPORT ERROR ROUTINE

```

6706 032440 001404          BEQ      46$
6707 032442 104040          ERROR   40
6708 032444 052737 000200 005414  BIS     #ABORT,RECODE
6709
6710 032452 032765 000010 000014 46$:   BIT     #UEXATT,P.PRST(R5) ;TEST UNEXPECTED ATTENTION
6711 032460 001404          BEQ     ALLTRM
6712 032462 104042          ERROR   42
6713 032464 052737 000200 005414  BIS     #ABORT,RECODE
6714
6715
6716                               ;ALL ERRORS MUST EXIT THROUGH THIS POINT
6717
6718 032472 012705 002630          ALLTRM: MOV    #PARAM,RS          ;RESTORE PARAMETER BLOCK SELECTION
6719 032476 012737 020204 003046  MOV    #DOCKHDL,A.ABNL      ;SET READ ERROR HANDLER
6720 032504 105737 003133          TSTB   TSTTYP              ;SEE IF READING WITH OFFSETS
6721 032510 001003          BNE    20$                 ;BR IF YES
6722 032512 012737 030640 003046  MOV    #ERRHDL,A.ABNL      ;RESTORE INTERRUPT VECTORS FOR RETRY
6723 032520 012737 027426 003044 20$:   MOV    #ERRFRE,A.NCRM      ;DRIVER OPERATIONS, IF ANY
6724 032526 032737 000200 005414  BIT    #ABORT,RECODE       ;IF ABORT IS NOT SET AND
6725 032534 001057          BNE    48$                 ;THE DRIVE IS READY (HAS NOT
6726                                     ;CYCLED DOWN)
6727 032536 013702 003036          MOV    RKBAS,R2           ;GET BASE ADDRESS
6728 032542 032762 000200 000012  BIT    #RDY,RKDS(R2)       ;TEST IF DRIVE READY SET
6729 032550 001004          BNE    47$                 ;RECALIBRATE REQUIRED BIT IS SET
6730 032552 052737 000200 005414  BIS    #ABORT,RECODE       ;ELSE ABORT WITH ABORT MESSAGE
6731 032560 000445          BR     48$
6732 032562 032737 004000 005414 47$:   BIT    #RCLREQ,RECODE      ;IF RECALIBRATE IS REQUIRED
6733 032570 001451          BEQ    BGNRTY              ;FOR RETRY SET UP PARAM
6734 032572 013737 002630 002714  MOV    PARAM,PARAM1       ;SAVE COMMAND
6735 032600 112765 000113 000001  MOVB   #RECAL,P.CMD(R5)   ;BLOCK TO DO IT.
6736 032606 012737 033072 003046  MOV    #RETANL,A.ABNL
6737 032614 004737 027302          JSR    PC,DRVCL
6738 032620 013737 002714 002630  MOV    PARAM1,PARAM       ;RESTORE COMMAND
6739 032626 012737 020204 003046  MOV    #DOCKHDL,A.ABNL    ;SET READ ERROR HANDLER
6740 032634 105737 003133          TSTB   TSTTYP              ;SEE IF READING WITH OFFSETS
6741 032640 001003          BNE    50$                 ;BR IF YES
6742 032642 012737 030640 003046  MOV    #ERRHDL,A.ABNL      ;RESTORE ERROR RETURN
6743 032650 032737 000400 005414 50$:   BIT    #LEVZER,RECODE      ;IF AN ERROR OCCURRED IN THE
6744 032656 001416          BEQ    BGNRTY              ;RECAL ATTEMPT SET ABORT
6745 032660 052737 000200 005414  BIS    #ABORT,RECODE       ;PRINT THE RECAL ERROR MESSAGE, AND
6746 032666 104060          ERROR   60                 ;GO REPORT DETAILS
6747 032670 000137 030660          JMP    ERZENT
6748 032674 104061          48$:   ERROR   61                 ;REPORT ABORT-RETRY FAILED
6749
6750                               ;THE PROGRAM WILL HALT HERE IF THE ABORT FLAG HAS BEEN SET OR
6751                               ;IF THE DRIVE READY BIT IS RESET.
6752
6753 032676 000005          HLTPRG: RESET              ;DISABLE ALL DEVICES
6754 032700 000000          HALT                          ;HALT THE CPU
6755 032702 105037 003125          CLRB   ERRCNT              ;CLEAR ERROR COUNT
6756 032706 000005          RESET              ;RESET ALL DEVICES
6757 032710 000137 012006          JMP    CMSTR
6758
6759
6760                               ;THE FOLLOWING CODE WILL DETERMINE IF AND DATA TYPE ERROR
6761                               ;HAS OCCURRED. IF YES, THE RETRY IS NOT DONE HERE BUT RETURNS TO

```

REPORT ERROR ROUTINE

```

6762
6763
6764
6765
6766 032714 032737 000136 005414
6767 032722 001404
6768 032724 052737 100000 005414
6769 032732 000453
6770 032734
6771 032734 105737 003140
6772 032740 001371
6773 032742 032737 001000 005414
6774 032750 001044
6775 032752 123737 003125 003126
6776 032760 001012
6777 032762 005037 001174
6778 032766 113737 003125 001174
6779 032774 104102
6780 032776 052737 000200 005414
6781 033004 000734
6782 033006 013702 003036
6783 033012 112765 000177 000001
6784 033020 004737 027302
6785 033024 012700 005166
6786 033030 012025
6787 033032 012025
6788 033034 012025
6789 033036 012025
6790 033040 012025
6791 033042 012025
6792 033044 012705 002630
6793 033050 012737 000000 177776
6794 033056 004737 027302
6795 033062 105037 003125
6796 033066 104410
6797 033070 000207
6798
6799
6800 033072 152737 000377 003122
6801 033100 052737 000400 005414
6802 033106 000207
6803 033110 152737 000377 003122
6804 033116 000207
6805
6806
6807
6808
6809
6810
6811
6812 033120
6813 033120 104407
6814 033122 013702 003036
6815 033126 012701 001174
6816 033132 016221 000000
6817 033136 016221 000010

```

```

; THE INITIATING ROUTINE FOR RETRY. ANY OTHER ERROR IS TO BE
; RETRIED HERE. IF RETRY IS UNSUCCESSFUL AFTER 4 ATTEMPTS, THE ABORT
; FLAG IS SET AND PROGRAM HALTS.

```

```

BGNRTY: BIT #BSERR!HVCRCR!OPIERR!DCKERR!WCERR,RECODE ;TEST IF ANY DATA ERROR
          BEQ 3$
          BIS #ANYDER,RECODE
          BR 2$
3$:
          TSTB NORTRY ;SEE IF "NO-RETRY" FLAG SET
          BNE 9$ ;BR IF YES
          BIT #BADSEC,RECODE ;TEST IF BAD SECTOR FLAG SET
          BNE 2$ ;IF YES-EXIT TO CALLER
          CMPB ERRCNT,ERRLMT ;BEGIN RETRY IF ERROR COUNT HAS
          BNE 1$ ;NOT BEEN EXCEEDED
          CLR $REGS
          MOVB ERRCNT,$REGS ;GET ERROR RETRY COUNT
          ERROR 102 ;REPORT RETRY UNSUCCESSFUL
          BIS #ABORT,RECODE ;SET ABORT & QUIT
          BR HLTPRG
1$:
          MOV RKBAS,R2 ;GET RK BASE ADDRESS
          MOVB #SUBCLR,P.CMND(R5) ;SET UP TO CLEAR SUBSYSTEM
          JSR PC,DRVCL ;GO DO IT
          MOV #COMSTR,R0 ;GO AND REESTABLISH THE COMMAND
          MOV (R0)+,(R5)+ ;AS IT WAS ENTERED INTO THE
          MOV (R0)+,(R5)+ ;PARAMETER BLOCK
          MOV (R0)+,(R5)+
          MOV (R0)+,(R5)+
          MOV (R0)+,(R5)+
          MOV (R0)+,(R5)+
          MOV #PARAM,R5
          MOV #PRO,PSW ;LOWER PRIORITY TO ALLOW INTERRUPT
          JSR PC,DRVCL ;CALL DRIVER
          CLRB ERRCNT ;CLEAR ERROR COUNT
          RESREG ;IF RETURN GETS HERE, NO ERROR
          RTS PC ;OCCURRED, RECOVERY WAS SUCCESSFUL

```

```

RETANL: BISB #377,DONE ;SET DONE
          BIS #LEV2ER,RECODE ;SET LEVEL TWO ERROR
          RTS PC
RETAML: BISB #377,DONE ;SET DONE
          RTS PC

```

```

;*****
;SBTTL TIME OUT PROCESSOR ROUTINE
;THIS ROUTINE SUPPORTS THE ERROR HANDLER BY PROCESSING TIME OUT STATUS
;GATHERING DUTIES.
;*****

```

```

TOPROC:
          SAVREG
          MOV RKBAS,R2
          MOV #REGS,R1 ;SET UP R1 FOR RK REGISTER STORAGE
          MOV RKCS1(R2),(R1)+ ;STORE ALL VALID RK611
          MOV RKCS2(R2),(R1)+ ;REGISTERS

```

```

6818 033142 016221 000020      MOV      RKDC(R2), (R1)+
6819 033146 016221 000006      MOV      RKDA(R2), (R1)+
6820 033152 016221 000002      MOV      RKWC(R2), (R1)+
6821 033156 016221 000004      MOV      RKBA(R2), (R1)+
6822 033162 016221 000016      MOV      RKASOF(R2), (R1)+
6823 033166 016221 000012      MOV      RKDS(R2), (R1)+
6824 033172 016221 000014      MOV      RKER(R2), (R1)+
6825 033176 005000      CLR      RO
6826
6827
6828 033200 012705 002714      MOV      #PARI, RS
6829 033204 112765 000141 000001      MOVVB   #RSTAT, P.CMND(RS) ; DO READ DRIVE STATUS COMMAND
6830 033212 004737 027302      JSR      PC, DRVCL
6831 033216 032765 000100 000014      BIT      #CMDO, P.PRST(RS) ; TEST FOR TIMEOUT
6832 033224 001403      BEQ      IS
6833 033226 052737 002000 005414      BIS      #TWOTOS, RECODE ; SET TWO TIMEOUTS FLAG
6834 033234 062705 000040      ADD      #P, ADRS ; BUMP RS TO POINT TO DRIVE STATUS
6835 033240 012521      MOV      (RS)+, (R1)+ ; MOVE ALL THE DRIVE STATUS INTO THE
6836 033242 012521      MOV      (RS)+, (R1)+ ; TEMP REGS FOR REPORTING.
6837 033244 012521      MOV      (RS)+, (R1)+
6838 033246 012521      MOV      (RS)+, (R1)+
6839 033250 012521      MOV      (RS)+, (R1)+
6840 033252 012521      MOV      (RS)+, (R1)+
6841 033254 012521      MOV      (RS)+, (R1)+
6842 033256 012521      MOV      (RS)+, (R1)+
6843
6844 033260 012705 002630      MOV      #PARMO, RS ; RESTORE PARM 0
6845 033264 104410      RESREG
6846 033266 000207      RTS      PC
6847
6848
6849
6850
6851
6852
6853 033270
6854 033270 104407
6855 033272 016501 000026      MOV      SAVREG
6856 033276 016500 000052      MOV      P.DTS(RS), R1 ; STORE TRACK AND SECTOR
6857 033302 042700 160017      BIC      P.BID(RS), RO ; GET THE CYLINDER AORS
6858 033306 006200      ASR      #160017, RO ; FROM THE DRIVE STATUS. CLEAR
6859 033310 006200      ASR      RO ; OFF UNUSED BITS AND POSITION
6860 033312 006200      ASR      RO ; FOR USE AS THE DESIRED
6861 033314 006200      ASR      RO ; CYLINDER IN THE READ
6862 033316 012705 002714      MOV      #PARI, RS ; SET UP TO USE P.B. 1
6863 033322 010165 000004      MOV      R1, P.SECT(RS) ; INSERT TRK, SECT FOR READ HDR
6864 033326 010065 000002      MOV      RO, P.CYLN(RS) ; SET CYL NO.
6865 033332 112765 000177 000001      MOVVB   #SUBCLR, P.CMND(RS) ; SET S.S. CLEAR CMND
6866 033340 012737 000000 177776      MOV      #PRO, #PSW ; ALLOW INTERRUPTS
6867 033346 004737 027302      JSR      PC, DRVCL ; INIT THE S.S.
6868 033352 112765 000125 000001      MOVVB   #RDRHD, P.CMND(RS) ; SET READ HEADER COMMAND
6869 033360 004737 027302      JSR      PC, DRVCL ; DO A READ HEADER
6870 033364 012762 000025 000000      MOV      #25, RKCS1(R2) ; DO A READ HDR, WITH FMT BIT = 0
6871 033372 032762 000200 000000 245:      BIT      #RDY, RKCS1(R2) ; WAIT FOR READY
6872 033400 001774      BEQ      245
6873 033402 012762 010025 000000      MOV      #10025, RKCS1(R2) ; DO A READ HDR, WITH FMT BIT = 1

```

```

; THIS CODE WILL ATTEMPT TO
; RETRIEVE THE STATUS FROM THE
; DRIVE.
; SET UP TO USE PARM1
; DO READ DRIVE STATUS COMMAND
; CALL DRIVER
; TEST FOR TIMEOUT
; NO - SKIP
; SET TWO TIMEOUTS FLAG
; BUMP RS TO POINT TO DRIVE STATUS
; MOVE ALL THE DRIVE STATUS INTO THE
; TEMP REGS FOR REPORTING.

```

```

; *****
; SBTTL READ HEADER 0 ROUTINE
; *****
RDHDO:

```

```

6874 033410 032762 000200 0C 1000 26$: BIT #RDY,RKCSI(R2) ;WAIT FOR READY
6875 033416 001774 BEQ 26$
6876 033420 142765 000020 000007 BICB #B.CFMT,P.CSIH(R5) ;CLEAR THE FORMAT BIT
6877 033426 153765 003124 000007 BISB FORMAT,P.CSIH(R5) ;RESTORE TYPE AND FMT IN USE
6878 033434 012705 002630 MOV #PARMO,R5 ;RESTORE P.B. 0 ADDRESS
6879 033440 104410 RESREG ;RESTORE R0-R5
6880 033442 000207 RTS PC ;RETURN
6881
6882
6883
6884
6885
6886
6887 033444 016537 000030 001174 GTPKAD: MOV P.DCYL(R5),SREG5 ;GET CYLINDER NUMBER
6888 033452 005037 001176 CLR SREG6 ;CLEAR REGISTERS FOR
6889 033456 005037 001200 CLR SREG7 ;TRACK & SECTOR STORAGE
6890 033462 116537 000026 001200 MOVB P.DTS(R5),SREG7 ;STORE THE TRACK AND SECTOR
6891 033470 116537 000027 001176 MOVB P.DTS+1(R5),SREG6
6892 033476 032737 000004 005414 BIT #HVR CER,RECODE ;SEE IF HVRC ERROR
6893 033504 001034 BNE 5$ ;BR IF YES
6894 033506 005737 001200 TST SREG7 ;ADJUST THE ADDRESS CONTAINED IN
6895 ;THE RK REGISTERS FOR THE AUTOMATIC
6896 ;INCREMENT
6896 033512 001403 BEQ 1$
6897 033514 005337 001200 DEC SREG7
6898 033520 000426 BR 5$
6899 033522 032765 010000 000016 1$: BIT #CFMT,P.CSI(R5)
6900 033530 001404 BEQ 2$
6901 033532 012737 000023 001200 MOV #19.,SREG7
6902 033540 000403 BR 3$
6903 033542 012737 000025 001200 2$: MOV #21.,SREG7
6904 033550 005737 001176 3$: TST SREG6
6905 033554 001403 BEQ 4$
6906 033556 005337 001176 DEC SREG6
6907 033562 000405 BR 5$
6908 033564 012737 000002 001176 4$: MOV #2,SREG6
6909 033572 005337 001174 DEC SREG5
6910 033576 000207 5$: RTS PC
6911
6912
6913
6914
6915
6916
6917
6918
6919
6920 033600 104407 BLDEXH: SAVREG
6921 033602 016537 000030 001174 MOV P.DCYL(R5),SREG5 ;CONSTRUCT EXPECTED HDR
6922 033610 016501 000026 MOV P.DTS(R5),R1 ;DESIRED CYLINDER & DESIRED TRACK
6923 033614 042701 174377 BIC #174377,R1 ;CLEAR ALL BUT TRACK BITS
6924 033620 006201 ASR R1 ;AND SECTOR. SHIFT THE TRACK
6925 033622 006201 ASR R1 ;OVER TO CONFORM TO HEADER FORMAT
6926 033624 006201 ASR R1 ;CHECK THE FORMAT BIT AND
6927 ;IF SET, SET THE HEADER FORMAT
6928 033626 016537 000026 001176 MOV P.DTS(R5),SREG6 ;BIT.
6929 033634 042737 177740 001176 BIC #177740,SREG6 ;CLEAR ALL BUT SECTOR

```

```

:*****
:SBTTL GET PACK ADDRESS ROUTINE
:*****

```

```

:*****
:SBTTL BUILD EXPECTED HEADER
: *USES DESIRED CYLINDER, TRACK AND SECTOR REGISTERS TO DETERMINE
: *WHICH HEADER WAS EXPECTED. LOADS EXPECTED VALUES IN SREG5, 6, AND
: *7 FOR REPORTING.
:*****

```


F11

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 135
BUILD EXPECTED HEADER

SEQ 0134

6930	033642	060137	001176			ADD	R1,\$REG6	;ADD TRACK AND SECTOR TOGETHER
6931	033646	052737	140000	001176		BIS	#140000,\$REG6	;INSERT BSE BITS
6932	033654	032765	010000	000016		BIT	#CFMT,P.CS1(R5)	
6933	033662	001403				BEQ	23\$	
6934	033664	052737	001000	001176		BIS	#1000,\$REG6	
6935	033672	013737	001174	001200	23\$:	MOV	\$REG5,\$REG7	;COMPUTE THE HEADER VRC
6936	033700	013701	001176			MOV	\$REG6,R1	
6937	033704	043737	001176	001200		BIC	\$REG6,\$REG7	
6938	033712	043701	001174			BIC	\$REG5,R1	
6939	033716	050137	001200			BIS	R1,\$REG7	
6940	033722	104410				RESREG		
6941	033724	000207				RTS	PC	
6942								

6943
6944
6945
6946
6947
6948
6949
6950
6951
6952
6953
6954
6955
6956
6957
6958
6959
6960
6961
6962
6963
6964
6965
6966
6967
6968
6969
6970
6971
6972
6973
6974
6975
6976
6977
6978
6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993
6994
6995
6996
6997
6998

033726 010546
033730 010446
033732 010346
033734 010246
033736 013746 177776
033742 005337 003056
033746 001034
033750 013737 003060 003056
033756 105737 003100
033762 001426
033764 013737 003042 177776
033772 013702 003036
033776 005337 003114
034002 001016

.SBTTL RK611/RK06 UNIBUS DRIVER FOR SEQUENTIAL OPERATIONS (REV. 0.08)

;*COPYRIGHT (C) 1975
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MA. 01754
;*AUTHOR: ROY SPITZER

.SBTTL *WATCH-DOG TIMER

```
*****
*
* THE WATCH-DOG TIMER DOES A PSEUDO-TIMING OF RK06 UNIBUS
* SUBSYSTEM COMMAND. SINCE ONE CAN NOT GUARANTEE THAT A
* REAL-TIME CLOCK (KW11-P OR KW11-L) IS ON THE SYSTEM
* THE RK06 DRIVER WILL USE THE LOCATION W.MTIM FOR
* MILLI-SECOND TIMING. WHEN W.MTIM REACHES ZERO THE
* WATCH-DOG TIMER WILL SCAN THE DRIVES IN USE AS
* DETERMINED BY THE LOCATION W.TIME. THE TIMER COUNTS
* (ONE FOR EACH DRIVE) ARE KEPT IN THE TABLE W.DRV.
* IF ANY COUNT IN THE TABLE W.DRV REACHES ZERO A COMMAND
* TIME-OUT WILL BE DESIGNATED IN THE PROGRAM DEVICE STATUS
* REGISTER OF THAT DRIVE'S PARAMETER BLOCK.
*
* THE DRIVER WILL USE THE LOCATION W.MIN AS THE NUMBER
* OF MILLISECONDS FOR AN UNLOAD OR START SPINDLE COMMAND.
* THE DRIVER WILL USE THE LOCATION W.SEC AS THE TIME
* LIMIT FOR ALL OTHER COMMANDS.
*
* FOR QUEUED OPERATIONS THE WATCH-DOG TIMER WILL
* WATCH UP TO 8 OPERATIONS SIMULTANEOUSLY. FOR SEQUENTIAL
* OPERATIONS ONLY ONE OPERATION WILL BE WATCHED.
*
*CALL JSR PC,W.WTCH
*      RETURN IF NO DRIVE ORDER EXCEEDED ITS TIME LIMIT
*
* OTHERWISE AN ABNORMAL RETURN TO THE ROUTINE ADDRESS
* BY LOCATION A.ABNL WILL OCCUR AND THE CMTO FLAG
* IN THE PROGRAM DEVICE STATUS REGISTER OF THE
* APPROPRIATE PARAMETER BLOCK WILL BE SET.
*****
```

```
W.WTCH: MOV R5,-(SP) ;SAVE R5 ON THE STACK
        MOV R4,-(SP) ;SAVE R4 ON THE STACK
        MOV R3,-(SP) ;SAVE R3 ON THE STACK
        MOV R2,-(SP) ;SAVE R2 ON STACK
        MOV PS,-(SP) ;SAVE PROGRAM STATUS WORD ON STACK
        DEC W.MTIM ;DECREMENT MILLISECOND TIMER
        BNE ZOS ;IF NOT ZERO RETURN
        MOV W.MILI,W.MTIM ;REINITIALIZE MILLISECOND TIMER
        TSTB W.TIME ;CHECK IF DRIVE IS BEING TIMED
        BEQ ZOS ;NO RETURN
        MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
        MOV RKBAS,R2 ;LOAD BASE OF RK06 REGISTERS
        DEC W.DRV ;DECREMENT COMMAND TIMER
        BNE ZOS ;RETURN IF NO TIME OUT
```

*WATCH-DOG TIMER

```

6999 034004 105037 003100      CLR      W.TIME      :RESET TIMING INDICATOR
7000 034010 013705 003112      MOV      PBLKT,R5    :LOAD ADDRESS OF PARAMETER BLOCK
7001                                :TABLE FOR INDEXING
7002 034014 052765 000100 000014  BIS      #CMDTO,P.PRST(R5) :SET COMMAND TIME OUT
7003 034022 020537 003054      CMP      R5,O.WAIT   :CHECK IF DRIVER IS WAITING FOR
7004                                :COMMAND COMPLETION
7005 034026 001002      BNE      SS          :NO, DO NOT ALTER WAITING FOR
7006                                :COMMAND COMPLETION
7007 034030 005037 003054      CLR      O.WAIT     :CLEAR WAIT FOR COMMAND COMPLETION
7008 034034 004737 037310      JSR      PC,R.ABNL  :BRANCH TO ERROR ROUTINE
7009 034040 012637 177776      MOV      (SP)+,PS   :RESTORE PSM
7010 034044 012602      MOV      (SP)+,R2   :RESTORE R2
7011 034046 012603      MOV      (SP)+,R3   :RESTORE R3
7012 034050 012604      MOV      (SP)+,R4   :RESTORE R4
7013 034052 012605      MOV      (SP)+,R5   :RESTORE R5
7014 034054 000207      RTS      PC        :RETURN

```

SS:
20\$:

7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044
7045
7046
7047
7048
7049
7050
7051
7052
7053
7054
7055
7056
7057
7058
7059
7060
7061
7062
7063
7064
7065
7066
7067
7068
7069
7070

.SBTTL *RK06 INTERRUPT SERVICE ROUTINE

```

*****
*
* THIS ROUTINE WILL SERVICE ALL RK06 INTERRUPTS.
*
* UPON RECEIVING AN INTERRUPT, THIS ROUTINE WILL
* PERFORM ONE OF THE FOLLOWING SERVICES:
*
* 1.) SERVICE PORT WAS SEIZED BY OTHER PORT
* 2.) SERVICE DRIVER IS WAIT FOR COMMAND COMPLETION
* 3.) SERVICE POSITIONING COMPLETION
* 4.) REQUEUE COMMAND IF DRIVE WAS RELEASED
*    FOR THE QUEUED RK06 DRIVER.
* 5.) IF NO SERVICE IS REQUIRED, THE COMMAND WILL BE ISSUED
*    FOR THE QUEUED RK06 DRIVER.
*
* THREE LINKS ARE PROVIDED TO THE DRIVING PROGRAM.
* THEY ARE:
*
* 1.) A.NORM ADDRESS OF NORMAL RETURN (SUCCESSFUL COMPLETION OF COMMAND)
* 2.) A.ABNL ADDRESS OF ABNORMAL RETURN (UNSUCCESSFUL COMPLETION OF COMMAND)
* 3.) A.CONT ADDRESS OF CONTROL ERROR RETURN
*
* FOR NORMAL AND ABNORMAL RETURNS, THE ADDRESS OF THE APPROPRIATE
* PARAMETER BLOCK WILL BE IN R5.
*
* FOR THE CONTROLLER ERROR RETURN, THE LOCATION E.CONT CONTAINS
* THE REASON FOR THE CONTROLLER ERROR.
*
* ROUTINES USED:
*   C.OPT (QUEUED ONLY)
*   Q.PUSH (QUEUED ONLY)
*   Q.RMOV (QUEUED ONLY)
*   R.CONT (SEQUENTIAL ONLY)
*   R.NORM (SEQUENTIAL ONLY)
*   R.ABNL (SEQUENTIAL ONLY)
*   I.CSTS
*   I.STAT
*   I.ISSU
*   I.CCLR
*****

```

```

034056 010546
034060 010446
034062 010346
034064 010246
034066 010146
034070 010046
034072 013702 003036
034076 016237 000010 003002
034104 032737 001000 003002
034112 001407
034114 052737 100000 003052
034122 004737 037334

```

```

I.INTR: MOV R5,-(SP) ;STORE R5 ON THE STACK
        MOV R4,-(SP) ;STORE R4 ON THE STACK
        MOV R3,-(SP) ;STORE R3 ON THE STACK
        MOV R2,-(SP) ;STORE R2 ON THE STACK
        MOV R1,-(SP) ;STORE R1 ON THE STACK
        MOV R0,-(SP) ;STORE R0 ON THE STACK
        MOV RKBAS R2 ;LOAD R2 TO ADDRESS RK06 REGISTER
        MOV RKCS2(R2),T.CS2 ;STORE CS2
        BIT #MDS,T.CS2 ;CHECK IF MULTIPLE DRIVE SELECT
        BEQ IS ;NO CONTINUE PROCESSING
        BIS #E.MDS,E.CONT ;SET MULTIPLE DRIVE SELECT
        JSR PC,R.CONT ;REPORT ERROR

```

```

7071 034126 000137 036306          JMP      I.RTRN          ;RETURN
7072
7073 034132 105737 003074          15:     TSTB      I.ISRL          ;CHECK IF INTERRUPT OR RELEASE
7074 034136 001410          BEQ      65             ;NO, CHECK IF DRIVE AVAILABLE
7075 034140 100403          BNY      55             ;CHECK IF RELEASE COMMAND
7076 034142 105037 003074          CLRB      I.ISRL          ;YES, CLEAR FLAG
7077 034146 000473          BR       I.I00          ;CONTINUE PROCESSING INTERRUPT
7078
7079 034150 105037 003074          55:     CLRB      I.ISRL          ;CLEAR FLAG
7080 034154 000137 035270          JMP      I.ATTN          ;GO PROCESS DRIVE ATTENTIONS
7081
7082 034160 032737 010400 003002      65:     BIT       #NED!UFE,T.CS2      ;CHECK FOR NON-EXISTENT DRIVE OR
7083                                     UNIT FIELD ERROR
7084 034166 001413          BEQ      75             ;NO, WAIT FOR DUAL ACCESS INTERRUPT
7085 034170 013704 003002          MOV      T.CS2,R4       ;LOAD R4 FOR DRIVE NUMBER
7086 034174 042704 177770          BIC      #1C<DRVMSK>,R4 ;KEEP DRIVE BITS
7087 034200 013705 003112          MOV      PBLKT,R5       ;STORE PARAMETER BLOCK ADDRESS
7088 034204 016237 000000 003000      MOV      RKCS1(R2),T.CS1 ;LOAD TEMPORARY CS1 FOR STATUS REPORT
7089 034212 000137 034500          JMP      I.ERRC          ;REPORT ERROR
7090
7091 034216 016237 000012 003020      75:     MOV      RKDS(R2),T.DS      ;STORE STATUS REGISTER FOR COMPARISON
7092 034224 032737 000001 003020      BIT      #DRA,T.DS       ;CHECK IF DRIVE SEIZED BY OTHER
7093                                     PORT
7094 034232 001041          BNE      I.I00          ;NO, CONTINUE PROCESSING INTERRUPT
7095
7096                                     ;CHECK IF ANY DATA TRANSFER ERROR EXISTS
7097 034234 032737 164000 003002      BIT      #DLT!WCE!UPE!NEM,T.CS2
7098
7099 034242 001007          BNE      105            ;INDICATE ERROR
7100 034244 016237 000014 003016      MOV      RKER(R2),T.ER   ;STORE ERROR REGISTER
7101
7102                                     ;CHECK FOR DATA TRANSFER ERROR TYPE ERROR
7103 034252 032737 125700 003016      BIT      #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
7104
7105 034260 001407          BEQ      115            ;NO, WAIT FOR RELEASE OF RK06 DRIVE
7106
7107 034262 052737 000010 003052      105:    BIS      #E.LDAT,E.CONT   ;SET UNEXPECTED DATA TYPE ERROR
7108 034270 004737 037334          JSR      PC,R.CONT      ;REPORT ERROR
7109 034274 000137 036306          JMP      I.RTRN          ;RESTORE REGISTERS
7110
7111 034300 105037 003100          115:    CLRB      W.TIME          ;RESET TIMING ON THIS DRIVE
7112 034304 005037 003114          CLR      W.DRV          ;CLEAR TIMING COUNT FOR THIS DRIVE
7113 034310 013705 003112          MOV      PBLKT,R5       ;LOAD R5 WITH PARAMETER BLOCK
7114                                     ADDRESS
7115 034314 052765 010000 000014      BIS      #DRVSZD,P.PRST(R5) ;SET DRIVE SEIZED IN THE
7116                                     PROGRAM DRIVE STATUS REGISTER
7117 034322 005037 003054          CLR      O.WAIT          ;CLEAR WAIT FOR COMMAND COMPLETION
7118 034326 004737 037310          JSR      PC,R.ABNL      ;INDICATE ABNORMAL TERMINATION
7119 034332 000137 036306          JMP      I.RTRN          ;GO RESTORE REGISTERS
7120
7121 034336 013705 003054          I.I00:  MOV      O.WAIT,R5       ;LOAD PARAMETER BLOCK ADDRESS INTO R5
7122 034342 001002          BNE      25             ;IS COMMAND WAITING PROCESSING
7123                                     YES, DO PROCESSING
7124 034344 000137 035270          JMP      I.ATTN          ;NO, PROCESS ATTENTION
7125
7126 034350 013704 003002          25:     MOV      T.CS2,R4       ;STORE RKCS2 FOR DRIVE NUMBER

```

K11

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 140
 #RK06 INTERRUPT SERVICE ROUTINE

SEQ 0139

7127	034354	042704	177770			BIC	#IC<DRVMSK>,R4	;MASK OUT UNNECESSARY BITS
7128								
7129								
7130	034360	126504	000000			CMPB	P.DRVN(R5),R4	;CHECK IF DRIVE NUMBER IS EXPECTED
7131	034364	001401				BEQ	3\$;YES, CONTINUE
7132	034366	000000				HALT		;NO, DRIVER ERROR
7133	034370	122765	000164	000001	3\$:	CMPB	#RDALHD,P.CMND(R5)	;CHECK IF READ ALL HEADERS
7134	034376	001002				BNE	10\$;NO, EXECUTE NORMAL DATA TRANSFER
7135	034400	000137	034736			JMP	I.HDAL	;GO EXECUTE SPECIAL HEADER SEQUENCE
7136								
7137	034404	005037	003054		10\$:	CLR	O.WAIT	;CLEAR WAIT FOR COMMAND COMPLETION
7138	034410	005037	003114			CLR	W.DRV	;CLEAR WATCH-DOG TIME
7139	034414	105037	003100			CLRB	W.TIME	;RESET TIMING ON THIS DRIVE
7140	034420	016237	000000	003000		MOV	RKCSI(R2),T.CS1	;STORE COMMAND AND STATUS REGISTER 1
7141	034426	032737	100000	003000		BIT	#CERR,T.CS1	;CHECK IF CONTROLLER ERROR
7142	034434	001021				BNE	I.ERRC	;YES, PROCESS ERROR
7143	034436	016237	000016	003014		MOV	RKASOF(R2),T.ASOF	;STORE ATTENTION SUMMARY
7144	034444	133737	003101	003015		BITB	INTMSK,T.ASOF+1	;CHECK IF DRIVE ATTENTION SET
7145	034452	001004				BNE	15\$;YES, REPORT ERROR
7146	034454	004737	037322			JSR	PC.R.NORM	;INDICATE NORMAL RETURN
7147	034460	000137	036306			JMP	I.RTRN	;RESTORE REGISTERS
7148								
7149	034464	052765	000010	000014	15\$:	BIS	#UEXATT,P.PRST(R5)	;SET UNEXPECTED ATTENTION
7150								
7151	034472	004737	036756			JSR	PC.I.CSTS	;STORE CONTROLLER STATUS
7152	034476	000405				BR	I.ERR	;STORE PATTERN AND POSITION INFORMATION
7153								
7154	034500	013765	003000	000016	I.ERRC:	MOV	T.CS1,P.CS1(R5)	;GET ERROR RKCSI
7155	034506	004737	037000			JSR	PC.I.CST1	;GET REST OF CONTROLLER STATUS
7156	034512	016265	000032	000062	I.ERR:	MOV	RKECPT(R2),P.EPAT(R5)	;STORE ECC PATTERN
7157	034520	016265	000030	000060		MOV	RKECPS(R2),P.EPOS(R5)	;STORE ECC POSITION
7158	034526	004037	036324			JSR	RO,I.CCLR	;CLEAR CONTROLLER
7159	034532	036306				I.RTRN		;ERROR RETURN
7160	034534	032765	010400	000020		BIT	#MED!LFE,P.CS2(R5)	;CHECK IF IT WAS NON-EXISTENT DRIVE OR
7161								UNIT FIELD ERROR
7162	034542	001046				BNE	5\$;YES, REPORT ERROR
7163	034544	004037	037062			JSR	RO,I.STAT	;GATHER DRIVE STATUS
7164	034550	036306				I.RTRN		;ERROR RETURN
7165	034552	112737	000005	003000		MOV	#DR.CLR,T.CS1	;LOAD COMMAND
7166	034560	004037	036406			JSR	RO,I.ISSU	;ISSUE DRIVE CLEAR
7167	034564	036306				I.RTRN		;ERROR RETURN
7168	034566	133737	003101	003015		BITB	INTMSK,T.ASOF+1	;CHECK IF ATTENTION RESET
7169	034574	001407				BEQ	2\$;NO, INDICATE DRIVE ERROR
7170	034576	052737	000020	003052		BIS	#E.CLAT,E.CONT	;SET ATTENTION DID NOT RESET
7171								WITH CLEAR
7172	034604	004737	037334			JSR	PC.R.CONT	;REPORT CONTROLLER ERROR
7173	034610	000137	036306			JMP	I.RTRN	;GO RESTORE REGISTERS
7174								
7175	034614	032737	040000	003024	2\$:	BIT	#S.DSC,T.MR2	;CHECK IF DRIVE STATUS CHANGE CLEARED
7176	034622	001403				BEQ	3\$;YES, CHECK FAULT
7177	034624	052765	000040	000014		BIS	#DRVDSC,P.PRST(R5)	;SET DSC DID NOT CLEAR
7178	034632	032737	001000	003026	3\$:	BIT	#S.PAR,T.MR3	;CHECK IF DRIVE PARITY ERROR
7179	034640	001407				BEQ	5\$;NO, INDICATE ABNORMAL TERMINATION
7180	034642	052737	002000	003052		BIS	#E.DPAR,E.CONT	;SET DRIVE PARITY ERROR
7181	034650	004737	037334			JSR	PC.R.CONT	;INDICATE CONTROLLER ERROR
7182	034654	000137	036306			JMP	I.RTRN	;RETURN

```

7183
7184 034660 032765 000020 000014 5$: BIT #ORVHRD,P.PRST(R5) ;CHECK IF HARD DRIVE ERROR
7185 034666 001017 10$: BNE 10$ ;YES, GO REPORT ERROR
7186 034670 032737 020000 003024 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS CYCLING DOWN
7187 034676 001413 10$: BEQ 10$ ;NO, REPORT ERROR
7188 034700 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING
7189 034706 113737 003101 003100 MOV# INTMSK,W.TIME ;SET UP 8 SECONDS FOR DRIVE TO CYCLE UP
7190 034714 013737 003064 003114 MOV W.BSEC,W.DRV
7191 034722 000137 036306 JMP I.RTRN ;GO RESTORE REGISTERS
7192
7193 034726 004737 037310 10$: JSR PC.R.ABNL ;GO REPORT ERROR
7194 034732 000137 036306 JMP I.RTRN ;GO RESTORE REGISTERS
7195
7196 .SBTTL *READ ALL HEADERS INTERRUPT SEQUENCE
7197
7198 034736 016237 000000 003000 I.HDAL: MOV RKCS1(R2),T.CS1 ;STORE CS1 TO CHECK CONTROLLER
7199 ERROR
7200 034744 032737 100000 003000 BIT #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR
7201 034752 001422 5$: BEQ 5$ ;NO, CHECK FOR ATTENTION
7202
7203 034754 005037 003054 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETE
7204 034760 105037 003100 CLR# W.TIME ;RESET TIMING ON DRIVE
7205 034764 005037 003114 CLR W.DRV ;CLEAR TIME OUT COUNT
7206 034770 013765 003000 000016 MOV T.CS1,P.CS1(R5) ;STORE ERROR RKCS1
7207 034776 004737 037000 JSR PC,I.CST1 ;STORE CONTROLLER REGISTERS
7208 035002 004037 036324 JSR RO,I.CCLR ;CLEAR CONTROLLER
7209 035006 036306 I.RTRN ;ERROR RETURN
7210 035010 004737 037310 JSR PC.R.ABNL ;INDICATE ERROR RETURN
7211 035014 000137 036306 JMP I.RTRN ;RESTORE REGISTERS
7212
7213 035020 016237 000016 003014 5$: MOV RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
7214 035026 133737 003101 003015 BIT# INTMSK,T.ASOF+1 ;CHECK IF DRIVE ATTENTION IS SET
7215 035034 001410 7$: BEQ 7$ ;NO, CHECK IF READ ALL HEADERS
7216 035036 005037 003054 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETION
7217 035042 105037 003100 CLR# W.TIME ;RESET TIMING ON DRIVE
7218 035046 005037 003114 CLR W.DRV ;CLEAR TIME OUT COUNT
7219 035052 000137 034472 JMP I.ERRA ;GO REPORT ERROR
7220
7221 035056 013701 003070 7$: MOV HDR.AO,R1 ;GET MAIN MEMORY ADDRESS
7222 035062 016221 000024 MOV RKDB(R2),(R1)+ ;GET FIRST WORD OF HEADER
7223 035066 016221 000024 MOV RKDB(R2),(R1)+ ;GET SECOND WORD OF HEADER
7224 035072 016221 000024 MOV RKDB(R2),(R1)+ ;GET THIRD WORD OF HEADER
7225 035076 010137 003070 MOV R1,HDR.AO ;STORE ADDRESS FOR NEXT HEADER
7226 035102 016237 000010 003002 MOV RKCS2(R2),T.CS2 ;STORE CS2 TO CHECK FOR DATA LATE
7227 035110 032737 100000 003002 BIT #DLT,T.CS2 ;CHECK FOR DATA LATE
7228 035116 001055 BNE 35$ ;YES, REPORT ERROR
7229 035120 005337 003072 DEC HDR.CT ;DECREMENT NUMBER OF HEADER YET TO READ
7230 035124 001026 BNE 25$ ;IF NON-ZERO, GO ISSUE NEXT READ HEADER
7231 035126 005037 003054 CLR O.WAIT ;CLEAR DRIVER WAITING FOR COMMAND COMPLETION
7232 035132 005037 003114 CLR W.DRV ;CLEAR TIME OUT COUNT FOR THIS DRIVE
7233 035136 105037 003100 CLR# W.TIME ;CLEAR WATCH DOG TIME ON THIS DRIVE
7234 035142 012762 000003 000026 MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER FOR SECTOR COUNT
7235 035150 112737 000001 003000 MOV# #DR.SEL,T.CS1 ;LOAD SELECT COMMAND
7236 035156 004037 036406 JSR RO,I.ISSU ;GET SECTOR COUNT
7237 035162 036306 I.RTRN ;ERROR RETURN
7238 035164 013765 003026 000056 MOV T.MR3,P.B11(R5) ;LOAD SECTOR COUNT

```

M11

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 142
 *READ ALL HEADERS INTERRUPT SEQUENCE

SEQ 0141

```

7239 035172 004737 037322 JSR PC,R,NORM ;INDICATE NORMAL TERMINATION
7240 035176 000137 036306 JMP I,RTRN ;RESTORE REGISTERS
7241
7242 035202 016562 000002 000020 25$: MOV P,CYL(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
7243 035210 016562 000004 000006 MOV P,SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
7244 035216 116565 000007 000017 MOV P,CSIH(R5),P,CSI+1(R5) ;STORE BITS 8-15 OF CSI
7245 035224 042765 165777 000016 BIC #1C<CDT!CFHT>,P,CSI(R5) ;CLEAR ALL BITS EXCEPT FORMAT AND
; DRIVE TYPE
7246
7247 035232 112765 000125 000016 MOV #RDHEAD,P,CSI(R5) ;STORE COMMAND ISSUED
7248 035240 016562 000016 000000 MOV P,CS,(R5),RKCSI(R2) ;ISSUE READ HEADER
7249 035246 000137 036306 JMP I,RTRN ;RESTORE REGISTERS
7250
7251 035252 052737 000400 003052 35$: BIS #E.DLT,E.CONT ;SET DATA LATE WHILE UNLOADING HEADER
7252 035260 004737 037334 JSR PC,R,CONT ;REPORT ERROR
7253 035264 000137 036306 JMP I,RTRN ;RESTORE REGISTERS
7254
7255 .SBTTL #DRIVE ATTENTION SCANNER
7256
7257 035270 016237 000000 003000 I.ATTN: MOV RKCSI(R2),T,CSI ;STORE COMMAND AND STATUS
; REGISTER 1 FOR COMPARISON
7258
7259 035276 032737 100000 003000 BIT #CERR,T,CSI ;CHECK IF CONTROLLER ERROR OCCURRED
7260 035304 001441 BEQ SS ;NO, CHECK IF ATTENTION
7261
7262 ;CHECK IF ANY DATA TRANSFER TYPE ERROR EXISTS
7263 035306 032737 164000 003002 BIT #DLT!WCE!UPE!NEM,T,CS2
7264
7265 035314 001007 BNE IS ;INDICATE ERROR
7266 035316 016237 000014 003016 MOV RKER(R2),T,ER ;STORE ERROR REGISTER
7267
7268 ;
7269 035324 032737 125700 003016 BIT #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T,ER ;CHECK FOR DATA TRANSFER ERROR TYPE
7270
7271 035332 001407 BEQ Z$ ;NO DATA TRANSFER ERROR
7272
7273 035334 052737 000010 003052 1$: BIS #E.UDAT,E.CONT ;SET UNEXPECTED DATA TYPE ERROR
7274 035342 004737 037334 JSR PC,R,CONT ;REPORT ERROR
7275 035346 000137 036306 JMP I,RTRN ;RESTORE REGISTERS
7276
7277 035352 013704 003002 2$: MOV T,CS2,R4 ;SAVE CS2 FOR REGISTER NUMBER
7278 035356 042704 177770 BIC #1C<DRVMSK>,R4 ;STRIP OFF JUNK
7279 035362 105037 003100 CLR W,TIME ;CLEAR WATCH DOG TIMER
7280 035366 005037 003114 CLR W,DRV ;RESET TIMER VALUE
7281 035372 013705 003112 MOV PBLKT,R5 ;STORE PARAMETER BLOCK ADDRESS IN R5
7282
7283 ;
7284 ; CLEAR DRIVE POSITIONING AND DRIVE POSITIONED FOR DATA TRANSFER
; IN PROGRAM DEVICE STATUS REGISTER
7285 035376 042765 000006 000014 BIC #DRVPOS!DRVPOT,P,PRST(R5)
7286
7287 035404 000137 034500 JMP I,ERRC ;GO REPORT ERROR
7288
7289 035410 032737 040000 003000 5$: BIT #DI,T,CSI ;CHECK IF ANY DRIVE ATTENTION
7290 035416 001002 BNE 6$ ;YES, PROCESS INTERRUPT
7291 035420 000137 036306 JMP I,RTRN ;RESTORE REGISTERS
7292
7293 035424 016237 000016 003014 6$: MOV RKASOF(R2),T,ASOF ;STORE ATTENTION SUMMARY
7294 035432 105737 003015 TSTB T,ASOF+1 ;CHECK IF ANY ATTENTIONS SET

```



```

7351 .SBTTL #ATTENTION ERROR HANDLER
7352
7353 035740 042765 000004 000014 I.AERR: BIC #DRVPDT,P.PRST(R5) ;RESET POSITIONING IN PROGRESS BECAUSE
7354 ;OF DATA TRANSFER
7355 035746 105037 003100 CLR W.TIME ;CLEAR TIMING FOR THIS DRIVE
7356 035752 005037 003114 CLR W.DRV ;RESET WATCH-DOG TIME
7357 035756 042765 177741 000016 BIC #177741,P.CS1(R5) ;KEEP COMMAND ISSUED
7358 035764 042737 000036 003000 BIC #36,T.CS1 ;KEEP CURRENT CONTROLLER STATUS
7359 035772 053765 003000 000016 BIS T.CS1,P.CS1(R5) ;MAKE GOOD MESSAGE
7360 036000 013765 003002 000020 MOV T.CS2,P.CS2(R5) ;STORE CONTROLLER REGISTERS
7361 036006 013765 003004 000022 MOV T.WCR,P.WCR(R5)
7362 036014 013765 003006 000024 MOV T.BA,P.BAR(R5)
7363 036022 013765 003010 000026 MOV T.DA,P.DTS(R5)
7364 036030 013765 003012 000030 MOV T.DC,P.DCYL(R5)
7365 036036 013765 003014 000032 MOV T.ASOF,P.ASOF(R5)
7366 036044 013765 003016 000034 MOV T.ER,P.ER(R5)
7367 036052 013765 003020 000036 MOV T.DS,P.DS(R5)
7368 036060 004037 037062 JSR RD,I.STAT ;GATHER DRIVE STATUS
7369 036064 036306 I.RTRN ;ERROR RETURN
7370 036066 112737 000005 003000 MOVB #DR.CLR,T.CS1 ;LOAD COMMAND
7371 036074 004037 036406 JSR RD,I.ISSU ;CLEAR DRIVE ERRORS
7372 036100 036306 I.RTRN ;ERROR RETURN
7373 036102 133737 003101 003015 BITB INTMSK,T.ASOF+1 ;CHECK IF ATTENTION RESET
7374 036110 001407 BEQ 2% ;YES, FLAG DRIVE ERROR
7375 036112 052737 000020 003052 BIS #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
7376 036120 004737 037334 JSR PC,R.CONT ;REPORT ERROR
7377 036124 000137 036306 JMP I.RTRN ;RESTORE REGISTERS
7378
7379 036130 032765 000020 000014 2%: BIT #DRVHRD,P.PRST(R5) ;CHECK IF AWARD DRIVE ERROR
7380 036136 001017 BNE 10% ;YES, REPORT ERROR
7381 036140 032737 020000 003024 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS UNLOADING
7382 036146 001413 BEQ 10% ;NO, REPORT ERROR
7383 036150 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING DUE TO ERROR
7384 036156 113737 003101 003100 MOVB INTMSK,W.TIME ;SET TIMING ON THIS DRIVE
7385 036164 013737 003064 003114 MOV W.BSEC,W.DRV ;LOAD 8 SECONDS FOR CYCLE UP TIME
7386 036172 000137 036306 JMP I.RTRN ;RESTORE REGISTERS
7387
7388 036176 004737 037310 10%: JSR PC,R.ABNL ;REPORT ERROR
7389 036202 000137 036306 JMP I.RTRN ;RESTORE REGISTERS
7390
7391 .SBTTL #ERROR CAUSING DRIVE TO UNLOAD
7392
7393 036206 052765 020000 000014 I.UNLD: BIS #E.UNLD,P.PRST(R5) ;CLEAR DRIVE UNLOADING BECAUSE OF ERROR
7394 036214 112737 000005 003000 MOVB #DR.CLR,T.CS1 ;LOAD IN DRIVE CLEAR
7395 036222 004037 036406 JSR RD,I.ISSU ;GO ISSUE DRIVE CLEAR
7396 036226 036306 I.RTRN ;ERROR RETURN
7397 036230 136437 003101 003015 BITB INTMSK(R4),T.ASOF+1 ;CHECK IF ATTENTION CLEARED
7398 036236 001406 BEQ 15% ;YES, CONTINUE
7399 036240 012737 000020 003052 MOV #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
7400 036246 004737 037334 JSR PC,R.CONT ;REPORT ERROR
7401 036252 000415 BR I.RTRN ;RESTORE REGISTERS
7402
7403 036254 032737 040000 003024 15%: BIT #S.DSC,T.MR2 ;CHECK IF DRIVE STAU CHANGE RESET
7404 036262 001403 BEQ 20% ;YES, CONTINUE
7405 036264 052765 000040 000014 BIS #DRVDSC,P.PRST(R5) ;SET DRIVE STAU CHANGE DID NOT CLEAR
7406 036272 105037 003100 20%: CLR W.TIME ;RESET TIMING ON THIS DRIVE

```

```

7407 036276 005037 003114          CLR      W.DRV      ;CLEAR TIME COUNT
7408 036302 004737 037310          JSR      PC,R.ABNL ;REPORT ERROR
7409
7410 036306 012600          I.RTRN: MOV      (SP)+,R0 ;RESTORE R0
7411 036310 012601          MOV      (SP)+,R1 ;RESTORE R1
7412 036312 012602          MOV      (SP)+,R2 ;RESTORE R2
7413 036314 012603          MOV      (SP)+,R3 ;RESTORE R3
7414 036316 012604          MOV      (SP)+,R4 ;RESTORE R4
7415 036320 012605          MCV      (SP)+,R5 ;RESTORE R5
7416 036322 000002          RTI
7417

```

.SBTTL #CONTROLLER CLEAR ROUTINE

7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452

```

*****
*
* THIS ROUTINE WILL BE USED BY THE DRIVER TO CLEAR THE CONTROLLER
* AND CHECK IF THE CONTROLLER ERRORS ARE RESET. IF THE ERROR IS NOT
* CLEARED, THE ROUTINE AS SPECIFIED IN A.CONT WILL BE CALLED WITH
* E.CCLR SET IN E.CONT.
*
* REGISTER      USE
* -----      ---
*
* R2            ADDRESS OF RK06 REGISTERS
* RS            ADDRESS OF PARAMETER BLOCK
*
*CALL JSR      RO,I.CCLR
*           <ADDRESS OF ERROR RETURN>
*           RETURN
*****
I.CCLR: MOV    #CCLR,RKCS1(R2) ; CLEAR CONTROLLER
        MOV    RKCS1(R2),T.CS1 ; STORE COMMAND AND STATUS REGISTER 1
        BIT    #CERR,T.CS1    ; CHECK IF CONTROLLER CLEAR DID
        ;      CLEAR ERROR
        BEQ    SS              ; YES, RETURN TO DRIVER PROCESSING
        BIS    #E.CCLR,E.CONT ; SET CLEAR CONTROLLER DID NOT CLEAR ERROR
        JSR    PC,R.CONT      ; REPORT CONTROLLER ERROR
        MOV    (R0),R0        ; SET UP ERROR RETURN
        RTS    R0             ; RETURN
SS:     MOV    #IE,RKCS1(R2)  ; SET INTERRUPT ENABLE
        MOVB  #-1,I.ISRL     ; SET INTERRUPT ENABLE ISSUED
        TST   (R0)+          ; ADJUST FOR NORMAL RETURN
        RTS    R0             ; RETURN

```

7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508

.SBTTL *COMMAND ISSUED BY DRIVER SERVICE ROUTINE

```

*****
*
* THIS ROUTINE WILL ISSUE THE COMMAND AS SPECIFIED IN T.CS1
* AND CHECK IF A CONTROLLER ERROR OCCURRED. IF A CONTROLLER
* ERROR OCCURRED, E.CERR WILL BE SET IN E.CONT AND
* CONTROL WILL BE TURN OVER TO THE ROUTINE SPECIFIED BY THE
* ADDRESS IN A.CONT.
*
* REGISTER      USE
* -----      ---
*
* R2            ADDRESS OF RK06 REGISTERS
* R5            ADDRESS OF PARAMETER BLOCK
*
*CALL JSR      RO,I.ISSU
*      <ADDRESS OF ERROR RETURN>
*      RETURN
*
* ROUTINES USED:
* -----
*
*           I.CCLR
*           I.STOR
*
*****

```

```

I.ISSU: MOV      T.CS1,-(SP)      ;STORE COMMAND ISSUED
        CLR      T.CS2          ;CLEAR TEMPORARY CS2
        MOVB     P.DRVN(R5),T.CS2 ;LOAD IN DRIVE NUMBER
        MOV      T.CS2,RKCS2(R2) ;LOAD DRIVE NUMBER FOR COMMAND
        MOVB     P.CS1H(R5),T.CS1+1 ;STORE BITS 8-15 OF CS1
        BICB     #+C(B.CDT!B.CFMT),T.CS1+1 ;CLEAR ALL BITS EXCEPT
        ;          FORMAT AND DRIVE TYPE
        MOV      T.CS1,RKCS1(R2) ;ISSUE COMMAND
15:     TSTB     RKCS1(R2)       ;WAIT FOR READY
        BPL      15$
        JSR      PC,I.STOR      ;GO STORE REGISTERS
        BIT      #CERR,T.CS1    ;CHECK IF CONTROLLER ERROR OCCURED
        BEQ      5$            ;NO RETURN
        BIT      #MDS,T.CS2     ;CHECK IF MULTIPLE DRIVE SELECT
        BEQ      2$            ;NO CHECK FOR OTHER CONTROLLER ERRORS
        BIS      #E.MDS,E.CONT  ;SET MULTIPLE DRIVE SELECT FLAG
        JSR      PC,R.CONT      ;REPORT CONTROLLER ERROR
        BR       10$           ;RETURN

;CHECK IF ANY CONTROLLER ERROR IS SET
2$:     BIT      #CTO!SPAR,T.CS1
        BNE     7$
        BIT      #LFE!PGE!NEM!NED!UPE!WCE!DLT,T.CS2
        BNE     7$
        BIT      #ILC!DTYE!FMTE!ECH!BSE!HVRC!COE!DTE!OPI!DCK,T.ER
        BNE     7$
        CMPB     #DR.CLR,(SP)   ;CHECK IF CLEAR DRIVE

```

7509	036556	001003				BNE	3\$;NO, DO NOT SET DRIVE HARD ERROR
7510	036560	052765	000020	000014		BIS	#DRVHRO,P.PRST(R5)	;SET HARD DRIVE ERROR
7511	036566	004037	036324		3\$:	JSR	RO,I.CCLR	;GO ISSUE A CONTROLLER CLEAR
7512	036572	036622				IOS		;ERROR RETURN
7513	036574	012762	000100	000000	5\$:	MOV	#IE,RKCS1(R2)	;SET INTERRUPT ENABLE
7514	036602	005726				TST	(SP)+	;ADJUST STACK
7515	036604	005720				TST	(RO)+	;ADJUST RO FOR NORMAL RETURN
7516	036606	000200				RTS	RO	;RETURN
7517								
7518	036610	052737	001000	003052	7\$:	BIS	#E.CERR,E.CONT	;SET CONTROLLER ERROR DURING
7519								;DRIVER SERVICING
7520	036616	004737	037334			JSR	PC,R.CONT	;REPORT ERROR
7521	036622	005726			10\$:	TST	(SP)+	;ADJUST STACK
7522	036624	011000				MOV	(RO),RO	;ADJUST RO FOR ERROR RETURN
7523	036626	000200				RTS	RO	;RETURN

.SBTTL #STORE RK611 UNIBUS REGISTERS

7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555

036630 016237 000000 003000
036636 016237 000010 003002
036644 016237 000002 003004
036652 016237 000004 003006
036660 016237 000006 003010
036666 016237 000012 003020
036674 016237 000014 003016
036702 016237 000016 003014
036710 016237 000020 003012
036716 016237 000026 003022
036724 016237 000034 003024
036732 016237 000036 003026
036740 016237 000030 003030
036746 016237 000032 003032
036754 000207

```
*****
*
* THIS SUBROUTINE IS CALLED BY THE RK06 DRIVER TO STORE ALL
* RK611 REGISTER IN TEMPORARY LOCATIONS.
*
*CALL JSR PC,I.STOR
* RETURN
*
* REGISTER USE
* -----
*
* R2 ADDRESS OF RK611 REGISTERS
*
*****
```

```
I.STOR: MOV RKCS1(R2),T.CS1 ;STORE ALL CONTROLLER REGISTERS
MOV RKCS2(R2),T.CS2 ; EXCEPT DATA BUFFER
MOV RKWC(R2),T.WCR
MOV RKBA(R2),T.BA
MOV RKDA(R2),T.DA
MOV RKDS(R2),T.DS
MOV RKER(R2),T.ER
MOV RKASOF(R2),T.ASOF
MOV RKDCYL(R2),T.DC
MOV RKMR1(R2),T.MR1
MOV RKMR2(R2),T.MR2
MOV RKMR3(R2),T.MR3
MOV RKECPS(R2),T.POS
MOV RKECPT(R2),T.PAT
RTS PC ;RETURN
```

7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600

.SBTTL *STORE CONTROLLER STATUS

```

*****
*
* THIS SUBROUTINE IS CALLED BY THE RK06 DRIVER AT PRIORITY 7.
* THE FOLLOWING REGISTERS WILL BE STORED:
*
*   COMMAND AND STATUS REGISTER 2
*   WORD COUNT REGISTER
*   BUS ADDRESS REGISTER
*   DESIRED TRACK AND SECTOR
*   STATUS REGISTER
*   ERROR REGISTER
*   ATTENTION SUMMARY/OFFSET REGISTER
*   CYLINDER ADDRESS REGISTER
*
*CALL JSR PC,I.CSTS
*RETURN
*
* THIS ROUTINE ASSUMES THE FOLLOWING REGISTERS CONTAIN:
*
*   REGISTER          CONTENTS
*   -----          -
*
*   R2                RK06 BASE ADDRESS
*   R5                ADDRESS OF PARAMETER BLOCK
*
*****

```

```

036756 042765 177741 000016 I.CSTS: BIC #177741,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FUNCTION
;OF LAST COMMAND ISSUED
036764 042737 000036 003000 BIC #36,T.CS1 ;CLEAR FUNCTION OF CS1 STATUS
036772 053765 003000 000016 BIS T.CS1,P.CS1(R5) ;GENERATE CS1 STATUS INFORMATION
037000 016265 000010 000020 I.CST1: MOV RKCS2(R2),P.CS2(R5) ;STORE COMMAND AND STATUS REGISTER 2
037006 016265 000002 000022 MOV RKWC(R2),P.WCR(R5) ;STORE WORD COUNT REGISTER
037014 016265 000004 000024 MOV RKBA(R2),P.BAR(R5) ;STORE BUS ADDRESS REGISTER
037022 016265 000006 000026 MOV RKDA(R2),P.DTS(R5) ;STORE DESIRED TRACK AND SECTOR
037030 016265 000012 000036 MOV RKDS(R2),P.DS(R5) ;STORE DRIVE STATUS REGISTER
037036 016265 000014 000034 MOV RKER(R2),P.ER(R5) ;STORE ERROR REGISTER
037044 016265 000016 000032 MOV RKASOF(R2),P.ASOF(R5) ;STORE ATTENTION SUMMARY AND
;OFFSET
037052 016265 000020 000030 MOV RKDCYL(R2),P.DCYL(R5) ;STORE CYLINDER ADDRESS
037060 000207 PC ;RETURN

```


.SBTTL #GATHER DRIVE STATUS

```

*****
*
* THIS SUBROUTINE WILL BE USED TO GATHER DRIVE STATUS
* BYTE 01, 10, AND 11. IT IS ASSUMED THAT THE DRIVE
* HAS PREVIOUSLY BEEN SEIZED. IT RUNS AT PRIORITY 7.
*
*CALL JSR RO,I.STAT
*      <ADDRESS OF ERROR RETURN>
*      RETURN
*
* THIS ROUTINE ASSUMES THE FOLLOWING REGISTERS CONTAIN:
*
*      REGISTER          CONTENTS
*      -----          -
*
*      R2                RK06 BASE ADDRESS
*      R5                ADDRESS OF PARAMETER BLOCK
*
* ROUTINES USED:
*      I.ISSU
*****

```

7627	037062	012762	000001	000026	I.STAT: MOV	#1,RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
7628							: FOR STATUS BYTE 01
7629	037070	112737	000001	003000	MOVB	#DR.SEL,T.CS1	: LOAD COMMAND
7630	037076	004037	036406		JSR	RO,I.ISSU	: GET STATUS BYTES 01
7631	037102	037272			3\$: ERROR RETURN
7632	037104	013765	003024	000044	MOV	T.MR2,P.A01(R5)	: STORE STATUS BYTE 01 MESS A
7633	037112	013765	003026	000046	MOV	T.MR3,P.B01(R5)	: STORE STATUS BYTE 01 MESS B
7634	037120	012762	000002	000026	MOV	#2,RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
7635							: FOR STATUS BYTE 10
7636	037126	112737	000001	003000	MOVB	#DR.SEL,T.CS1	: LOAD COMMAND
7637	037134	004037	036406		JSR	RO,I.ISSU	: GET STATUS BYTES 10
7638	037140	037272			3\$: ERROR RETURN
7639	037142	013765	003024	000050	MOV	T.MR2,P.A10(R5)	: STORE STATUS BYTE 10 MESS A
7640	037150	013765	003026	000052	MOV	T.MR3,P.B10(R5)	: STORE STATUS BYTE 10 MESS B
7641	037156	012762	000003	000026	MOV	#3,RKMR1(R2)	: LOAD MAINTENANCE REGISTER
7642							: FOR STATUS BYTE 11
7643	037164	112737	000001	003000	MOVB	#DR.SEL,T.CS1	: LOAD COMMAND
7644	037172	004037	036406		JSR	RO,I.ISSU	: GET STATUS BYTES 11
7645	037176	037272			3\$: ERROR RETURN
7646	037200	013765	003024	000054	MOV	T.MR2,P.A11(R5)	: STORE STATUS BYTE 11 MESS A
7647	037206	013765	003026	000056	MOV	T.MR3,P.B11(R5)	: STORE STATUS BYTE 11 MESS B
7648	037214	005062	000026		CLR	RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
7649							: FOR STATUS BYTE 00
7650	037220	112737	000001	003000	MOVB	#DR.SEL,T.CS1	: LOAD COMMAND
7651	037226	004037	036406		JSR	RO,I.ISSU	: GET STATUS BYTES 00
7652	037232	037272			3\$: ERROR RETURN
7653	037234	013765	003024	000040	MOV	T.MR2,P.A00(R5)	: STORE STATUS BYTE 00 MESS A
7654	037242	013765	003026	000042	MOV	T.MR3,P.B00(R5)	: STORE STATUS BYTE 00 MESS B
7655	037250	032737	001000	003026	BIT	#S.PAR,T.MR3	: CHECK IF BAD PARITY DETECTED BY DRIVE
7656	037256	001407			BEQ	5\$: NO, RETURN NORMALLY

7657	037260	052737	002000	003052		BIS	#E.DPAR,E.CONT	; INDICATE BAD PARITY DETECTED BY DRIVE
7658	037266	004737	037334			JSR	PC,R.CONT	; REPORT ERROR
7659	037272	011000			3\$:	MOV	(R0),R0	; LOAD R0 FOR ERROR RETURN
7660	037274	000200				RTS	R0	; RETURN
7661								
7662	037276	052765	001000	000014	5\$:	BIS	#PGSVAL,P.PRST(R5)	; SET PARAMETER BLOCK STATUS VALID
7663	037304	005720				TST	(R0)+	; ADJUST R0 FOR NORMAL RETURN
7664	037306	000200				RTS	R0	; RETURN
7665								

.SBTTL *COMMON DRIVER RETURNS

7666							
7667							
7668	037310	105037	003101	R.ABNL:	CLRB	INTMSK	;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
7669	037314	004777	143526		JSR	PC,QA.ABNL	;INDICATE ABNORMAL RETURN
7670	037320	000207			RTS	PC	;RETURN
7671							
7672	037322	105037	003101	R.NORM:	CLRB	INTMSK	;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
7673	037326	004777	143512		JSR	PC,QA.NORM	;INDICATE NORMAL RETURN
7674	037332	000207			RTS	PC	;RETURN
7675							
7676	037334	105037	003101	R.CONT:	CLRB	INTMSK	;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
7677	037340	105037	003100		CLRB	W.TIME	;RESET WATCH DOG TIMING ON THIS DRIVE
7678	037344	005037	003114		CLR	W.DRV	;CLEAR TIMING COUNT FOR THIS DRIVE
7679	037350	004777	143474		JSR	PC,QA.CONT	;INDICATE CONTROLLER ERROR RETURN
7680	037354	000207			RTS	PC	;RETURN

7681
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736

037356 010546
037360 010446
037362 010346
037364 010246
037366 010146
037370 010046
037372 013746 177776
037376 013737 003042 177776
037404 017605 000016
037410 062766 000002 000016
037416 016504 000000
037422 042704 177770
037426 010537 003112
037432 116437 003102 003101
037440 116437 003102 003100
037446 013737 003062 003114
037454 013702 003036

.SBTTL *COMMAND INITIATOR

```
*****
*
* THIS SUBROUTINE WILL INITIATE ALL COMMANDS AS SPECIFIED
* BY THE COMMAND FIELD OF THE PARAMETER BLOCK. THE FOLLOWING
* SPECIAL COMMAND ARE ALSO EXECUTED:
*
*     RELEASE
*     CONTROLLER CLEAR
*     SUBSYSTEM CLEAR
*     READ ALL DRIVE STATUS
*     READ SPECIFIED HEADER
*
* THE ABOVE COMMANDS ARE TRANSLATED INTO A SEQUENCE OF COMMANDS
*
*CALL JSR    PC.C.INIT
*      <ADDRESS OF PARAMETER BLOCK>
*      RETURN
*
* FOR THE SEQUENTIAL OPERATIONS, THE DRIVER WILL LOAD THE
* LOCATIONS, PBLKT AND INTMSK.
*
* ROUTINES USED:
*     W.WTCH
*     I.CSTS
*     I.STAT
*     I.CCLR
*
*****
```

```
C.INIT: MOV    R5, -(SP)      ; STORE R5 ON STACK
        MOV    R4, -(SP)      ; STORE R4 ON STACK
        MOV    R3, -(SP)      ; STORE R3 ON STACK
        MOV    R2, -(SP)      ; STORE R2 ON STACK
        MOV    R1, -(SP)      ; STORE R1 ON STACK
        MOV    R0, -(SP)      ; STORE R0 ON STACK
        MOV    PS, -(SP)      ; STORE PSW ON STACK
        MOV    RKPRI, PS      ; LOCK OUT RK06 INTERRUPTS
        MOV    @16(SP), R5     ; STORE PARAMETER BLOCK ADDRESS
        ADD    #2, 16(SP)     ; ADJUST RETURN
        MOV    P.DRVN(R5), R4  ; STORE DRIVE NUMBER
        BIC    #1C(DRVMSK), R4 ; MASK OUT JUNK
        MOV    R5, PBLKT      ; LOAD PARAMETER BLOCK TABLE
        MOVB   I.DRV(R4), INTMSK ; LOAD INTERRUPT MASK
        MOVB   I.DRV(R4), W.TIME ; SET WATCH-DOG TIMER FLAG
        MOV    W.SEC, W.DAV    ; LOAD WATCH-DOG TIME

        MOV    RKBAS, R2      ; LOAD R2 WITH RK06 ADDRESS BASE

        ; RESET ALL BITS IN PROGRAM DEVICE STATUS REGISTER EXCEPT
        ; DRIVE IN USE
        ; WRITE FOR WRITE CHECK
        ; NO CHECK
        ; DROP DRIVE FROM TEST SEQUENCE
        ; INHIBIT BUS ADDRESS INCREMENT
```

```

7737 037460 042765 075176 000014      BIC      #1C(DRVUSE!W.WCK!NOCHK!DRPDRV!DTBAI),P.PRST(R5)
7738
7739 037466 010500      MOV      R5,R0      ;STORE PARAMETER BLOCK ADDRESS
7740 037470 062700 000016      ADD      #P.CS1,R0   ;CALCULATE FIRST LOCATION TO BE CLEARED
7741 037474 010501      MOV      R5,R1      ;STORE PARAMETER BLOCK ADDRESS
7742 037476 062701 000062      ADD      #P.EPAT,R1  ;CALCULATE LAST LOCATION TO BE CLEARED
7743
7744 037502 005020      CLR      (R0)+      ;CLEAR RETURN PARAMETER
7745 037504 020001      CMP      R0,R1      ;CHECK IF FINISHED
7746 037506 101775      BLOS    1$         ;NO, CLEAR NEXT RETURN PARAMETER
7747 037510 105037 003074      CLRB    I.ISRL     ;CLEAR RELEASE OR INTERRUPT ISSUED
7748 037514 010465 000020      MOV      R4,P.CS2(R5) ;STORE DRIVE NUMBER
7749 037520 005062 000026      CLR      RKMR1(R2)  ;CLEAR RK06 MAINTENANCE REGISTER 1
7750 037524 132765 000040 000001      BITB    #BITS,P.CMND(R5) ;CHECK IF SPECIAL COMMAND
7751 037532 001402      BEQ     3$         ;NO, PROCESS
7752 037534 000137 040250      JMP     C.SPEC     ;JUMP TO SPECIAL COMMAND PROCESSOR
7753
7754 037540 122765 000107 000001 3$:      CMPB    #UNLOAD,P.CMND(R5) ;CHECK IF POSITIONING COMMAND
7755                                     ;START SPINDLE
7756                                     ;RECALIBRATE
7757                                     ;OFFSET
7758                                     ;SEEK
7759                                     ;UNLOAD
7760
7761 037546 101174      BHI     25$        ;NO, DRIVE COMMAND
7762                                     ;SELECT DRIVE
7763                                     ;PACK ACKNOWLEDGE
7764                                     ;CLEAR
7765
7766 037550 122765 000117 000001      CMPB    #SEEK,P.CMND(R5) ;CHECK IF DATA TRANSFER
7767 037556 103540      BLO     20$        ;YES, DATA TRANSFER COMMAND
7768                                     ;READ DATA
7769                                     ;WRITE DATA
7770                                     ;READ HEADER
7771                                     ;WRITE HEADER
7772                                     ;WRITE CHECK
7773 037560 016562 000020 000010      MOV      P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
7774 037566 052765 000002 000014      BIS      #DRVPOS,P.PRST(R5) ;SET DRIVE POSITIONING
7775 037574 005037 003054      CLR      0.WAIT     ;CLEAR WAIT FOR COMMAND
7776 037600 122765 000117 000001      CMPB    #SEEK,P.CMND(R5) ;CHECK IF SEEK
7777 037606 001007      BNE     5$         ;NO, CHECK FOR OFFSET
7778 037610 016562 000002 000020      MOV      P.CYL(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
7779 037616 016562 000004 000006      MOV      P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
7780 037624 000431      BR      8$         ;GO ISSUE COMMAND
7781
7782 037626 122765 000115 000001 5$:      CMPB    #OFFSET,P.CMND(R5) ;CHECK IF OFFSET
7783 037634 001007      BNE     6$         ;NO, CHECK FOR UNLOAD
7784 037636 116565 000006 000032      MOVB    P.OFST(R5),P.ASOF(R5) ;STORE OFFSET
7785 037644 016562 000032 000016      MOV      P.ASOF(R5),RKASOF(R2) ;LOAD OFFSET REGISTER
7786 037652 000416      BR      8$         ;GO ISSUE COMMAND
7787
7788 037654 122765 000111 000001 6$:      CMPB    #SRTSPL,P.CMND(R5) ;CHECK IF START SPINDLE
7789 037662 001003      BNE     7$         ;NO, CHECK IF RECAL
7790 037664 013737 003066 003114      MOV      W.MIN,W.DRV ;LOAD WATCH DOG TIME FOR 1 MINUTE
7791 037672 122765 000113 000001 7$:      CMPB    #RECAL,P.CMND(R5) ;CHECK IF RECAL
7792 037700 001003      BNE     8$         ;NO, CONTINUE
  
```

*COMMAND INITIATOR

```

7793 037702 013737 003064 003114      MOV      W.BSEC,W.DRV      ;LOAD RECAL TIME FOR 8 SECONDS
7794 037710 116565 000007 000017 85:      MOVVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
7795 037716 042765 165777 000016      BIC     #↑C(CFMT!CDT),P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT
7796                                     ; AND DRIVE TYPE
7797 037724 116565 000001 000016      MOVVB   P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
7798 037732 042765 000200 000014      BIC     #W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
7799 037740 032765 000400 000014      BIT     #NOCHK,P.PRST(R5) ;CHECK IN NO CHECK MODE
7800 037746 001533                                     BEQ     30$ ;NO, SKIP CLEAR OF INTERRUPT ENABLE
7801 037750 042765 000100 000016      BIC     #IE,P.CS1(R5) ;CLEAR INTERRUPT ENABLE
7802 037756 016562 000016 000000      MOV     P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
7803 037764 004737 033726 000000 10$:     JSR     PC.W.WTCH ;CALL WATCH DOG TIMER
7804 037770 016237 000000 003000      MOV     RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS REGISTER 1
7805 037776 032737 000200 003000      BIT     #RDY,T.CS1 ;WAIT FOR READY
7806 040004 001767                                     BEQ     10$
7807 040006 032737 100000 003000      BIT     #CERR,T.CS1 ;CHECK FOR ERROR
7808 040014 001011                                     BNE     15$ ;YES, GIVE NORMAL RETURN
7809 040016 004737 033726 000000 11$:     JSR     PC.W.WTCH ;CALL WATCH DOG TIMER
7810 040022 016237 000016 003014      MOV     RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
7811 040030 133737 003101 003015      BITB   INTMSK,T.ASOF+1 ;CHECK IF INTERRUPT HAS OCCURRED
7812 040036 001767                                     BEQ     11$ ;WAIT FOR DRIVE INTERRUPT
7813 040040 105037 003100 000000 15$:     CLRB   W.TIME ;RESET TIMING ON THIS DRIVE
7814 040044 005037 003114                                     CLR     W.DRV ;CLEAR DRIVE TIMING COUNT
7815 040050 004737 037322                                     JSR     PC.R.NORM ;INDICATE COMMAND IS FINISHED
7816 040054 000137 041230                                     JMP     C.ATRN ;RESTORE REGISTERS
7817
7818 040060 016562 000010 000004 20$:     MOV     P.BALO(R5),RKBA(R2) ;LOAD BUS ADDRESS REGISTER
7819 040066 016562 000012 000002      MOV     P.WC(R5),RKWC(R2) ;LOAD WORD COUNT REGISTER
7820 040074 016562 000002 000020      MOV     P.CYL(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
7821 040102 016562 000004 000006      MOV     P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK NUMBER
7822 040110 122765 000131 000001      CMPB   #RTCHK,P.CMND(R5) ;CHECK IF WRITE CHECK COMMAND
7823 040116 001010                                     BNE     25$ ;NO, GO ISSUE THE COMMAND
7824 040120 032765 000200 000014      BIT     #W.WCK,P.PRST(R5) ;CHECK IF WRITE COMMAND SHOULD BE ISSUED
7825 040126 001404                                     BEQ     25$ ;NO, GO ISSUE THE COMMAND
7826 040130 012765 000123 000016      MOV     #RDATA,P.CS1(R5) ;ISSUE WRITE COMMAND
7827 040136 000406                                     BR      26$ ;GO ISSUE COMMAND
7828
7829 040140 116565 000001 000016 25$:     MOVVB   P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
7830 040146 042765 000200 000014      BIC     #W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
7831 040154 116565 000007 000017 26$:     MOVVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
7832 040162 142765 177750 000017      BICB   #↑C(B.CFMT!B.CDT!B.BA16!B.BA17),P.CS1+1(R5) ;CLEAR ALL BITS EXCEPT
7833                                     ; FORMAT, DRIVE TYPE, AND BUS ADDRESS
7834                                     ; BITS 16-17
7835 040170 010537 003054                                     MOV     R5,0.WAIT ;LOAD WAITING FOR COMMAND
7836 040174 032765 100000 000014      BIT     #CIBAI,P.PRST(R5) ;CHECK IF INHIBIT BUS ADDRESS INCREMENT
7837 040202 001403                                     BEQ     27$ ;NO, LOAD CS2
7838 040204 052765 000020 000020      BIS     #BAI,P.CS2(R5) ;SET INHIBIT BUS ADDRESS INCREMENT
7839 040212 016562 000020 000010 27$:     MOV     P.CS2(R5),RKCS2(R2) ;LOAD CS2
7840 040220 032765 000400 000014      BIT     #NOCHK,P.PRST(R5) ;CHECK IN NO CHECK MODE
7841 040226 001403                                     BEQ     30$ ;NO, SKIP CLEAR OF INTERRUPT ENABLE
7842 040230 042765 000100 000016      BIC     #IE,P.CS1(R5) ;CLEAR INTERRUPT ENABLE
7843 040236 016562 000016 000000 30$:     MOV     P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
7844 040244 000137 041230                                     JMP     C.ATRN ;RESTORE REGISTERS
7845
7846                                     .SBTTL  *SPECIAL COMMAND PROCESSING
7847
7848 040250 122765 000141 000001  C.SPEC: CMPB   #RSTAT,P.CMND(R5) ;CHECK IF READ DRIVE STATUS

```

7849	040256	001132			BNE	10\$: NO, PROCESS OTHER COMMANDS
7850	040260	016562	000020	000010	MOV	P.CS2(R5),RKCS2(R2)	: LOAD CS2 FOR COMMAND
7851	040266	116565	000007	000017	MOVB	P.CS1H(R5),P.CS1+1(R5)	: STORE BITS 8-15 OF CS1
7852	040274	042765	165777	000016	BIC	#1C<CFMT!COT>,P.CS1(R5)	: CLEAR ALL BITS EXCEPT FORMAT AND DRIVE TYPE
7853							
7854	040302	112765	000001	000016	MOVB	#DR.SEL,P.CS1(R5)	: STORE COMMAND
7855	040310	016562	000016	000000	MOV	P.CS1(R5),RKCS1(R2)	: ISSUE COMMAND
7856	040316	004737	033726		JSR	PC.W.WTCH	: CALL WATCH-DOG TIMER
7857	040322	016265	000000	000016	MOV	RKCS1(R2),P.CS1(R5)	: STORE COMMAND AND STATUS REG. 1
7858	040330	032765	000200	000016	BIT	#RDY,P.CS1(R5)	: WAIT FOR READY
7859	040336	001767			BEQ	2\$	
7860	040340	004737	037000		JSR	PC.I.CST1	: STORE CONTROLLER REGISTERS
7861	040344	016265	000034	000040	MOV	RKMR2(R2),P.A00(R5)	: STORE STATUS BYTE 00 MESSAGE A
7862	040352	016265	000036	000042	MOV	RKMR3(R2),P.B00(R5)	: STORE STATUS BYTE 00 MESSAGE B
7863	040360	032765	100000	000016	BIT	#CERR,P.CS1(R5)	: CHECK IF CONTROLLER ERROR
7864	040366	001436			BEQ	6\$: NO, GATHER DRIVE STATUS
7865	040370	105037	003100		CLRB	W.TIME	: RESET WATCH DOG TIMING ON THIS DRIVE
7866	040374	005037	003114		CLR	W.DRV	: CLEAR WATCH DOG COUNT
7867	040400	032765	000400	000014	BIT	#NOCHK,P.PRST(R5)	: CHECK IF NO CHECK MODE
7868	040406	001043			BNE	8\$: YES, INDICATE NORMAL RETURN
7869	040410	032765	001000	000020	BIT	#MDS,P.CS2(R5)	: CHECK IF MULTIPLE DRIVE SELECT
7870	040416	001043			BNE	9\$: YES, INDICATE CONTROLLER ERROR
7871	040420	004037	036324		JSR	RO,I.CCLR	: CLEAR ERROR
7872	040424	041230			C.RTRN		: ERROR RETURN
7873	040426	032765	010400	000020	BIT	#NED!LFE,P.CS2(R5)	: CHECK IF NON-EXISTENT DRIVE OR UNIT FIELD ERROR
7874	040434	001007			BNE	5\$: REPORT ERROR
7875	040436	032765	000001	000036	BIT	#ORA,P.DS(R5)	: CHECK IF DRIVE AVAILIABLE
7876	040444	001003			BNE	5\$: YES, REPORT ERROR
7877	040446	052765	010000	000014	BIS	#DRVSZD,P.PRST(R5)	: INDICATE DRIVE IS SEIZED BY OTHER PORT
7878	040454	004737	037310		JSR	PC.R.ABNL	: INDICATE ABNORMAL RETURN
7879	040460	000137	041230		JMP	C.RTRN	: RESTORE REGISTERS
7880							
7881	040464	004037	037062		JSR	RO,I.STAT	: GATHER DRIVE STATUS
7882	040470	041230			C.RTRN		: ERROR RETURN
7883	040472	105037	003100		CLRB	W.TIME	: STOP WATCH-DOG TIMING ON DRIVE
7884	040476	005037	003114		CLR	W.DRV	: RESET WATCH-DOG TIME
7885	040502	032765	000400	000014	BIT	#NOCHK,P.PRST(R5)	: CHECK IF NO CHECK MODE
7886	040510	001402			BEQ	8\$: NO, REPORT ERROR
7887	040512	005062	000000		CLR	RKCS1(R2)	: CLEAR INTERRUPT ENABLE
7888	040516	004737	037322		JSR	PC.R.NORM	: REPORT COMMAND COMPLETE
7889	040522	000137	041230		JMP	C.RTRN	: RESTORE REGISTERS
7890							
7891	040526	052737	100000	003052	BIS	#E.MDS,E.CONT	: SET MULTIPLE DRIVE SELECT
7892	040534	004737	037334		JSR	PC.R.CONT	: INDICATE CONTROLLER ERROR
7893	040540	000137	041230		JMP	C.RTRN	
7894							
7895	040544	122765	000140	000001	10\$: CMPB	#RELEAS,P.CMND(R5)	: CHECK IF RELEASE COMMAND
7896	040552	001040			BNE	13\$: NO, CHECK IF READ ALL HEADERS
7897	040554	010537	003054		MOV	RS,0.WAIT	: STORE PARAMETER BLOCK ADDRESS IN WAIT FOR COMMAND
7898							
7899	040560	052765	000010	000020	BIS	#RLS,P.CS2(R5)	: SET RELEASE BIT
7900	040566	016562	000020	000010	MOV	P.CS2(R5),RKCS2(R2)	: LOAD CS2 FOR DESELECT
7901	040574	112737	000001	003074	MOVB	#I.ISRL	: SET FLAG FOR RELEASE COMMAND
7902	040602	116565	000007	000017	MOVB	P.CS1H(R5),P.CS1+1(R5)	: STORE BITS 8-15 OF CS1
7903	040610	042765	165777	000016	BIC	#1C<CFMT!COT>,P.CS1(R5)	: CLEAR ALL BITS EXCEPT FORMAT AND DRIVE TYPE
7904							

7905	040616	112765	000101	000016		MOV	#SELDRV,P.CS1(R5)	:STORE COMMAND
7906	040624	032765	000400	000014		BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE
7907	040632	001403				BEQ	11\$:NO,DO NOT RESET INTERRUPT ENABLE
7908	040634	042765	000100	000016		BIC	#IE,P.CS1(R5)	:RESET INTERRUPT ENABLE
7909	040642	016562	000016	000000	11\$:	MOV	P.CS1(R5),RKCS1(R2)	:ISSUE COMMAND
7910	040650	000137	041230			JMP	C.RTRN	:RESTORE REGISTERS
7911								
7912	040654	122765	000164	000001	13\$:	CMPB	#RDALHD,P.CMND(R5)	:CHECK IF READ ALL HEADERS
7913	040662	001053				BNE	30\$:NO,CHECK IF CONTROLLER CLEAR
7914	040664	010537	003054			MOV	RS,0.WAIT	:SET WAITING FOR COMMAND COMPLETION
7915	040670	016537	000010	003070		MOV	P.BALO(R5),HDR.A0	:LOAD HEADER ADDRESS
7916	040676	132765	000020	000007		BITB	#B.CFMT,P.CS1H(R5)	:CHECK IF 22 SECTOR FORMAT
7917	040704	001404				BEQ	14\$:YES,LOAD 22 IN HEADER COUNT
7918	040706	012737	000024	003072		MOV	#20.,HDR.CT	:LOAD 20 IN SECTOR COUNT
7919	040714	000403				BR	22\$:GO ISSUE READ HEADER COMMAND
7920								
7921	040716	012737	000026	003072	14\$:	MOV	#22.,HDR.CT	:LOAD 22 IN SECTOR COUNT
7922	040724	016562	000002	000020	22\$:	MOV	P.CYL(R5),RKDCYL(R2)	:LOAD CYLINDER ADDRESS
7923	040732	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2)	:LOAD TRACK NUMBER
7924	040740	016562	000020	000010		MOV	P.CS2(R5),RKCS2(R2)	:LOAD DRIVE NUMBER
7925	040746	116565	000007	000017		MOV	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1
7926	040754	042765	165777	000016		BIC	#1<CFMT!COT>,P.CS1(R5)	:CLEAR ALL BITS EXCEPT DRIVE TYPE
7927								:AND FORMAT
7928	040762	112765	030125	000016		MOV	#RDHEAD,P.CS1(R5)	:STORE READ HEADER COMMAND
7929	040770	032765	000400	000014		BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE
7930	040776	001027				BNE	34\$:YES,INDICATE ILLEGAL DRIVER COMMAND
7931	041000	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2)	:ISSUE READ HEADER
7932	041006	000137	041230			JMP	C.RTRN	:RESTORE REGISTERS
7933								
7934	041012	122765	000176	000001	30\$:	CMPB	#CONCLR,P.CMND(R5)	:CHECK IF CONTROLLER CLEAR
7935	041020	001012				BNE	32\$:NO,CHECK IF SUBSYSTEM CLEAR
7936	041022	004037	036324			JSR	RD,I.CCLR	:CLEAR CONTROLLER
7937	041026	041230				C.RTRN		:ERROR RETURN
7938	041030	032765	000400	000014		BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE
7939	041036	001472				BEQ	40\$:NO,INDICATE NORMAL RETURN
7940	041040	005062	000000			CLR	RKCS1(R2)	:RESET INTERRUPT ENABLE
7941	041044	000467				BR	40\$:INDICATE NORMAL RETURN
7942								
7943	041046	122765	000177	000001	32\$:	CMPB	#SUBCLR,P.CMND(R5)	:CHECK IF SUBSYSTEM CLEAR
7944	041054	001406				BEQ	36\$:YES,CLEAR SUBSYSTEM
7945	041056	052737	000100	003052	34\$:	BIS	#E.IILD,E.CONT	:SET ILLEGAL DRIVER COMMAND
7946	041064	004737	037334			JSR	PC,R.CONT	:REPORT ERROR
7947	041070	000457				BR	C.RTRN	:RESTORE REGISTERS
7948								
7949	041072	012762	000040	000010	36\$:	MOV	#SCLR,RKCS2(R2)	:ISSUE SUBSYSTEM CLEAR
7950	041100	016265	000000	000016		MOV	RKCS1(R2),P.CS1(R5)	:STORE COMMAND AND STATUS REGISTER 1
7951	041106	032765	100000	000016		BIT	#CERR,P.CS1(R5)	:CLEAR IF CONTROLLER ERROR RESET
7952	041114	001406				BEQ	37\$:NO,FINISH COMMAND
7953	041116	052737	000001	003052		BIS	#BIT0,E.CONT	:SET CLEAR SUBSYSTEM DID NOT CLEAR
7954								:CONTROLLER ERROR
7955	041124	004737	037334			JSR	PC,R.CONT	:REPORT ERROR
7956	041130	000437				BR	C.RTRN	:RESTORE REGISTERS
7957								
7958	041132	013746	003060		37\$:	MOV	W.MILI,-(SP)	:LOAD 16 MILLI-SECOND COUNT FOR ATTENTION
7959								:TO DISAPPEAR
7960	041136	016265	000000	000016	38\$:	MOV	RKCS1(R2),P.CS1(R5)	:STORE CS1


```

7961 041144 032765 040000 000016 BIT #01,P.CS1(R5) ;CHECK IF ATTENTIONS CLEARED
7962 041152 001411 BEQ 39$ ;YES FINISH COMMAND
7963 041154 005316 DEC (SP) ;DECREMENT 16 MILISECOND COUNT
7964 041156 001367 BNE 38$ ;CHECK DRIVE INTERRUPT AGAIN
7965 041160 005726 TST (SP)+ ;ADJUST STACK
7966 041162 052737 000040 003052 BIS #E.SCLR,E.CONT ;SET SUBSYSTEM CLEAR DID NOT CLEAR
7967 041170 004737 037334 JSR PC,R.CONT ;REPORT ERROR
7969 041174 000415 BR C.RTRN ;RESTORE REGISTER
7971 041176 005726 39$: TST (SP)+ ;ADJUST STACK
7972 041200 032765 000400 000014 BIT #NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
7973 041206 001010 BNE C.RTRN ;YES, RESTORE REGISTERS
7974 041210 112737 177777 003074 MOVB #-1,I.ISRL ;SET INTERRUPT ENABLE SET
7975 041216 012762 000100 000000 MOV #IE,RKCS1(R2) ;SET INTERRUPT ENABLE
7976 041224 004737 037322 40$: JSR PC,R.NORM ;INDICATE NORMAL TERMINATION
7977 041230 012637 177776 C.RTRN: MOV (SP)+,PS ;RESTORE PSW
7979 041234 012600 MOV (SP)+,R0 ;RESTORE R0
7980 041236 012601 MOV (SP)+,R1 ;RESTORE R1
7981 041240 012602 MOV (SP)+,R2 ;RESTORE R2
7982 041242 012603 MOV (SP)+,R3 ;RESTORE R3
7983 041244 012604 MOV (SP)+,R4 ;RESTORE R4
7984 041246 012605 MOV (SP)+,R5 ;RESTORE R5
7985 041250 000207 RTS PC ;RETURN
7986 .SBTTL OCTAL TO BINARY CONVERSION ROUTINE
7987
7988 *****
7989 *
7990 * THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
7991 * WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL
7992 * IT WILL GENERATE TWO BINARY WORDS PLACING THE LOW 16 BITS
7993 * ON THE STACK AND THE HIGH 16 BITS IN LOCATION SHIOCT.
7994 *
7995 *CALL
7996 * MOV <ADDRESS OF ASCII STRING>,-(SP)
7997 * JSR PC,OCTBIN
7998 * <ADDRESS OF ERROR RETURN>
7999 * RETURN
8000 *****
8001
8002
8003 041252 010046 OCTBIN: MOV R0,-(SP) ;SAVE R0
8004 041254 010146 MOV R1,-(SP) ;SAVE R1
8005 041256 010246 MOV R2,-(SP) ;SAVE R2
8006 041260 016600 000010 MOV 10(SP),R0 ;GET ADDRESS OF ASCII STRING
8007 041264 005001 CLR R1 ;CLEAR DATA WORDS
8008 041266 005002 CLR R2
8009 041270 112046 2$: MOVB (R0)+,-(SP) ;PICK THIS CHARACTER
8010 041272 001423 BEQ 3$ ;IF ZERO GET OUT
8011 041274 121627 000054 CMPB (SP),#', ;CHECK IF COMMA
8012 041300 001420 BEQ 3$ ;IF COMMA GET OUT
8013 041302 122716 000060 CMPB #'0,(SP) ;MAKE SURE THIS CHARACTER IS
8014 041306 003030 BGT 4$ ; AN OCTAL DIGIT
8015 041310 122716 000067 CMPB #'7,(SP)
8016 041314 002425 BLT 4$

```

E13

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 160
 OCTAL TO BINARY CONVERSION ROUTINE

SEQ 0159

```

8017 041316 006301      ASL    R1          ; #2
8018 041320 006102      ROL    R2
8019 041322 006301      ASL    R1          ; #4
8020 041324 006102      ROL    R2
8021 041326 006301      ASL    R1          ; #8
8022 041330 006102      ROL    R2
8023 041332 042716 177770 BIC    #1C7,(SP)   ;STRIP THE ASCII JUNK
8024 041336 062601      ADD    (SP)+,R1   ;ADD THIS DIGIT
8025 041340 000753      BR     2$         ;LOOP
8026 041342 005726 3$:    TST    (SP)+       ;CLEAN PARTIAL FROM STACK
8027 041344 010166 000010 MOV    R1,10(SP)  ;SAVE RESULT
8028 041350 010237 041404 MOV    R2,$HI OCT
8029 041354 012602      MOV    (SP)+,R2   ;RESTORE R2
8030 041356 012601      MOV    (SP)+,R1   ;RESTORE R1
8031 041360 012600      MOV    (SP)+,R0   ;RESTORE R0
8032 041362 062716 000002 ADD    #2,(SP)    ;ADJUST RETURN
8033 041366 000207      RTS    PC         ;RETURN
8034
8035 041370 005726 4$:    TST    (SP)+       ;CLEAN UP PARTIAL FROM STACK
8036 041372 012602      MOV    (SP)+,R2   ;RESTORE R2
8037 041374 012601      MOV    (SP)+,R1   ;RESTORE R1
8038 041376 012600      MOV    (SP)+,R0   ;RESTORE R0
8039 041400 013616      MOV    2(SP)+,(SP) ;PUT ADDRESS OF ERROR ROUTINE ON STACK
8040 041402 000207      RTS    PC         ;GO PROCESS ERROR
8041 041404 000000      .WORD 0          ;HIGH ORDER BITS GO HERE
8042
8043 .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
8044
8045 ;*****
8046 ;THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
8047 ;WITH A RANGE OF 0 TO 2(+33)-1.
8048 ;CALL:
8049 ;* JSR    PC,$RAND ;CALL THE ROUTINE
8050 ;* RETURN ;RETURN HERE THE RANDOM
8051 ;* ;NUMBER WILL BE IN
8052 ;* ;$HINUM,$LONUM
8053
8054 $RAND:
8055 MOV    R0,-(SP)   ;PUSH R0 ON STACK
8056 MOV    R1,-(SP)   ;PUSH R1 ON STACK
8057 MOV    R2,-(SP)   ;PUSH R2 ON STACK
8058 MOV    $LONUM,R0  ;SET R0 WITH LOW
8059 MOV    $HINUM,R1  ;SET R1 WITH HIGH
8060 MOV    #-7,R2     ;SET SHIFT COUNT
8061 1$: ASL    R0       ;SHIFT R0 LEFT AND
8062 ROL    R1         ;ROTATE CARRY INTO R1 AND
8063 INC    R2         ;CHECK FOR DONE
8064 BNE   1$         ;CONTINUE SHIFT LOOP
8065 ADD    $LONUM,R0  ;ADD NUMBER TO MAKE X 129
8066 ROL    R1         ;PROPOGATE CARRY
8067 ADD    $HINUM,R1  ;ADD NUMBER TO MAKE X 129
8068 ADD    #1057,R0   ;ADD LOW CONSTANT
8069 ROL    R1         ;PROPOGATE CARRY
8070 ADD    #47401,R1  ;ADD HIGH CONSTANT
8071 MOV    R0,$LONUM ;SAVE R0
8072 MOV    R1,$HINUM ;SAVE R1
      MOV    (SP)+,R2 ;POP STACK INTO R2

```

8073 041476 012601
8074 041500 012600
8075 041502 000207
8076 041504 176543
8077 041506 123456

MOV (SP)+,R1 ;; POP STACK INTO R1
MOV (SP)+,R0 ;; POP STACK INTO R0
RTS PC ;; RETURN
\$HINUM: .WORD 176543
\$LONUM: .WORD 123456
.SBTTL TYPE ROUTINE

8078
8079
8080
8081
8082
8083
8084
8085
8086
8087
8088
8089
8090
8091
8092
8093
8094

*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
*

8095 041510 105737 001157
8096 041514 100002
8097 041516 000000
8098 041520 000430
8099 041522 010046
8100 041524 017600 000002
8101 041530 122737 000001 001350
8102 041536 001011
8103 041540 132737 000100 001351
8104 041546 001405
8105 041550 010037 041560
8106 041554 004737 044120
8107 041560 000000
8108 041562 132737 000040 001351
8109 041570 001003
8110 041572 112046
8111 041574 001005
8112 041576 005726
8113 041600 012600
8114 041602 062716 000002
8115 041606 000002
8116 041610 122716 000011
8117 041614 001430
8118 041616 122716 000200
8119 041622 001006
8120 041624 005726
8121 041626 104401
8122 041630 001325
8123 041632 105037 041766
8124 041636 000755
8125 041640 004737 041722
8126 041644 123726 001156
8127 041650 001350
8128 041652 013746 001154

\$TYPE: TSTB \$TPFLG ;; IS THERE A TERMINAL?
BPL 1\$;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3\$;; LEAVE
1\$: MOV RO,-(SP) ;; SAVE RO
MOV 22(SP),RO ;; GET ADDRESS OF ASCIZ STRING
CMPB #APTENV,\$ENV ;; RUNNING IN APT MODE
BNE 62\$;; NO GO CHECK FOR APT CONSOLE
BITB #APTPOOL,\$ENVM ;; SPOOL MESSAGE TO APT
BEQ 62\$;; NO GO CHECK FOR CONSOLE
MOV RO,61\$;; SETUP MESSAGE ADDRESS FOR APT
JSR PC,\$ATY3 ;; SPOOL MESSAGE TO APT
61\$: .WORD 0 ;; MESSAGE ADDRESS
62\$: BITB #APTCSUP,\$ENVM ;; APT CONSOLE SUPPRESSED
BNE 60\$;; YES, SKIP TYPE OUT
2\$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4\$;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60\$: MOV (SP)+,RO ;; RESTORE RO
3\$: ADD #2,(SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4\$: CMPB #HT,(SP) ;; BRANCH IF <HT>
BEQ 8\$
CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
BNE 5\$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
\$CRLF
CLRB \$CHARCNT ;; CLEAR CHARACTER COUNT
BR 2\$;; GET NEXT CH
5\$: JSR PC,\$TPEC ;; GO TYPE THIS FR
6\$: CMPB \$FILLC,(SP)+ ;; IS IT TIME FOR \$FILLC CHARS.?
BNE 2\$;; IF NO GO GET N
MOV \$NULL,-(SP) ;; GET # OF FILLER CHARS NEEDED

```

8129
8130 041656 105366 000001 7$: DECB 1(SP) ; AND THE NULL CHAR.
8131 041662 002770 BLT 6$ ; DOES A NULL NEED TO BE TYPED?
8132 041664 004737 041722 JSR PC,$TYPEC ; BR IF NO--GO POP THE NULL OFF OF STACK
8133 041670 105337 041766 DECB $CHARCNT ; GO TYPE A NULL
8134 041674 000770 BR 7$ ; DO NOT COUNT AS A COUNT
8135 ; LOOP
8136 ; HORIZONTAL TAB PROCESSOR
8137
8138 041676 112716 000040 8$: MOVB #' (SP) ; REPLACE TAB WITH SPACE
8139 041702 004737 041722 9$: JSR PC,$TYPEC ; TYPE A SPACE
8140 041706 132737 000007 041766 BITB #7,$CHARCNT ; BRANCH IF NOT AT
8141 041714 001372 BNE 9$ ; TAB STOP
8142 041716 005726 TST (SP)+ ; POP SPACE OFF STACK
8143 041720 000724 BR 2$ ; GET NEXT CHARACTER
8144 041722 105777 137222 $TYPEC: TSTB 2$TPS ; WAIT UNTIL PRINTER IS READY
8145 041726 100375 BPL $TYPEC
8146 041730 116677 000002 137214 MOVB 2(SP),2$TPB ; LOAD CHAR TO BE TYPED INTO DATA REG.
8147 041736 122766 000015 000002 CMPB #CR,2(SP) ; IS CHARACTER A CARRIAGE RETURN?
8148 041744 001003 BNE 1$ ; BRANCH IF NO
8149 041746 105037 041766 CLRB $CHARCNT ; YES--CLEAR CHARACTER COUNT
8150 041752 000406 BR $TYPEX ; EXIT
8151 041754 122766 000012 000002 1$: CMPB #LF,2(SP) ; IS CHARACTER A LINE FEED?
8152 041762 001402 BEQ $TYPEX ; BRANCH IF YES
8153 041764 105227 INCB (PC)+ ; COUNT THE CHARACTER
8154 041766 000000 $CHARCNT: WORD 0 ; CHARACTER COUNT STORAGE
8155 041770 000207 $TYPEX: RTS PC
8156
8157
8158
8159
8160 ; *****
8161 ; SBTTL DOUBLE-PRECISION MULTIPLY SUBROUTINE
8162 ; SUBROUTINE TO MULTIPLY TWO DOUBLE PRECISION INTEGERS
8163 ; USES ALL REGISTERS (R0-R5)
8164 ;
8165 ; ENTER WITH JSR PC,M.DPIM
8166 ; MULTIPLIER IN R2-R3
8167 ; MULTIPLICAND IN R4-R5
8168 ; PRODUCT RETURNED IN R0-R1-R2-R3
8169 ; *****
8170
8171 041772 005000 M.DPIM: CLR R0 ; CLEAR HI ORDER WORDS
8172 041774 005001 CLR R1
8173 041776 012746 000041 MOV #41,-(SP) ; MOVE 33 (DEC) TO COUNTER
8174 042002 006000 M.DP01: ROR R0
8175 042004 006001 ROR R1
8176 042006 006002 ROR R2 ; SHIFT TO ADD
8177 042010 006003 ROR R3
8178 042012 103003 BCC M.DP02 ; NO CARRY NO ADD
9179 042014 060501 ADD R5,R1
8180 042016 005500 ADC R0 ; ADD DOUBLE PRECISION TO OBTAIN NEW PARTIAL
8181 042020 060400 ADD R4,R0 ; PRODUCT
8182 042022 005316 M.DP02: DEC 2$P ; DECREMENT COUNTER
8183 042024 001366 BNE M.DP01
8184 042026 005726 TST (SP)+ ; REMOVE THE COUNTER

```

H13

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 163
 DOUBLE-PRECISION MULTIPLY SUBROUTINE

SEQ 0162

8185	042030	000207		RTS	PC				
8186									
8187									
8188									
8189									
8190									
8191									
8192									
8193									
8194									
8195									
8196									
8197									
8198									
8199									
8200	042032	012746	000040	M.DP10:	MOV	#40,-(SP)			;COUNTER FOR DIVISION CYCLES
8201	042036	010446			MOV	R4,-(SP)			;HI ORDER
8202	042040	010546			MOV	R5,-(SP)			;LO ORDER DIVISOR TO THE STACK
8203	042042	005466	000002		NEG	2(SP)			;FORM NEGATIVE
8204	042046	005416			NEG	2SP			;VERSION OF THE DIVISOR
8205	042050	005666	000002		SBC	2(SP)			
8206	042054	061601			ADD	2SP,R1			
8207	042056	005500			ADC	R0			;PERFORM THE INITIAL SUBTRACTION
8208	042060	066600	000002		ADD	2(SP),R0			
8209	042064	103445			BCS	M.DP50			;IF CARRY THEN OVERFLOW HAS OCCURRED
8210	042066	005046			CLR	-(SP)			;THIS IS A LONGER LASTING CARRY BIT
8211	042070	006103		M.DP40:	ROL	R3			
8212	042072	006102			ROL	R2			
8213	042074	006101			ROL	R1			
8214	042076	006100			ROL	R0			
8215	042100	005716			TST	2SP			;TEST "CARRY INDICATOR"
8216	042102	001410			BEQ	M.DP41			;IF NO "CARRY" THEN ADD ELSE SUBTRACT
8217	042104	005016			CLR	2SP			;CLEAR UP FOR NEXT TIME
8218	042106	066601	000002		ADD	2(SP),R1			
8219	042112	005500			ADC	R0			;ADD -(DIVISOR)
8220	042114	005516			ADC	2SP			;SET "CARRY"
8221	042116	066600	000004		ADD	4(SP),R0			
8222	042122	000404			BR	M.DP42			
8223	042124	060501		M.DP41:	ADD	R5,R1			
8224	042126	005500			ADC	R0			;ADD +(DIVISOR)
8225	042130	005516			ADC	2SP			;SET "CARRY"
8226	042132	060400			ADD	R4,R0			
8227	042134	005516		M.DP42:	ADC	2SP			;SET "CARRY"
8228	042136	005716			TST	2SP			;TEST THE UPDATE INDICATOR
8229	042140	001401			BEQ	.+4			;IF ZERO FORGET IT
8230	042142	005203			INC	R3			;NO CARRY POSSIBLE HERE
8231	042144	005366	000006		DEC	6(SP)			;DECREMENT COUNTER
8232	042150	003347			BGT	M.DP40			;BR IF MORE TO DO
8233	042152	006003			ROR	R3			
8234	042154	103404			BCS	M.DP44			
8235	042156	060501			ADD	R5,R1			
8236	042160	005500			ADC	R0			
8237	042162	060400			ADD	R4,R0			
8238	042164	000241			CLC				
8239	042166	006103		M.DP44:	ROL	R3			
8240	042170	062706	000010		ADD	#10,SP			;ADJUST STACK BY 4 WORDS

8241 042174 000242
 8242 042176 000207
 8243 042200 062706 000006
 8244 042204 000262
 8245 042206 000207

CLV
 RTS PC
 M.DPS0: ADD #b,SP
 SEV
 RTS PC

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

8251
 8252
 8253
 8254
 8255
 8256
 8257
 8258
 8259

```

*****
*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
*UNSIGNED OCTAL ASCII NUMBER.
*CALL
*   MOV     #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
*   JSR    PC,2#S0B20      ;; CALL THE ROUTINE
*   RETURN                                     ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
  
```

8260 042210 104407
 8261 042212 016601 000002
 8262 042216 012705 042327
 8263 042222 012704 000014
 8264 042226 012703 177770
 8265 042232 012100
 8266 042234 012101
 8267 042236 005002
 8268 042240 110245
 8269 042242 010002
 8270 042244 005304
 8271 042246 003007
 8272 042250 001405
 8273 042252 005205
 8274 042254 010566 000002
 8275 042260 104410
 8276 042262 000207
 8277 042264 006203
 8278 042266 006001
 8279 042270 006000
 8280 042272 006001
 8281 042274 006000
 8282 042276 006001
 8283 042300 006000
 8284 042302 040302
 8285 042304 062702 000060
 8286 042310 000753
 8287 042312 000016

```

S0B20: SAVREG      ;; SAVE ALL REGISTERS
        MOV     2(SP),R1  ;; PICKUP THE POINTER TO LOW WORD
        MOV     #S0CTVL+13.,R5  ;; POINTER TO DATA TABLE
        MOV     #12.,R4      ;; DO ELEVEN CHARACTERS
        MOV     #7,R3       ;; MASK
        MOV     (R1)+,R0     ;; LOWER WORD
        MOV     (R1)+,R1     ;; HIGH WORD
        CLR     R2          ;; TERMINATOR
1$:     MOV     R2,-(R5)     ;; PUT CHARACTER IN DATA TABLE
        MOV     R0,R2      ;; GET THIS DIGIT
        DEC     R4         ;; COUNT THIS CHARACTER
        BGT    3$         ;; BR IF NOT THE LAST DIGIT
        BEQ    2$         ;; BR IF IT IS THE LAST DIGIT
        INC     R5         ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
        MOV     R5,2(SP)   ;; ASCII CHAR. & PUT IT ON THE STACK
        RESREG             ;; RESTORE ALL REGISTERS
        RTS     PC        ;; RETURN TO USER
2$:     ASR     R3         ;; POSITION THE MASK FOR THE LAST DIGIT
3$:     ROR     R1         ;; POSITION THE BINARY NUMBER FOR
        ROR     R0         ;; THE NEXT OCTAL DIGIT
        ROR     R1
        ROR     R0
        ROR     R1
        ROR     R0
        BIC     R3,R2     ;; MASK OUT ALL JUNK
        ADD     #0,R2     ;; MAKE THIS CHAR. ASCII
        BR     1$        ;; GO PUT IT IN THE DATA TABLE
S0CTVL: .BLKB 14.      ;; RESERVE DATA TABLE
.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
  
```

8288
 8289
 8290
 8291
 8292
 8293
 8294
 8295
 8296

```

*****
*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,2#S0B20
  
```

```

8297 ;* RETURN ;: THE FIRST ADDRESS OF ASCIZ
8298 ;: IS ON THE STACK
8299
8300
8301 042330 104407 $D820: SAVREG ;: SAVE REGISTERS
8302 042332 016602 000002 MOV 2(SP),R2 ;: PICKUP THE DATA POINTER
8303 042336 012700 042510 MOV $DDECL,RO ;: GET ADDRESS OF "SOELVA" STRING
8304 042342 010066 000002 MOV RO,2(SP) ;: PUT ADDRESS OF ASCIZ STRING ON STACK
8305 042346 012201 MOV (R2)+,R1 ;: PICKUP THE BINARY NUMBER
8306 042350 012202 MOV (R2)+,R2
8307 042352 012737 000012 042426 MOV #10,R3 ;: SET UP TO DO 10 CONVERSIONS
8308 042360 012704 042440 MOV $STNPWR,R4 ;: ADDRESS OF TEN POWER
8309 042364 012705 042442 MOV $STNPWR+2,R5
8310 042370 005003 1$: CLR R3 ;: CLEAR PARTIAL
8311 042372 161401 2$: SUB (R4),R1 ;: SUBTRACT TEN POWER
8312 042374 005602 SBC R2
8313 042376 161502 SUB (R5),R2
8314 042400 002402 BLT 3$: ;: BR IF TEN POWER TO LARGE
8315 042402 005203 INC R3 ;: ADD 1 TO PARTIAL
8316 042404 000772 BR 2$ ;: LOOP
8317 042406 062401 3$: ADD (R4)+,R1 ;: RESTORE SUBTRACTED VALUE
8318 042410 005502 ADC R2
8319 042412 062402 ADD (R4)+,R2
8320 042414 022525 CMP (R5)+,(R5)+ ;: MOVE TO NEXT TEN POWER
8321 042416 052703 000060 BIS #0,R3 ;: CHANGE PARTIAL TO ASCII
8322 042422 110320 MOVB R3,(RO)+ ;: SAVE IT
8323 042424 005327 DEC (PC)+ ;: DONE?
8324 042426 000000 4$: .WORD 0
8325 042430 001357 BNE 1$ ;: BR IF NO
8326 042432 105020 CLRB (RO)+ ;: TERMINATOR
8327 042434 104410 RESREG ;: RESTORE REGISTERS
8328 042436 000207 RTS PC ;: RETURN
8329 042440 145000 $STNPWR: 145000 ;: 1.0E09
8330 042442 035632 35632 ;:
8331 042444 160400 160400 ;: 1.0E08
8332 042446 002765 2765 ;:
8333 042450 113200 113200 ;: 1.0E07
8334 042452 000230 230 ;:
8335 042454 041100 041100 ;: 1.0E06
8336 042456 000017 17 ;:
8337 042460 103240 103240 ;: 1.0E05
8338 042462 000001 1 ;:
8339 042464 023420 23420 ;: 1.0E04
8340 042466 000000 0 ;:
8341 042470 001750 1750 ;: 1.0E03
8342 042472 000000 0 ;:
8343 042474 000144 144 ;: 1.0E02
8344 042476 000000 0 ;:
8345 042500 000012 12 ;: 1.0E01
8346 042502 000000 0 ;:
8347 042504 000001 1 ;: 1.0E00
8348 042506 000000 0 ;:
8349 042510 000014 $DECL: .BLKB 12 ;: RESERVE STORAGE FOR ASCIZ STRING
8350 .SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
8351
8352 ;:*****

```

```

8353
8354
8355
8356
8357
8358
8359
8360 042524 010046
8361 042526 016600 000004
8362 042532 105710
8363 042534 001403
8364 042536 122720 000060
8365 042542 001773
8366 042544 005300
8367 042546 010037 042554
8368 042552 104401
8369 042554 000000
8370 042556 012600
8371 042560 012616
8372 042562 000207
8373
8374
8375
8376
8377
8378
8379
8380
8381
8382
8383
8384
8385
8386
8387
8388
8389
8390
8391
8392
8393
8394
8395
8396
8397
8398 042564 017646 000000
8399 042570 116637 000001 043007
8400 042576 112637 043011
8401 042602 062716 000002
8402 042606 000406
8403 042610 112737 000001 043007
8404 042616 112737 000006 043011
8405 042624 112737 000005 043006
8406 042632 010346
8407 042634 010446
8408 042636 010546

```

```

; *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
;
; $SUPRS: MOV   R0, -(SP)   ;; SAVE R0
;         MOV   4(SP), R0   ;; PICKUP THE POINTER
; 1$:     TSTB  (R0)        ;; TERMINATE OR?
;         BEQ  2$          ;; BR IF YES
;         CMPB #'0, (R0)+  ;; IS THIS AN ASCII "0" ?
;         BEQ  1$          ;; BR IF YES
; 2$:     DEC  R0          ;; BACKUP BY "1"
;         MOV  R0, 3$     ;; SAVE FOR TYPING
;         TYPE                ;; GO TYPE
; 3$:     .WORD 0          ;; ASCII POINTER GOES HERE
;         MOV  (SP)+, R0   ;; RESTORE R0
;         MOV  (SP)+, (SP) ;; RESTORE THE STACK
;         RTS  PC         ;; RETURN
;
; .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
;
; *****
; *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
; *OCTAL (ASCII) NUMBER AND TYPE IT.
; *$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 $TYPOC
; *CALL:
; *   MOV   NUM, -(SP)     ;; NUMBER TO BE TYPED
; *   TYPON                ;; CALL FOR TYPEOUT
; *
; *$STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
; *CALL:
; *   MOV   NUM, -(SP)     ;; NUMBER TO BE TYPED
; *   TYPOC                ;; CALL FOR TYPEOUT
; *
; $TYPOS: MOV   @2(SP), -(SP) ;; PICKUP THE MODE
;         MOVB  1(SP), $OFILL ;; LOAD ZERO FILL SWITCH
;         MOVB  (SP)+, $OMODE+1 ;; NUMBER OF DIGITS TO TYPE
;         ADD  #2, (SP)      ;; ADJUST RETURN ADDRESS
;         BR   $STYPON
; $TYPOC: MOVB  #1, $OFILL   ;; SET THE ZERO FILL SWITCH
;         MOVB  #6, $OMODE+1 ;; SET FOR SIX(6) DIGITS
; $STYPON: MOVB  #5, $OCNT   ;; SET THE ITERATION COUNT
;         MOV  R3, -(SP)    ;; SAVE R3
;         MOV  R4, -(SP)    ;; SAVE R4
;         MOV  R5, -(SP)    ;; SAVE R5

```



```

8409 042640 113704 043011
8410 042644 005404
8411 042646 062704 000006
8412 042652 110437 043010
8413 042658 113704 043007
8414 042662 016605 000012
8415 042666 005003
8416 042670 006105 1S:
8417 042672 000404 2S:
8418 042674 006105
8419 042676 006105
8420 042700 006105
8421 042702 010503
8422 042704 006103 3S:
8423 042706 105337 043010
8424 042712 100016
8425 042714 042703 177770
8426 042720 001002
8427 042722 005704
8428 042724 001403
8429 042726 005204 4S:
8430 042730 052703 000060
8431 042734 052703 000040 5S:
8432 042740 110337 043004
8433 042744 104401 043004
8434 042750 105337 043006 7S:
8435 042754 003347
8436 042756 002402
8437 042760 005204
8438 042762 000744
8439 042764 012605 6S:
8440 042766 012604
8441 042770 012603
8442 042772 016666 000002 000004
8443 043000 012616
8444 043002 000002
8445 043004 000
8446 043005 000
8447 043006 000
8448 043007 000
8449 043010 000000
8450
8451
8452
8453
8454
8455
8456
8457
8458
8459
8460
8461
8462 043012
8463 043012 010046
8464 043014 010146

```

```

MOV B $OMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
NEG R4
ADD #6,R4 ;; SUBTRACT IT FOR MAX. ALLOWED
MOV B R4,$OMODE ;; SAVE IT FOR USE
MOV B $OFILL,R4 ;; GET THE ZERO FILL SWITCH
MOV R12,(SP),R5 ;; PICKUP THE INPUT NUMBER
CLR R3 ;; CLEAR THE OUTPUT WORD
ROL R5 ;; ROTATE MSB INTO "C"
BR 3S ;; GO DO MSB
ROL R5 ;; FORM THIS DIGIT
ROL R5
MOV R5,R3
ROL R3
DECB $OMODE ;; GET LSB OF THIS DIGIT
BR 7S ;; TYPE THIS DIGIT?
BPL 7S ;; BR IF NO
BIC #177770,R3 ;; GET RID OF JUNK
BNE 4S ;; TEST FOR 0
TST R4 ;; SUPPRESS THIS 0?
BEQ 5S ;; BR IF YES
INC R4 ;; DON'T SUPPRESS ANYMORE 0'S
BIS #'0,R3 ;; MAKE THIS DIGIT ASCII
BIS #' ,R3 ;; MAKE ASCII IF NOT ALREADY
MOV B R3,B5 ;; SAVE FOR TYPING
TYPE B5 ;; GO TYPE THIS DIGIT
DECB $OCNT ;; COUNT BY 1
BGT 2S ;; BR IF MORE TO DO
BLT 6S ;; BR IF DONE
INC R4 ;; INSURE LAST DIGIT ISN'T A BLANK
BR 2S ;; GO DO THE LAST DIGIT
MOV (SP)+,R5 ;; RESTORE R5
MOV (SP)+,R4 ;; RESTORE R4
MOV (SP)+,R3 ;; RESTORE R3
MOV 2(SP),4(SP) ;; SET THE STACK FOR RETURNING
MOV (SP)+,(SP)
RTI ;; RETURN
8S: .BYTE 0 ;; STORAGE FOR ASCII DIGIT
.BYTE 0 ;; TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE 0 ;; OCTAL DIGIT COUNTER
$OFILL: .BYTE 0 ;; ZERO FILL SWITCH
$OMODE: .WORD 0 ;; NUMBER OF DIGITS TO TYPE
.SBTL 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
$TYPDS:
MOV R0,-(SP) ;; PUSH R0 ON STACK
MOV R1,-(SP) ;; PUSH R1 ON STACK

```

M13

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
 DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 168
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0167

8465	043016	010246			MOV	R2,-(SP)	;; PUSH R2 ON STACK	
8466	043020	010346			MOV	R3,-(SP)	;; PUSH R3 ON STACK	
8467	043022	010546			MOV	R5,-(SP)	;; PUSH R5 ON STACK	
8468	043024	012746	020200		MOV	#20200,-(SP)	;; SET BLANK SWITCH AND SIGN	
8469	043030	016605	000020		MOV	20(SP),R5	;; GET THE INPUT NUMBER	
8470	043034	100004			BPL	1\$;; BR IF INPUT IS POS.	
8471	043036	005405			NEG	R5	;; MAKE THE BINARY NUMBER POS.	
8472	043040	112766	000055	000001	MOVB	#'-,1(SP)	;; MAKE THE ASCII NUMBER NEG.	
8473	043046	005000			1\$:	CLR	R0	;; ZERO THE CONSTANTS INDEX
8474	043050	012703	043226		MOV	#\$DBLK,R3	;; SETUP THE OUTPUT POINTER	
8475	043054	112723	000040		MOVB	#',(R3)+	;; SET THE FIRST CHARACTER TO A BLANK	
8476	043060	005002			2\$:	CLR	R2	;; CLEAR THE BCD NUMBER
8477	043062	016001	043216		MOV	\$DTBL(R0),R1	;; GET THE CONSTANT	
8478	043066	160105			3\$:	SUB	R1,R5	;; FORM THIS BCD DIGIT
8479	043070	002402				BLT	4\$;; BR IF DONE
8480	043072	005202				INC	R2	;; INCREASE THE BCD DIGIT BY 1
8481	043074	000774				BR	3\$	
8482	043076	060105			4\$:	ADD	R1,R5	;; ADD BACK THE CONSTANT
8483	043100	005702				TST	R2	;; CHECK IF BCD DIGIT=0
8484	043102	001002				BNE	5\$;; FALL THROUGH IF 0
8485	043104	105716				TSTB	(SP)	;; STILL DOING LEADING 0'S?
8486	043106	100407				BMI	7\$;; BR IF YES
8487	043110	106316			5\$:	ASLB	(SP)	;; MSD?
8488	043112	103003				BCC	6\$;; BR IF NO
8489	043114	116663	000001	177777		MOVB	1(SP)-1(R3)	;; YES--SET THE SIGN
8490	043122	052702	000060		6\$:	BIS	#'0,R2	;; MAKE THE BCD DIGIT ASCII
8491	043126	052702	000040		7\$:	BIS	#'0,R2	;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
8492	043132	110223				MOVB	R2,(R3)+	;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
8493	043134	005720				TST	(R0)+	;; JUST INCREMENTING
8494	043136	020027	000010			CMP	R0,#10	;; CHECK THE TABLE INDEX
8495	043142	002746				BLT	2\$;; GO TO THE NEXT DIGIT
8496	043144	003002				BGT	8\$;; GO TO EXIT
8497	043146	010502				MOV	R5,R2	;; GET THE LSD
8498	043150	000764				BR	6\$;; GO CHANGE TO ASCII
8499	043152	105726			8\$:	TSTB	(SP)+	;; WAS THE LSD THE FIRST NON-ZERO?
8500	043154	100003				BPL	9\$;; BR IF NO
8501	043156	116663	177777	177776		MOVB	-1(SP),-2(R3)	;; YES--SET THE SIGN FOR TYPING
8502	043164	105013			9\$:	CLRB	(R3)	;; SET THE TERMINATOR
8503	043166	012605				MOV	(SP)+,R5	;; POP STACK INTO R5
8504	043170	012603				MOV	(SP)+,R3	;; POP STACK INTO R3
8505	043172	012602				MOV	(SP)+,R2	;; POP STACK INTO R2
8506	043174	012601				MOV	(SP)+,R1	;; POP STACK INTO R1
8507	043176	012600				MOV	(SP)+,R0	;; POP STACK INTO R0
8508	043200	104401	043226			TYPE	\$DBLK	;; NOW TYPE THE NUMBER
8509	043204	016666	000002	000004		MOV	2(SP),4(SP)	;; ADJUST THE STACK
8510	043212	012616				MOV	(SP)+,(SP)	
8511	043214	000002				RTI		;; RETURN TO USER
8512	043216	023420			\$DTBL:	10000.		
8513	043220	001750				1000.		
8514	043222	000144				100.		
8515	043224	000012				10.		
8516	043226	000004			\$DBLK:	.BLKW 4		
8517					.SBTTL	ERROR HANDLER ROUTINE		
8518								
8519								
8520								

;;*****
 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.

ERROR HANDLER ROUTINE

8521
8522
8523
8524
8525
8526
8527
8528
8529
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540
8541
8542
8543
8544
8545
8546
8547
8548
8549
8550
8551
8552
8553
8554
8555
8556
8557
8558
8559
8560
8561
8562
8563
8564
8565
8566
8567
8568
8569
8570
8571
8572
8573
8574
8575
8576

043236
043236 105237 001103
043242 001775
043244 013777 001102 135670
043252 032777 002000 135660
043260 001402
043262 104401 001320
043266 005237 001112
043272 011637 001116
043276 162737 000002 001116
043304 117737 135606 001114
043312 032777 020000 135620
043320 001004
043322 004737 027506
043326 104401 001325
043332
043332 122737 000001 001350
043340 001007
043342 113737 001114 043354
043350 004737 044130
043354 000
043355 000
043356 000777
043360 005777 135554
043364 100001
043366 000000
043370 032777 001000 135542
043376 001402
043400 013716 001110
043404 005737 001316
043410 001402
043412 013716 001316
043416
043416 022737 016712 000042
043424 001001
043426 000000
043430
043430 000002

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

$ERROR:
7$: INCB SERFLG ;; SET THE ERROR FLAG
BEQ 7$ ;; DON'T LET THE FLAG GO TO ZERO
MOV $STNM, $DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10, $SWR ;; BELL ON ERROR?
BEQ 1$ ;; NO - SKIP
TYPE $BELL ;; RING BELL
1$: INC $ERTTL ;; COUNT THE NUMBER OF ERRORS
MOV (SP), $ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
SUB #2, $ERRPC
MOVB $ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13, $SWR ;; SKIP TYPEOUT IF SET
BNE 20$ ;; SKIP TYPEOUTS
JSR PC, TYPERR ;; GO TO USER ERROR ROUTINE
TYPE , $CRLF

20$: CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
BNE 2$ ;; NO SKIP APT ERROR REPORT
MOVB $ITEMB, 21$ ;; SET ITEM NUMBER AS ERROR NUMBER
JSR PC, $ATY4 ;; REPORT FATAL ERROR TO APT

21$: .BYTE 0
.BYTE 0

22$: BR 22$ ;; APT ERROR LOOP
2$: TST $SWR ;; HALT ON ERROR
BPL 3$ ;; SKIP IF CONTINUE
HALT ;; HALT ON ERROR!
3$: BIT #BIT09, $SWR ;; LOOP ON ERROR SWITCH SET?
BEQ 4$ ;; BR IF NO
MOV $LPERR, (SP) ;; FUDGE RETURN FOR LOOPING
TST $ESCAPE ;; CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ ;; BR IF NONE
MOV $ESCAPE, (SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE

5$: CMP #SENDAD, #42 ;; ACT-11 AUTO-ACCEPT?
BNE 6$ ;; BRANCH IF NO
HALT ;; YES

6$: RTI ;; RETURN

.SBTTL TTY INPUT ROUTINE

;*****
;ENABL LSB
;DSABL LSB
```

```

8577
8578
8579
8580
8581
8582
8583
8584
8585 043432 011646
8586 043434 016666 000004 000002
8587 043442 105777 135476
8588 043446 100375
8589 043450 117766 135472 000004
8590 043456 042766 177600 000004
8591 043464 026627 000004 000023
8592 043472 001013
8593 043474 105777 135444
8594 043500 100375
8595 043502 117746 135440
8596 043506 042716 177600
8597 043512 022627 000021
8598 043516 001366
8599 043520 000750
8600 043522 026627 000004 000140
8601 043530 002407
8602 043532 026627 000004 000175
8603 043540 003003
8604 043542 042766 000040 000004
8605 043550 000002
8606 043552 052536 005015 000
8607 043557 136 006507 000012
8608 043564 005015 053523 020122
8609 043572 020075 000
8610 043575 040 047040 053505
8611 043602 036440 000040
8612
8613
8614
8615
8616
8617
8618 043606 012737 043620 000024
8619 043614 000000
8620 043616 000776
8621
8622
8623 043620 005037 043672
8624 043624 005237 043672
8625 043630 001375
8626 043632 012737 043606 000024
8627 043640 012737 000340 000026
8628 043646 012737 000340 000036
8629 043654 012706 001100
8630 043660 104401 043674
8631 043664 000005
8632 043666 000177 135214

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

SRDCR: MOV (SP), -(SP) ; PUSH DOWN THE PC
MOV 4(SP), 2(SP) ; SAVE THE PS
1$: TSTB 2$TKS ; WAIT FOR
BPL 1$ ; A CHARACTER
MOV 2$TKB, 4(SP) ; READ THE TTY
BIC 81C(177), 4(SP) ; GET RID OF JUNK IF ANY
CMP 4(SP), 23 ; IS IT A CONTROL-S?
BNE 3$ ; BRANCH IF NO
2$: TSTB 2$TKS ; WAIT FOR A CHARACTER
BPL 2$ ; LOOP UNTIL ITS THERE
MOV 2$TKB, -(SP) ; GET CHARACTER
BIC 81C(177), (SP) ; MAKE IT 7-BIT ASCII
CMP (SP)+, 21 ; IS IT A CONTROL-Q?
BNE 2$ ; IF NOT DISCARD IT
BR 1$ ; YES, RESUME
3$: CMP 4(SP), 140 ; IS IT UPPER CASE?
BLT 4$ ; BRANCH IF YES
CMP 4(SP), 175 ; IS IT A SPECIAL CHAR?
BGT 4$ ; BRANCH IF YES
BIC 840, 4(SP) ; MAKE IT UPPER CASE
4$: RTI ; GO BACK TO USER
SCNTLU: .ASCIZ /1U/1512 ; CONTROL "U"
SCNTLG: .ASCIZ /1G/1512 ; CONTROL "G"
SMSWR: .ASCIZ <15>12/SWR = /
SMNEW: .ASCIZ / NEW = /

.SBTTL POWER DOWN AND UP ROUTINES

: POWER DOWN ROUTINE
$PWRDN: MOV $PWRUP, PWRVEC ; SET VECTOR FOR POWER UP
HALT ; HANG UP
BR -2

: POWER UP ROUTINE
$PWRUP: CLR $PWRCT ; WAIT LOOP FOR TTY TO COME UP
4$: INC $PWRCT
BNE 4$
MOV $PWRDN, PWRVEC ; SET VECTOR FOR POWER DOWN
MOV $PR7, PWRVEC+2 ; RE-ESTABLISH POWER AND TRAP PRIORITIES
MOV $PR7, TRAPVEC+2
MOV $STACK, SP ; RE-INITIALIZE THE STACK
TYPE ,PWRMSG ; TYPE "POWER FAILED"
RESET ; CLEAR THE UNIBUS
JMP 2$LPADR ; RESTART THE CURRENT TEST

```

```

8633 043672 000000
8634 043674 005015 047520 042527
8635 043702 020122 040506 046111
8636 043710 042105 005015 000
8637 043716
8638
8639
8640
8641
8642
8643
8644
8645 043716 105737 001103
8646 043722 001406
8647 043724 032777 001000 135206
8648 043732 001402
8649 043734 013716 001110
8650 043740 000002
8651
8652
8653
8654
8655
8656
8657
8658
8659
8660
8661
8662
8663
8664
8665
8666 043742
8667 043742 032777 040000 135170
8668 043750 001052
8669
8670 043752 000416
8671
8672 043754 013746 000004
8673 043760 012737 044000 000004
8674 043766 005737 177060
8675 043772 012637 000004
8676 043776 000421
8677 044000 022626
8678 044002 012637 000004
8679 044006 000407
8680 044010
8681 044010 105737 001103
8682 044014 001412
8683 044016 032777 001000 135114
8684 044024 001404
8685 044026 013737 001110 001106
8686 044034 000420
8687 044036 105037 001103
8688 044042 105237 001102

```

```

$PWRC: .WORD 0
$PWRMSG: .ASCIZ <15><12>/POWER FAILED/<15><12>

.EVEN

*****
* SCOPE1 - INTERNAL SCOPE ON ERROR ROUTINE
* CALLED BY "SCOPE"
*****
SCOPE1: TSTB $ERFLG ;SEE IF AN ERROR HAS OCCURRED
        BEQ 6$ ;BR IF NOT
        BIT #BIT9,$SWR ;SEE IF LOOP ON ERROR DESIRED
        BEQ 6$ ;BR IF NOT
        MOV $LPERR,(SP) ;SET ERROR LOOP ADDRESS ON STACK
6$: RTI ;RETURN

.SBTTL SCOPE HANDLER ROUTINE

*****
* THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
* AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
* AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
* THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
* SW14=1 LOOP ON TEST
* SW09=1 LOOP ON ERROR
* CALL SCOPE ;;SCOPE=IOT
*****
$SCOPE:
1$: BIT #BIT14,$SWR ;;LOOP ON PRESENT TEST?
   BNE $OVER ;;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
        MOV @ERRVEC,-(SP) ;THIS INSTRUCTION TO A "NOP" (NOP=240)
        MOV #5,$ERRVEC ;SAVE THE CONTENTS OF THE ERROR VECTOR
        TST @177060 ;SET FOR TIME OUT
        MOV (SP)+,$ERRVEC ;RESTORE THE ERROR VECTOR
        BR $SVLAD ;GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
   MOV (SP)+,$ERRVEC ;RESTORE THE ERROR VECTOR
   BR 7$ ;LOOP ON THE PRESENT TEST
6$;*****END OF CODE FOR THE XOR TESTER*****
2$: TSTB $ERFLG ;HAS AN ERROR OCCURRED?
   BEQ $SVLAD ;BR IF NO
   BIT #BIT09,$SWR ;LOOP ON ERROR?
   BEQ 4$ ;BR IF NO
7$: MOV $LPERR,$LPADR ;SET LOOP ADDRESS TO LAST SCOPE
   BR $OVER
4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
$SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS

```

SCOPE HANDLER ROUTINE

```

8689 044046 113737 001102 001334      MOV      $STNM,$STNM      ;; SET TEST NUMBER IN APT MAILBOX
8690 044054 011637 001106              MOV      (SP), $LPADR     ;; SAVE SCOPE LOOP ADDRESS
8691 044060 011637 001110              MOV      (SP), $LPERR     ;; SAVE ERROR LOOP ADDRESS
8692 044064 005037 001316              CLR      $ESCAPE         ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
8693 044070 112737 000001 001115      MOV      #1,$ERMAX       ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
8694 044076 013777 001102 135036 $OVER:  MOV      $STNM,$DISPLAY  ;; DISPLAY TEST NUMBER
8695 044104 013716 001106              MOV      $LPADR,(SP)     ;; FUDGE RETURN ADDRESS
8696 044110 000002              RTI                      ;; FIXES PS

```

.SBTTL APT COMMUNICATIONS ROUTINE

```

8697
8698
8699
8700 044112 112737 000001 044356 $ATY1:  MOV      #1,$FFLG      ;; TO REPORT FATAL ERROR
8701 044120 112737 000001 044354 $ATY3:  MOV      #1,$MFLG      ;; TO TYPE A MESSAGE
8702 044126 000403              BR        $ATYC
8703 044130 112737 000001 044356 $ATY4:  MOV      #1,$FFLG      ;; TO ONLY REPORT FATAL ERROR
8704 044136 $ATYC:
8705 044136 010046              MOV      RO,-(SP)        ;; PUSH RO ON STACK
8706 044140 010146              MOV      R1,-(SP)        ;; PUSH R1 ON STACK
8707 044142 105737 044354      TSTB     $MFLG           ;; SHOULD TYPE A MESSAGE?
8708 044146 001450              BEQ      $S              ;; IF NOT: BR
8709 044150 122737 000001 001350      CMPEB   $APTENV,$ENV     ;; OPERATING UNDER APT?
8710 044156 001031              BNE      $S              ;; IF NOT: BR
8711 044160 132737 000100 001351      BITB     $APTPOOL,$ENVM  ;; SHOULD SPOOL MESSAGES?
8712 044166 001425              BEQ      $S              ;; IF NOT: BR
8713 044170 017600 000004              MOV      #4(SP),RO       ;; GET MESSAGE ADDR.
8714 044174 062766 000002 000004      ADD      #2,4(SP)        ;; BUMP RETURN ADDR.
8715 044202 005737 001330 1$:      TST      $MSGTYPE        ;; SEE IF DONE W/ LAST XMISSION?
8716 044206 001375              BNE      $S              ;; IF NOT: WAIT
8717 044210 010037 001344      MOV      RO,$MSGAD       ;; PUT ADDR IN MAILBOX
8718 044214 105720 2$:      TSTB     (RO)+           ;; FIND END OF MESSAGE
8719 044216 001376              BNE      $S
8720 044220 163700 001344      SUB      $MSGAD,RO       ;; SUB START OF MESSAGE
8721 044224 006200              ASR      RO              ;; GET MESSAGE LNTH IN WORDS
8722 044226 010037 001346      MOV      RO,$MSGGLT      ;; PUT LENGTH IN MAILBOX
8723 044232 012737 000004 001330      MOV      #4,$MSGTYPE     ;; TELL APT TO TAKE MSG.
8724 044240 000413              BR        $S
8725 044242 017637 000004 044266 3$:      MOV      #4(SP),4$       ;; PUT MSG ADDR IN JSR LINKAGE
8726 044250 062766 000002 000004      ADD      #2,4(SP)        ;; BUMP RETURN ADDRESS
8727 044256 013746 177776              MOV      177776,-(SP)    ;; PUSH 177776 ON STACK
8728 044262 004737 041510      JSR      PC,$TYPE        ;; CALL TYPE MACRO
8729 044266 000000 4$:      .WORD    0
8730 044270 5$:
8731 044270 105737 044356 10$:      TSTB     $FFLG           ;; SHOULD REPORT FATAL ERROR?
8732 044274 001416              BEQ      $S              ;; IF NOT: BR
8733 044276 105737 001350      TST      $ENV            ;; RUNNING UNDER APT?
8734 044302 001413              BEQ      $S              ;; IF NOT: BR
8735 044304 005737 001330 11$:      TST      $MSGTYPE        ;; FINISHED LAST MESSAGE?
8736 044310 001375              BNE      $S              ;; IF NOT: WAIT
8737 044312 017637 000004 001332      MOV      #4(SP),$FATAL   ;; GET ERROR #
8738 044320 062766 000002 000004      ADD      #2,4(SP)        ;; BUMP RETURN ADDR.
8739 044326 005237 001330      INC      $MSGTYPE        ;; TELL APT TO TAKE ERROR
8740 044332 105037 044356 12$:      CLRB     $FFLG           ;; CLEAR FATAL FLAG
8741 044336 105037 044355      CLRB     $LFLG           ;; CLEAR LOG FLAG
8742 044342 105037 044354      CLRB     $MFLG           ;; CLEAR MESSAGE FLAG
8743 044346 012601              MOV      (SP)+,R1        ;; POP STACK INTO R1
8744 044350 012600              MOV      (SP)+,RO        ;; POP STACK INTO RO

```

APT COMMUNICATIONS ROUTINE

```

8745 044352 000207
8746 044354 000
8747 044355 000
8748 044356 000
8749 044360
8750 000200
8751 000001
8752 000100
8753 000040
8754
8755
8756
8757
8758
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768
8769
8770
8771 044360
8772 044360 010046
8773 044362 010146
8774 044364 010246
8775 044366 010346
8776 044370 010446
8777 044372 010546
8778 044374 016646 000022
8779 044400 016646 000022
8780 044404 016646 000022
8781 044410 016646 000022
8782 044414 000002
8783
8784
8785
8786
8787 044416
8788 044416 012666 000022
8789 044422 012666 000022
8790 044426 012666 000022
8791 044432 012666 000022
8792 044436 012605
8793 044440 012604
8794 044442 012603
8795 044444 012602
8796 044446 012601
8797 044450 012600
8798 044452 000002
8799
8800

```

```

RTS PC ;: RETURN
SMFLG: .BYTE 0 ;: MESSG. FLAG
SLFLG: .BYTE 0 ;: LOG FLAG
SFFLG: .BYTE 0 ;: FATAL FLAG
.EVEN
APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
.SBTTL SAVE AND RESTORE RO-R5 ROUTINES
;: *****
;: *SAVE RO-R5
;: *CALL:
;: * SAVREG
;: *UPON RETURN FROM $$SAVREG THE STACK WILL LOOK LIKE:
;: *
;: *TOP---(+16)
;: * +2---(+18)
;: * +4---R5
;: * +6---R4
;: * +8---R3
;: * +10---R2
;: * +12---R1
;: * +14---R0
$$SAVREG:
MOV RO, -(SP) ;: PUSH RO ON STACK
MOV R1, -(SP) ;: PUSH R1 ON STACK
MOV R2, -(SP) ;: PUSH R2 ON STACK
MOV R3, -(SP) ;: PUSH R3 ON STACK
MOV R4, -(SP) ;: PUSH R4 ON STACK
MOV R5, -(SP) ;: PUSH R5 ON STACK
MOV 22(SP), -(SP) ;: SAVE PS OF MAIN FLOW
MOV 22(SP), -(SP) ;: SAVE PC OF MAIN FLOW
MOV 22(SP), -(SP) ;: SAVE PS OF CALL
MOV 22(SP), -(SP) ;: SAVE PC OF CALL
RTI
;: *RESTORE RO-R5
;: *CALL:
;: * RESREG
$$RESREG:
MOV (SP)+, 22(SP) ;: RESTORE PC OF CALL
MOV (SP)+, 22(SP) ;: RESTORE PS OF CALL
MOV (SP)+, 22(SP) ;: RESTORE PC OF MAIN FLOW
MOV (SP)+, 22(SP) ;: RESTORE PS OF MAIN FLOW
MOV (SP)+, R5 ;: POP STACK INTO R5
MOV (SP)+, R4 ;: POP STACK INTO R4
MOV (SP)+, R3 ;: POP STACK INTO R3
MOV (SP)+, R2 ;: POP STACK INTO R2
MOV (SP)+, R1 ;: POP STACK INTO R1
MOV (SP)+, R0 ;: POP STACK INTO R0
RTI
.SBTTL TRAP DECODER

```

```

8801
8802
8803
8804
8805
8806
8807 044454 010046
8808 044456 016600 000002
8809 044462 005740
8810 044464 111000
8811 044466 006300
8812 044470 016000 044510
8813 044474 000200
8814
8815
8816
8817
8818 044476 011646
8819 044500 016666 000004 000002
8820 044506 000002
8821
8822
8823
8824
8825
8826
8827
8828
8829 044510 044476
8830 044512 041510
8831 044514 042610
8832 044516 042564
8833 044520 042624
8834 044522 043012
8835
8836
8837 044524 043432
8838 044526 044360
8839 044530 044416
8840 044532 043716
8841
8842 044534 020040 020040 042524
8843 044542 052123 000040
8844 044546 025052 020040 000
8845 044553 125 044516 052502
8846 044560 020123 040520 044522
8847 044566 054524 042440 051122
8848 044574 051117 000
8849 044577 116 047117 042455
8850 044604 044530 052123 047101
8851 044612 020124 042515 047515
8852 044620 054522 000
8853 044623 116 047117 042455
8854 044630 044530 052123 047101
8855 044636 020124 051104 053111
8856 044644 000105

```

```

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

```

```

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

```

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

```

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

```

```
.SBTTL TRAP TABLE
```

```

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

```

```

ROUTINE
-----
$TRPAD: .WORD  $TRAP2
        $TYPE  ;; CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
        $TYPOC ;; CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS ;; CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON ;; CALL=TYPON    TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPDS ;; CALL=TYPDS    TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

        $RDCHR ;; CALL=RDCHR    TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
        $SAVREG ;; CALL=SAVREG  TRAP+7(104407) SAVE RO-RS ROUTINE
        $RESREG ;; CALL=RESREG  TRAP+10(104410) RESTORE RO-RS ROUTINE
        $SCOPE1 ;; CALL=SCOPE1  TRAP+11(104411) INTERNAL LOOP ON ERROR ROUTINE

```

```

TSTMSG: .ASCIZ / TEST /
AS2SP2: .ASCIZ /** /
EM1:    .ASCIZ /UNIBUS PARITY ERROR/

EM2:    .ASCIZ /NON-EXISTANT MEMORY/

EM3:    .ASCIZ /NON-EXISTANT DRIVE/

```


8857	044646	047125	052111	043040	EM4:	.ASCIZ	/UNIT FIELD ERROR/
8858	044654	042511	042114	042440			
8859	044662	051122	051117	000			
8860	044667	123	041125	054523	EM5:	.ASCIZ	/SUBSYS TIMEOUT/
8861	044674	020123	044524	042515			
8862	044702	052517	000124				
8863	044706	020104	047524	041440	EM6:	.ASCIZ	/D TO C PARITY ERROR/
8864	044714	050040	051101	052111			
8865	044722	020131	051105	047522			
8866	044730	000122					
8867	044732	051104	053111	020105	EM7:	.ASCIZ	/DRIVE DETECTED PARITY ERROR/
8868	044740	042504	042524	052103			
8869	044746	042105	050040	051101			
8870	044754	052111	020131	051105			
8871	044762	047522	000122				
8872	044766	041501	046040	053517	EM10:	.ASCIZ	/AC LOW/
8873	044774	000					
8874	044775	123	042520	042105	EM11:	.ASCIZ	/SPEED LOSS/
8875	045002	046040	051517	000123			
8876	045010	046111	042514	040507	EM12:	.ASCIZ	/ILLEGAL FUNCTION/
8877	045016	020114	052506	041516			
8878	045024	044524	047117	000			
8879	045031	120	047522	051107	EM13:	.ASCIZ	/PROGRAMMING ERROR/
8880	045036	046501	044515	043516			
8881	045044	042440	051122	051117			
8882	045052	000					
8883	045053	116	047117	042455	EM14:	.ASCIZ	/NON-EXISTANT FUNCTION/
8884	045060	044530	052123	047101			
8885	045066	020124	052506	041516			
8886	045074	044524	047117	000			
8887	045101	104	044522	042526	EM15:	.ASCIZ	/DRIVE TYPE ERROR/
8888	045106	052040	050131	020105			
8889	045114	051105	047522	000122			
8890	045122	047506	046522	052101	EM16:	.ASCIZ	/FORMAT ERROR/
8891	045130	042440	051122	051117			
8892	045136	000					
8893	045137	127	044522	042524	EM17:	.ASCIZ	/WRITE LOCK ERROR/
8894	045144	046040	041517	020113			
8895	045152	051105	047522	000122			
8896	045160	051104	053111	020105	EM20:	.ASCIZ	/DRIVE UNSAFE/
8897	045166	047125	040523	042506			
8898	045174	000					
8899	045175	123	042505	020113	EM21:	.ASCIZ	/SEEK INCOMPLETE/
8900	045202	047111	047503	050115			
8901	045210	042514	042524	000			
8902	045215	103	046131	047111	EM22:	.ASCIZ	/CYLINDER OVERFLOW/
8903	045222	042504	020122	053117			
8904	045230	051105	046106	053517			
8905	045236	000					
8906	045237	111	046114	043505	EM23:	.ASCIZ	/ILLEGAL CYLINDER ADDRESS/
8907	045244	046101	041440	046131			
8908	045252	047111	042504	020122			
8909	045260	042101	051104	051505			
8910	045266	060123					
8911	045270	051104	053111	020105	EM24:	.ASCIZ	/DRIVE OFF TRACK/
8912	045276	043117	020106	051124			

8913	045304	041501	000113			
8914	045310	051104	053111	020105	EM25:	.ASCIZ /DRIVE TIMING ERROR/
8915	045316	044524	044515	043516		
8916	045324	042440	051122	051117		
8917	045332	000				
8918	045333	104	052101	020101	EM26:	.ASCIZ /DATA LATE/
8919	045340	040514	042524	000		
8920	045345	103	047117	051124	EM27:	.ASCIZ /CONTROLLER TIMEOUT/
8921	045352	046117	042514	020122		
8922	045360	044524	042515	052517		
8923	045366	000124				
8924	045370	050117	051105	052101	EM30:	.ASCIZ /OPERATION INCOMPLETE/
8925	045376	047511	020116	047111		
8926	045404	047503	050115	042514		
8927	045412	042524	000			
8928	045415	110	040505	042504	EM31:	.ASCIZ /HEADER VRC ERROR/
8929	045422	020122	051126	020103		
8930	045430	051105	047522	000122		
8931	045436	040504	040524	041440	EM32:	.ASCIZ /DATA CHECK ERROR/
8932	045444	042510	045503	042440		
8933	045452	051122	051117	000		
8934	045457	127	044522	042524	EM33:	.ASCIZ /WRITE CHECK ERROR/
8935	045464	041440	042510	045503		
8936	045472	042440	051122	051117		
8937	045500	000				
8938	045501	104	052101	020101	EM34:	.ASCIZ /DATA MISCOMPARE/
8939	045506	044515	041523	046517		
8940	045514	040520	042522	000		
8941	045521	116	020117	051104	EM35:	.ASCIZ /NO DRIVE RESPONSE-UFE & NXD/
8942	045526	053111	020105	042522		
8943	045534	050123	047117	042523		
8944	045542	052455	042506	023040		
8945	045550	047040	042130	000		
8946	045555	104	044522	042526	EM36:	.ASCIZ /DRIVE ERROR WILL NOT CLEAR/
8947	045562	042440	051122	051117		
8948	045570	053440	046111	020114		
8949	045576	047516	020124	046103		
8950	045604	040505	000122			
8951	045610	051104	053111	020105	EM37:	.ASCIZ /DRIVE STATUS CHANGE WILL NOT CLEAR/
8952	045616	052123	052101	051525		
8953	045624	041440	040510	043516		
8954	045632	020105	044527	046114		
8955	045640	047040	052117	041440		
8956	045646	042514	051101	000		
8957	045653	101	052124	047047	EM40:	.ASCIZ /ATT'N BUT NO STATUS CHANGE OR FAULT/
8958	045660	041040	052125	047040		
8959	045666	020117	052123	052101		
8960	045674	051525	041440	040510		
8961	045702	043516	020105	051117		
8962	045710	047040	052501	052114		
8963	045716	000				
8964	045717	101	052124	047047	EM41:	.ASCIZ /ATT'N BUT DRIVE NOT AVAILABLE/
8965	045724	041040	052125	042040		
8966	045732	044522	042526	047040		
8967	045740	052117	040440	040526		
8968	045746	046111	041101	042514		

8969	045754	000							
8970	045755	101	052124	047047	EM42:	.ASCIZ	/ATT'N WHEN NOT EXPECTED/		
8971	045762	053440	042510	020116					
8972	045770	047516	020124	054105					
8973	045776	042520	052103	042105					
8974	046004	000							
8975	046005	105	051122	051117	EM43:	.ASCIZ	/ERROR GATHERING DRIVE STATUS/		
8976	046012	043440	052101	042510					
8977	046020	044522	043516	042040					
8978	046026	044522	042526	051440					
8979	046034	040524	052524	000123					
8980	046042	052515	052114	050111	EM52:	.ASCIZ	/MULTIPLE DRIVE SELECT/		
8981	046050	042514	042040	044522					
8982	046056	042526	051440	046105					
8983	046064	041505	000124						
8984	046070	042510	042101	051105	EM53:	.ASCIZ	/HEADER COMPARE ERROR/		
8985	046076	041440	046517	040520					
8986	046104	042522	042440	051122					
8987	046112	051117	000						
8988	046115	123	041125	054523	EM56:	.ASCIZ	/SUBSYS TIMEOUT/		
8989	046122	020123	044524	042515					
8990	046130	052517	000124						
8991	046134	051105	047522	020122	EM60:	.ASCIZ	/ERROR IN RECAL FOR RECOVERY/		
8992	046142	047111	051040	041505					
8993	046150	046101	043040	051117					
8994	046156	051040	041505	053117					
8995	046164	051105	000131						
8996	046170	051120	043517	040522	EM61:	.ASCIZ	/PROGRAM ABORTING FATAL ERROR IN RETRY/		
8997	046176	020115	041101	051117					
8998	046204	044524	043516	043040					
8999	046212	052101	046101	042440					
9000	046220	051122	051117	044440					
9001	046226	020116	042522	051124					
9002	046234	000131							
9003	046236	054503	044514	042116	EM62:	.ASCIZ	/CYLINDER MISCOMPARE/		
9004	046244	051105	046440	051511					
9005	046252	047503	050115	051101					
9006	046260	000105							
9007	046262	046103	040505	020122	EM63:	.ASCIZ	/CLEAR CONTROLLER DID NOT CLEAR ERROR/		
9008	046270	047503	052116	047522					
9009	046276	046114	051105	042040					
9010	046304	042111	047040	052117					
9011	046312	041440	042514	051101					
9012	046320	042440	051122	051117					
9013	046326	000							
9014	046327	116	020117	052101	EM64:	.ASCIZ	/NO ATT'N IN ATT'N SUMMARY REGISTER/		
9015	046334	023524	020116	047111					
9016	046342	040440	052124	047047					
9017	046350	051440	046525	040515					
9018	046356	054522	051040	043505					
9019	046364	051511	042524	000122					
9020	046372	047125	047523	044514	EM65:	.ASCIZ	/UNSOLICITED ATTENTION/		
9021	046400	044503	042524	020104					
9022	046406	052101	042524	052116					
9023	046414	047511	000116						
9024	046420	047125	054105	042520	EM66:	.ASCIZ	/UNEXPECTED DATA TYPE ERROR/		

9025	046426	052103	042105	042040		
9026	046434	052101	020101	054524		
9027	046442	042520	042440	051122		
9028	046450	051117	000			
9029	046453	101	052124	047047	EM67:	.ASCIZ /ATT'N DID NOT RESET WITH CLEAR/
9030	046460	042040	042111	047040		
9031	046466	052117	051040	051505		
9032	046474	052105	053440	052111		
9033	046502	020110	046103	040505		
9034	046510	000122				
9035	046512	052523	051502	051531	EM70:	.ASCIZ /SUBSYS CLEAR DID NOT CLEAR DRIVE ATT'N/
9036	046520	041440	042514	051101		
9037	046526	042040	042111	047040		
9038	046534	052117	041440	042514		
9039	046542	051101	042040	044522		
9040	046550	042526	040440	052124		
9041	046556	047047	000			
9042	046561	104	052101	020101	EM71:	.ASCIZ /DATA LATE WHEN UNLOADING HEADER/
9043	046566	040514	042524	053440		
9044	046574	042510	020116	047125		
9045	046602	047514	042101	047111		
9046	046610	020107	042510	042101		
9047	046616	051105	000			
9048	046621	103	047117	051124	EM72:	.ASCIZ /CONTROLLER ERROR WHEN DRIVER SERVICING/
9049	046626	046117	042514	020122		
9050	046634	051105	047522	020122		
9051	046642	044127	047105	042040		
9052	046650	044522	042526	020122		
9053	046656	042523	053122	041511		
9054	046664	047111	000107			
9055	046670	051104	053111	020105	EM73:	.ASCIZ /DRIVE DETECTED PARITY ERROR/
9056	046676	042504	042524	052103		
9057	046704	042105	050040	051101		
9058	046712	052111	020131	051105		
9059	046720	047522	000122			
9060	046724	047125	042504	044506	EM74:	.ASCIZ /UNDEFINED ERROR/
9061	046732	042516	020104	051105		
9062	046740	047522	000122			
9063	046744	040515	045522	047111	EM75:	.ASCIZ /MARKING THIS SECTOR BAD/
9064	046752	020107	044124	051511		
9065	046760	051440	041505	047524		
9066	046766	020122	040502	000104		
9067	046774	040502	020104	040504	EM76:	.ASCIZ /BAD DATA VERIF'N WITH READ. ECC OF LAST RETRY IS:/
9068	047002	040524	053040	051105		
9069	047010	043111	047047	053440		
9070	047016	052111	020110	042522		
9071	047024	042101	020056	041505		
9072	047032	020103	043117	046040		
9073	047040	051501	020124	042522		
9074	047046	051124	020131	051511		
9075	047054	000072				
9076	047056	042522	051124	020131	EM77:	.ASCIZ /RETRY SUCCESSFUL/
9077	047064	052523	041503	051505		
9078	047072	043123	046125	000		
9079	047077	122	052105	054522	EM100:	.ASCIZ /RETRY UNSUCCESSFUL/
9080	047104	052440	051516	041525		

9081	047112	042503	051523	052506		
9082	047120	000114				
9083	047122	040503	047116	052117	EM101:	.ASCIZ /CANNOT FIND A VALID HEADER IN TRACK JUST READ/
9084	047130	043040	047111	020104		
9085	047136	020101	040526	044514		
9086	047144	020104	042510	042101		
9087	047152	051105	044440	020116		
9088	047160	051124	041501	020113		
9089	047166	052512	052123	051040		
9090	047174	040505	000104			
9091	047200	040502	020104	042523	EM102:	.ASCIZ /BAD SECTOR ERROR ON SECTOR NOT LISTED BAD/
9092	047206	052103	051117	042440		
9093	047214	051122	051117	047440		
9094	047222	020116	042523	052103		
9095	047230	051117	047040	052117		
9096	047236	046040	051511	042524		
9097	047244	020104	040502	000104		
9098	047252	044524	042515	026504	EM103:	.ASCIZ /TIMED-OUT ON READ HDR/
9099	047260	052517	020124	047117		
9100	047266	051040	040505	020104		
9101	047274	042110	000122			
9102	047300	044524	042515	026504	EM104:	.ASCIZ /TIMED-OUT ON SEEK/
9103	047306	052517	020124	047117		
9104	047314	051440	042505	000113		
9105	047322	051104	053111	020105	EM105:	.ASCIZ /DRIVE SIEZED BY OTHER PORT/
9106	047330	044523	055105	042105		
9107	047336	041040	020131	052117		
9108	047344	042510	020122	047520		
9109	047352	052122	000			
9110	047355	104	052101	020101	EM106:	.ASCIZ /DATA MISCMPR WHILE BAI SET/
9111	047362	044515	041523	050115		
9112	047370	020122	044127	046111		
9113	047376	020105	040502	020111		
9114	047404	042523	000124			
9115	047410	047516	047040	046505	EM107:	.ASCIZ /NO NEM ERROR WHEN REF'ING LOC 760000/
9116	047416	042440	051122	051117		
9117	047424	053440	042510	020116		
9118	047432	042522	023506	047111		
9119	047440	020107	047514	020103		
9120	047446	033067	030060	030060		
9121	047454	000				
9122	047455	111	052116	050122	EM110:	.ASCIZ /INTRPT WHEN CNTRLR NOT RDY/
9123	047462	020124	044127	047105		
9124	047470	041440	052116	046122		
9125	047476	020122	047516	020124		
9126	047504	042122	000131			
9127	047510	047516	040440	052124	EM111:	.ASCIZ /NO ATT'N ON SEEK/
9128	047516	047047	047440	020116		
9129	047524	042523	045505	000		
9130	047531	104	044522	042526	EM112:	.ASCIZ /DRIVE'S CYLINDER INCORRECT/
9131	047536	051447	041440	046131		
9132	047544	047111	042504	020122		
9133	047552	047111	047503	051122		
9134	047560	041505	000124			
9135	047564	041101	051117	026524	EM113:	.ASCIZ /ABORT- CAN'T READ BSF/
9136	047572	041440	047101	052047		

L14

9137	047600	051040	040505	020104						
9138	047606	051502	000106							
9139	047612	052113	030461	043040	EM114:	.ASCIZ	/KT11 FAILURE/			
9140	047620	044501	052514	042522						
9141	047626	000								
9142	047627	115	046505	050040	EM115:	.ASCIZ	/MEM PARITY ERR/			
9143	047634	051101	052111	020131						
9144	047642	051105	000122							
9145	047646	052503	051122	047105	DH100:	.ASCIZ	/CURRENT COMMAND :/			
9146	047654	020124	047503	046515						
9147	047662	047101	020104	000072						
9148	047670	051120	053105	047511	DH105:	.ASCIZ	/PREVIOUS COMMAND :/			
9149	047676	051525	041440	046517						
9150	047704	040515	042116	035040						
9151	047712	000								
9152	047713	122	033113	030461	DH200:	.ASCIZ	/RK611 REGISTERS :/			
9153	047720	051040	043505	051511						
9154	047726	042524	051522	035040						
9155	047734	000								
9156	047735	122	046505	044501	DH500:	.ASCIZ	/REMAINING REGISTERS NOT VALID/			
9157	047742	044516	043516	051040						
9158	047750	043505	051511	042524						
9159	047756	051522	047040	052117						
9160	047764	053040	046101	042111						
9161	047772	000								
9162	047773	124	042510	043040	DH501:	.ASCIZ	/THE FOLLOWING REGISTER DATA MAY BE INCORRECT :/			
9163	050000	046117	047514	044527						
9164	050006	043516	051040	043505						
9165	050014	1511	042524	020122						
9166	050022	040504	040524	046440						
9167	050030	054501	041040	020105						
9168	050036	047111	047503	051122						
9169	050044	041505	020124	000072						
9170	050052	051105	020122	041520	DH101:	.ASCIZ	/ERR PC DRIVE CMND CYLNRD TRACK SECTOR WD CNT/			
9171	050060	020040	051104	053111						
9172	050066	020105	020040	046503						
9173	050074	042116	020040	020040						
9174	050102	054503	047114	051104						
9175	050110	020040	051124	041501						
9176	050116	020113	020040	042523						
9177	050124	052103	051117	020040						
9178	050132	042127	041440	052116						
9179	050140	000								
9180	050141	110	020111	040502	DH102:	.ASCIZ	/HI BA LO BA/			
9181	050146	020040	046040	020117						
9182	050154	040502	000							
9183	050157	120	042522	027126	DH103:	.ASCIZ	/PREV. UNIBUS MAP :/			
9184	050164	052440	044516	052502						
9185	050172	020123	040515	020120						
9186	050200	000072								
9187	050202	044510	051040	043505	DH104:	.ASCIZ	/HI REGO LO REGO/			
9188	050210	020060	047514	051040						
9189	050216	043505	000060							
9190	050222	052503	027122	052440	DH106:	.ASCIZ	/CUR. UNIBUS MAP :/			
9191	050230	044516	052502	020123						
9192	050236	040515	020120	000072						

9305	051376	020105	052123	052101				
9306	051404	051525	000					
9307	051407	116	046525	042502	DH800:	.ASCIZ	/NUMBER OF RETRIES:/	
9308	051414	020122	043117	051040				
9309	051422	052105	044522	051505				
9310	051430	000072						
9311						.EVEN		
9312	051432	001116	005420	005416	DT100:	.WORD	SERRPC, DRIVE, ERRCOM, SREG0, SREG1, SREG2, SREG3	
9313	051440	001162	001164	001166				
9314	051446	001170						
9315	051450	001256	001260		DT102:	.WORD	SREG36, SREG37	
9316	051454	001174	001176	001200	DT201:	.WORD	SREG5, SREG6, SREG7, SREG10, SREG11, SREG12, SREG13	
9317	051462	001202	001204	001206				
9318	051470	001210						
9319	051472	001212	001214		DT202:	.WORD	SREG14, SREG15	
9320	051476	001216	001220	001222	DT203:	.WORD	SREG16, SREG17, SREG20, SREG21, SREG22, SREG23, SREG24, SREG25	
9321	051504	001224	001226	001230				
9322	051512	001232	001234					
9323	051516	001174	001176	001200	DT601:	.WORD	SREG5, SREG6, SREG7	
9324	051524	001202	001204	001206	DT602:	.WORD	SREG10, SREG11, SREG12, SREG13, SREG14, SREG15, SREG16	
9325	051532	001210	001212	001214				
9326	051540	001216						
9327								
9328	051542	000006			DF01:	.WORD	6	;NUMBER OF HEADER LINES
9329	051544	000				.BYTE	0	;NUMBER OF COL FOR FIRST HDR
9330	051545	000				.BYTE	0	;ALL CHARACTERS OCTAL
9331	051546	050052				.WORD	DH101	;SECOND HDR LINE
9332	051550	007	000			.BYTE	7,0	;NUM OF COL-ALL OCTAL
9333	051552	050141				.WORD	DH102	
9334	051554	002	000			.BYTE	2,0	
9335	051556	047713				.WORD	DH200	
9336	051560	000	000			.BYTE	0,0	
9337	051562	050244				.WORD	DH201	
9338	051564	007	000			.BYTE	7,0	
9339	051566	047735				.WORD	DH500	
9340	051570	000	000			.BYTE	0,0	
9341								
9342	051572	000007			DF02:	.WORD	7	
9343	051574	000	000			.BYTE	0,0	
9344	051576	050052				.WORD	DH101	
9345	051600	007	000			.BYTE	7,0	
9346	051602	050141				.WORD	DH102	
9347	051604	002	000			.BYTE	2,0	
9348	051606	047713				.WORD	DH200	
9349	051610	0,0	000			.BYTE	0,0	
9350	051612	050244				.WORD	DH201	
9351	051614	007	000			.BYTE	7,0	
9352	051616	050333				.WORD	DH202	
9353	051620	002	000			.BYTE	2,0	
9354	051622	050350				.WORD	DH203	
9355	051624	010	000			.BYTE	10,0	
9356								
9357	051626	000010			DF03:	.WORD	10	
9358	051630	000	000			.BYTE	0,0	
9359	051632	050052				.WORD	DH101	
9360	051634	007	000			.BYTE	7,0	

9361	051636	050141		.WORD	DH102	
9362	051640	002	000	.BYTE	2,0	
9363	051642	047713		.WORD	DH200	
9364	051644	000	000	.BYTE	0,0	
9365	051646	050244		.WORD	DH201	
9366	051650	007	000	.BYTE	7,0	
9367	051652	047773		.WORD	DH501	; "THE FOLLOWING REG DATA MB INCORRECT"
9368	051654	000	000	.BYTE	0,0	
9369	051656	050333		.WORD	DH202	
9370	051660	002	000	.BYTE	2,0	
9371	051662	050350		.WORD	DH203	
9372	05.164	010	000	.BYTE	10,0	
9373						
9374	051666	000003		DF04: .WORD	3	
9375	051670	000	000	.BYTE	0,0	
9376	051672	050052		.WORD	DH101	
9377	051674	007	000	.BYTE	7,0	
9378	051676	050141		.WORD	DH102	
9379	051700	002	000	.BYTE	2,0	
9380						
9381	051702	000007		DF05: .WORD	7	; OPI, HVRC
9382	051704	000	000	.BYTE	0,0	
9383	051706	050052		.WORD	DH101	
9384	051710	007	000	.BYTE	7,0	
9385	051712	050141		.WORD	DH102	
9386	051714	002	000	.BYTE	2,0	
9387	051716	050444		.WORD	DH601	
9388	051720	000	000	.BYTE	0,0	
9389	051722	050705		.WORD	DH606	
9390	051724	003	000	.BYTE	3,0	
9391	051726	050466		.WORD	DH602	
9392	051730	000	000	.BYTE	0,0	
9393	051732	050705		.WORD	DH606	
9394	051734	003	000	.BYTE	3,0	
9395						
9396	051736	000007		DF07: .WORD	7	
9397	051740	000	000	.BYTE	0,0	
9398	051742	050052		.WORD	DH101	
9399	051744	007	000	.BYTE	7,0	
9400	051746	050141		.WORD	DH102	
9401	051750	002	000	.BYTE	2,0	
9402	051752	050533		.WORD	DH604	
9403	051754	000	000	.BYTE	0,0	
9404	051756	050566		.WORD	DH6041	
9405	051760	003	000	.BYTE	3,0	
9406	051762	050633		.WORD	DH605	
9407	051764	000	000	.BYTE	0,0	
9408	051766	050650		.WORD	DH6051	
9409	051770	003	000	.BYTE	3,0	
9410						
9411	051772	000005		DF10: .WORD	5	
9412	051774	000	000	.BYTE	0,0	
9413	051776	050052		.WORD	DH101	
9414	052000	007	000	.BYTE	7,0	
9415	052002	050141		.WORD	DH102	
9416	052004	002	000	.BYTE	2,0	

9417	052006	050533		.WORD	DH604	
9418	052010	000	000	.BYTE	0,0	
9419	052012	050566		.WORD	DH6041	
9420	052014	003	000	.BYTE	3,0	
9421						
9422	052016	000004		DF11: .WORD	4	
9423	052020	000	000	.BYTE	0,0	
9424	052022	050533		.WORD	DH604	
9425	052024	000	000	.BYTE	0,0	
9426	052026	050566		.WORD	DH6041	
9427	052030	003	000	.BYTE	3,0	
9428	052032	050753		.WORD	DH701	
9429	052034	000	000	.BYTE	0,0	
9430						
9431	052036	000006		DF12: .WORD	6	
9432	052040	000	000	.BYTE	0,0	
9433	052042	050052		.WORD	DH101	
9434	052044	007	000	.BYTE	7,0	
9435	052046	050141		.WORD	DH102	
9436	052050	002	000	.BYTE	2,0	
9437	052052	047713		.WORD	DH200	
9438	052054	000	000	.BYTE	0,0	
9439	052056	050244		.WORD	DH201	
9440	052060	007	000	.BYTE	7,0	
9441	052062	051233		.WORD	DH204	
9442	052064	004	000	.BYTE	4,0	
9443						
9444	052066	000006		DF13: .WORD	6	
9445	052070	000	000	.BYTE	0,0	
9446	052072	050052		.WORD	DH101	
9447	052074	007	000	.BYTE	7,0	
9448	052076	050141		.WORD	DH102	
9449	052100	002	000	.BYTE	2,0	
9450	052102	050444		.WORD	DH601	
9451	052104	000	000	.BYTE	0,0	
9452	052106	050705		.WORD	DH606	
9453	052110	003	000	.BYTE	3,0	
9454	052112	051271		.WORD	DH502	
9455	052114	000	000	.BYTE	0,0	
9456						
9457	052116	000006		DF14: .WORD	6	
9458	052120	000	000	.BYTE	0,0	
9459	052122	050052		.WORD	DH101	
9460	052124	007	000	.BYTE	7,0	
9461	052126	050141		.WORD	DH102	
9462	052130	002	000	.BYTE	2,0	
9463	052132	050444		.WORD	DH601	
9464	052134	000	000	.BYTE	0,0	
9465	052136	050705		.WORD	DH606	
9466	052140	003	000	.BYTE	3,0	
9467	052142	051271		.WORD	DH502	
9468	052144	000	000	.BYTE	0,0	
9469						
9470	052146	000011		DF15: .WORD	11	
9471	052150	000	000	.BYTE	0,0	
9472	052152	050052		.WORD	DH101	

;FORMAT FOR 2ND LEVEL ERROR
 ;IN HEADER COMPARE ERROR
 ;AND 2ND LEVEL HEADER
 ;VRC ERROR

;FORMAT FOR 2ND LEVEL ERROR
 ;IN OPERATION INCOMPLETE ERROR

9473	052154	007	000	.BYTE	7,0
9474	052156	050141		.WORD	04102
9475	052160	002	000	.BYTE	2,0
9476	052162	047713		.WORD	04200
9477	052164	000	000	.BYTE	0,0
9478	052166	050244		.WORD	04201
9479	052170	007	000	.BYTE	7,0
9480	052172	050333		.WORD	04202
9481	052174	002	000	.BYTE	2,0
9482	052176	051332		.WORD	04503
9483	052200	000	000	.BYTE	0,0
9484	052202	047773		.WORD	04501
9485	052204	000	000	.BYTE	0,0
9486	052206	050350		.WORD	04203
9487	052210	010	000	.BYTE	10,0
9488					
9489	052212	000011		DF16: .WORD	11
9490	052214	000	000	.BYTE	0,0
9491	052216	050052		.WORD	04101
9492	052220	007	000	.BYTE	7,0
9493	052222	050141		.WORD	04102
9494	052224	002	000	.BYTE	2,0
9495	052226	047713		.WORD	04200
9496	052230	000	000	.BYTE	0,0
9497	052232	050244		.WORD	04201
9498	052234	007	000	.BYTE	7,0
9499	052236	050333		.WORD	04202
9500	052240	002	000	.BYTE	2,0
9501	052242	051271		.WORD	04502
9502	052244	000	000	.BYTE	0,0
9503	052246	047773		.WORD	04501
9504	052250	000	000	.BYTE	0,0
9505	052252	050350		.WORD	04203
9506	052254	010	000	.BYTE	10,0
9507					
9508	052256	000005		DF17: .WORD	5
9509	052260	000	000	.BYTE	0,0
9510	052262	050052		.WORD	04101
9511	052264	007	000	.BYTE	7,0
9512	052266	050141		.WORD	04102
9513	052270	002	000	.BYTE	2,0
9514	052272	051217		.WORD	04711
9515	052274	000	000	.BYTE	0,0
9516	052276	051042		.WORD	04702
9517	052300	002	000	.BYTE	2,0
9518					
9519	052302	000002		DF20: .WORD	2
9520	052304	000	000	.BYTE	0,0
9521	052306	050202		.WORD	04104
9522	052310	002	000	.BYTE	2,0
9523					
9524	052312	000002		DF21: .WORD	2
9525	052314	000	000	.BYTE	0,0
9526	052316	050650		.WORD	046051
9527	052320	003	000	.BYTE	3,0
9528					

9529	052322	000106		DF22:	.WORD	6
9530	052324	000	000		.BYTE	0,0
9531	052326	047646			.WORD	DH100
9532	052330	000	000		.BYTE	0,0
9533	052332	050052			.WORD	DH101
9534	052334	007	000		.BYTE	7,0
9535	052336	050141			.WORD	DH102
9536	052340	002	000		.BYTE	2,0
9537	052342	050222			.WORD	DH106
9538	052344	000	000		.BYTE	0,0
9539	052346	050202			.WORD	DH104
9540	052350	002	000		.BYTE	2,0
9541						
9542	052352	000002		DF23:	.WORD	2
9543	052354	000	000		.BYTE	0,0
9544	052356	051407			.WORD	DH800
9545	052360	001	000		.BYTE	1,0
9546						
9547	052362	000001		DF24:	.WORD	1
9548	052364	002	000		.BYTE	2,0
9549						
9550	052366	000000		DF25:	.WORD	0
9551	052370	007	000		.BYTE	7,0
9552						
9553	052372	000002		DF26:	.WORD	2
9554	052374	002	000		.BYTE	2,0
9555	052376	050350			.WORD	DH203
9556	052400	010	000		.BYTE	10,0
9557						
9558	052402	000005		DF27:	.WORD	5
9559	052404	000	000		.BYTE	0,0
9560	052406	050052			.WORD	DH101
9561	052410	007	000		.BYTE	7,0
9562	052412	050141			.WORD	DH102
9563	052414	002	000		.BYTE	2,0
9564	052416	051056			.WORD	DH703
9565	052420	000	000		.BYTE	0,0
9566	052422	051042			.WORD	DH702
9567	052424	002	000		.BYTE	2,0
9568						
9569	052426	000005		DF30:	.WORD	5
9570	052430	000	000		.BYTE	0,0
9571	052432	050052			.WORD	DH101
9572	052434	007	000		.BYTE	7,0
9573	052436	050141			.WORD	DH102
9574	052440	002	000		.BYTE	2,0
9575	052442	051122			.WORD	DH705
9576	052444	000	000		.BYTE	0,0
9577	052446	051102			.WORD	DH704
9578	052450	004	000		.BYTE	4,0
9579						
9580	052452	000005		DF31:	.WORD	5
9581	052454	000	000		.BYTE	0,0
9582	052456	050052			.WORD	DH101
9583	052460	007	000		.BYTE	7,0
9584	052462	050141			.WORD	DH102

9585	052464	002	000		.BYTE	2,0	
9586	052466	051136			.WORD	04706	
9587	052470	000	000		.BYTE	0,0	
9588	052472	051201			.WORD	04710	
9589	052474	003	000		.BYTE	3,0	
9590							
9591	052476	000400			RWBUF: .BLKW	256.	;READ/WRITE DATA BUFFER
9592							
9593	053476	005015	020040	020040	NOTMSG: .ASCII	<15><12>/	*** NOTE ***/<15><12><12>
9594	053504	020040	020040	020040			
9595	053512	025052	020052	047516			
9596	053520	042524	025040	025052			
9597	053526	005015	012				
9598	053531	101	046114	042040	.ASCII	/ALL DRIVES TO BE TESTED MUST HAVE :/<15><12><12>	
9599	053536	044522	042526	020123			
9600	053544	047524	041040	020105			
9601	053552	042524	052123	042105			
9602	053560	046440	051525	020124			
9603	053566	040510	042526	035040			
9604	053574	005015	012				
9605	053577	061	020056	042504	.ASCII	/1. DESIRED PORT SELECTED/<15><12>	
9606	053604	044523	042522	020104			
9607	053612	047520	052122	051440			
9608	053620	046105	041505	042524			
9609	053626	006504	012				
9610	053631	062	020056	051127	.ASCII	/2. WRITE LOCK DISABLED/<15><12>	
9611	053636	052111	020105	047514			
9612	053644	045503	042040	051511			
9613	053652	041101	042514	006504			
9614	053660	012					
9615	053661	104	044522	042526	.ASCII	/DRIVES NOT TO BE TESTED MUST HAVE BOTH/<15><12>	
9616	053666	020123	047516	020124			
9617	053674	047524	041040	020105			
9618	053702	042524	052123	042105			
9619	053710	046440	051525	020124			
9620	053716	040510	042526	041040			
9621	053724	052117	006510	012			
9622	053731	120	051117	051524	.ASCII	/PORTS DE-SELECTED./<15><12>	
9623	053736	042040	026505	042523			
9624	053744	042514	052103	042105			
9625	053752	006456	012				
9626	053755	124	042510	041440	.ASCII	/THE CARTRIDGE MUST BE FORMATTED ON A KNOWN GOOD/<15><12>	
9627	053762	051101	051124	042111			
9628	053770	042507	046440	051525			
9629	053776	020124	042502	043040			
9630	054004	051117	040515	052124			
9631	054012	042105	047440	020116			
9632	054020	020101	047113	053517			
9633	054026	020116	047507	042117			
9634	054034	005015					
9635	054036	042527	046114	040455	.ASCII	/WELL-ALIGNED DRIVE, PRIOR TO TESTING./<15><12>	
9636	054044	044514	047107	042105			
9637	054052	042040	044522	042526			
9638	054060	020054	051120	047511			
9639	054066	020122	047524	052040			
9640	054074	051505	044524	043516			

H15

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR63A.P11 11-APR-77 14:38 TRAP TABLE

MACY11 27(1006) 12-APR-77 08:48 PAGE 189

SEQ 0188

9641	054102	006456	012	
9642	054105	124	042510	041440
9643	054112	046517	046120	052105
9644	054120	020105	052522	020116
9645	054126	044524	042515	044440
9646	054134	020123	026470	030061
9647	054142	046440	047111	020056
9648	054150	042520	020122	051104
9649	054156	053111	027105	005015
9650	054164	047524	040440	047502
9651	054172	052122	052040	051505
9652	054200	044524	043516	020054
9653	054206	054524	042520	041440
9654	054214	047117	051124	046117
9655	054222	041455	024040	041536
9656	054230	027051	005015	000012
9657				
9658				
9659				
9660				
9661				
9662				
9663		000001		

.ASCII /THE COMPLETE RUN TIME IS 8-10 MIN. PER DRIVE./<<15><12>

.ASCIZ /TO ABORT TESTING, TYPE CONTROL-C (↑C)./<<15><12><12>

.END

J15

MD-11-DZ60-A - RK06 DRIVE COMPATIBILITY PROGRAM

MACY11 27(1006) 12-APR-77 08:48 PAGE 192

SEQ 0190

DZ60A.P11

11-APR-77 14:38

CROSS REFERENCE TABLE -- USER SYMBOLS

APTS17=	000200	3482	8750#														
APTS40=	000100	8103	8711	8752#													
RSKSYS	013212	3619	3630	3635	3641#	3645	3646	3648	3653	3664	4949	5021	5654				
ASWREG=	000000	1560	1573														
AS2SP2	044546	6240	8944#														
ATESTN=	000000	1560	1564														
RUNIT =	000000	1560	1567														
RUSMR =	000000	1560	1574														
AVECT1=	120210	1463#	1560	1599	3521												
AVECT2=	000000	1560	1600														
AVSCOR	017452	4021	4090	4154	4474#												
A.ABNL	003046	2804#	3760#	3763#	4766#	4834#	4839#	5645#	5827#	5844#	5853#	6037#	6052#	6454#			
		6719#	6722#	6736#	6739#	6742#	7669										
A.CONT	003050	2805#	7679														
A.NORM	003044	2803#	4767#	6455#	6723#	7673											
BADDRV	011053	3332#	4278#	4279#	4283	4837#	4842#	4843#	4844	4853#	4854#	4855	4861#	4862#			
		4863	4883#	4884#	4885												
BADSEC=	001000	3179#	5569	5754	6117	6125	6661	6773									
BAD632	011606	3402#	4880														
BAI =	000020	2482#	7838														
BA16 =	000400	2458#															
BA17 =	001000	2459#															
BOSCHD	026140	5827	5244	5858#													
BOSECT	011557	3396#	5876														
BOSRCK	027176	5567	5752	6107#	6660												
BGNRTY	032714	6378	6733	6744	6766#												
BIT0 =	000001	1395#	2455	2472	2498	2520	2668	2812	2905	3716	3745	4819	5489	5538			
		5578	5765	6337	7953												
BIT00 =	000001	1385#	1395														
BIT01 =	000002	1384#	1394														
BIT02 =	000004	1383#	1393														
BIT03 =	000010	1382#	1392														
BIT04 =	000020	1381#	1391														
BIT05 =	000040	1380#	1390														
BIT06 =	000100	1379#	1389														
BIT07 =	000200	1378#	1388														
BIT08 =	000400	1377#	1387	5380													
BIT09 =	001000	1376#	1386	8557	8683												
BIT1 =	000002	1394#	2473	2500	2669	2813	2906	3171	6341								
BIT10 =	002000	1375#	2460	2489	2511	2546	2560	2573	2588	2602	2681	2822	3180	6370			
		8535															
BIT11 =	004000	1374#	2461	2490	2512	2531	2547	2561	2574	2589	2603	2682	3181				
BIT12 =	010000	1373#	2463	2491	2513	2548	2562	2575	2576	2590	2604	2683	3182	6252			
		6280															
BIT13 =	020000	1372#	2464	2492	2514	2532	2549	2563	2577	2591	2605	2684	8542				
BIT14 =	040000	1371#	2465	2493	2515	2533	2550	2564	2578	2592	2606	2685	2823	8667			
BIT15 =	100000	1370#	2466	2467	2494	2516	2534	2551	2611	2686	2824	3183	6373				
BIT2 =	000004	1393#	2474	2501	2502	2522	2670	2814	2907	3172	6345						
BIT3 =	000010	1392#	2480	2481	2503	2523	2671	2815	2908	3173	6352						
BIT4 =	000020	1391#	2475	2482	2504	2524	2525	2540	2582	2596	2672	2816	2909	3174			
		6356															
BIT5 =	000040	1390#	2483	2484	2505	2526	2541	2555	2568	2583	2597	2673	2817	3175			
		4638	6360	7750													
BIT6 =	000100	1389#	2456	2485	2506	2527	2542	2556	2569	2584	2598	2674	2819	3176			
		4643	4676	5605	5792												
BIT7 =	000200	1388#	2457	2486	2507	2528	2529	2543	2557	2570	2585	2599	2676	3177			

K15

MD-11-DZ60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZ60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 193
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0191

BIT8 = 000400	3519 1387#	3713 2458	3726 2487	4982 2508	2509	2530	2544	2558	2571	2586	2600	2677	2820
BIT9 = 001000	3178 1386#	6364 2459	2488	2510	2545	2559	2572	2587	2601	2678	2821	3179	4971
BKSLSH 011707	6367 3414#	8647 5271	5279										
BIDEXH 033600	6607 1402#	6637	6920#										
BPTVEC= 000014	2507#	5858	6652	7103	7269	7505							
BSE = 000200	3171#	6040	6654	6668	6766								
BSERR = 000002	2915#	5826	5874	6108									
BSFACT 004152	2914#	5838	5907	6113	6115								
BSSOFT 003152	2920#	3559	3563	3565	3570	3581	3583	3585	3631	3634	3636	3649	3651
BUFF0 005202	3654 4719	3656 4723	3697 4938	4246* 4950	4247* 5010	4249* 5022	4251 5274	4254 5288*	4259 5293*	4264* 5294	4281* 5298*	4286 5298*	4290*
BUFPR1 005412	2923#												
B.BA16= 000001	2472#	7832											
B.BA17= 000002	2473#	7832											
B.CDT = 000004	2474#	7486	7832										
B.CFMT= 000020	2475#	4973	6876	7486	7832	7916							
CART 011510	5388#	5357											
CCLR = 100000	2467#	7439											
CDT = 002000	2460#	7245	7795	7852	7903	7926							
CERR = 100000	2466#	6616	6646	7141	7200	7259	7441	7492	7807	7863	7951		
CFMT = 010000	2463#	6899	6932	7245	7795	7852	7903	7926					
CKMEM1 025250	5729#												
CKMEM2 025262	5733#												
CKMO 025270	5732	5735#											
CKSCOR 022602	5159#	6043											
CLEAR = 000105	2426#												
CLR = 000040	2483#												
CLRPRM 020710	4771	4788#											
CMOTO = 000100	2674#	6501	6831	7002									
CMPIBUF 024202	5525#												
CMSTRT 012006	3438	3443#	6757										
CNTROY 011307	3363#	5659											
CNTRLC 011652	3408#	4642	5242										
CNTRLG 011676	3412#	4656	5260										
CNTRLR 011664	3410#	4652											
CNTRLU 011671	3411#	5254											
CNTRLZ 011657	3409#	4648	5248										
COE = 001000	2510#	6573	7103	7269	7505								
COMAVG 007266	3143#	4290	4532*	4544*	4545*	4572							
COMMA 011711	3415#	3609											
COMSTR 005166	2918#	6172	6181	6785									
CONCLR= 000176	2444#	7934											
CONERR 030134	2805	6333#											
CR = 000015	1310#	8147	8157										
CRLF = 000200	1311#	8118	8157										
CRTSER 023440	4981	5346#											
CRZLF 011703	3413#	6239	6251	6275									
CTLOUT 024722	4235	5632#	6153										
CTO = 004000	2461#	6536	7501										
CYLNDR 005424	2928#												
C.INIT 037356	6158	7712#											
C. RTN 041230	7816	7844	7872	7879	7882	7889	7993	7910	7932	7937	7947	7956	7969

M.COIF =	017760	2613#											
M.DPID	042032	4500	4565	8200#									
M.DPIM	041772	8171#											
M.DP01	042002	8174#	8183										
M.DP02	042022	8178	8182#										
M.DP40	042070	8211#	8232										
M.DP41	042124	8216	8223#										
M.DP42	042134	8222	8227#										
M.DP44	042166	8234	8239#										
M.DP50	042200	8209	8243#										
M.DRV =	000007	2610#											
M.HEAD=	007000	2617#											
M.ID =	000003	2612#											
M.PAR =	100000	2611#											
M.SECT=	000760	2616#											
M.SER =	077770	2615#											
NCSCRO	006750	3095#	4129	4131	4157	4174	4452	4487	4534				
NCSCR1	007010	3098#											
NCSCR2	007050	3101#											
NED =	010000	2491#	4586	6485	6495	7082	7160	7503	7873				
NEDHDL	020126	3760	4586#	4834									
NEM =	004000	2490#	6480	7097	7263	7503							
NEWMSG	011024	3327#	4709										
NEWON	005466	2942#	3761#	3764	4598*	4835*	4840						
NGOFST	005442	2935#											
NOCHK =	000400	2677#	7737*	7799	7840	7867	7885	7906	7929	7938	7972		
NOORLS	010422	3277#	3663										
NOORTS	011256	3358#											
NOOSC =	004000	2682#	6705	7325									
NOFALS	011577	3399#	5884										
NORTRY	003140	2896#	4770#	6771									
NOSCR0	006450	3073#	4004	4006	4230	4448	4482	4533	4550				
NOSCR1	006510	3076#											
NOSCR2	006550	3079#											
NOTMSG	053476	3497	9593#										
NOTYPE	003143	2900#	3495	3498#									
NSCOR0	007270	3148#	4065#	4363	4365	4387	4393	4435	4455	5189			
NSCOR1	007310	3151#	4066#	4437									
NSCOR2	007330	3154#	4067#	4439									
NSSCR0	007250	3118#	4050#	4158	4436*	4477							
NSSCR1	007252	3121#	4051#	4438*									
NSSCR2	007254	3124#	4052#	4440*									
NTREDY	011103	3337#	4856										
NOXDRIV	011064	3334#	4845										
NOXF =	000004	2502#											
ORVERG	010702	3309#	4282										
OCTBIN	041252	4724	8003#										
OFFSET=	000115	2430#	4404	4413	7782								
OFINUS	005426	2929#											
OFSCNT	003147	2904#	4379*	4398	4406*	5200							
OFSDIR	003146	2903#	4376*	4388	4395	4420*	4491*	4497	4506*	5190			
OFST =	000004	2522#											
ONSUBS	007724	3220#	3615*	3616									
OPCOMP	003121	2881#											
OPCONT	003130	2888#											
OPI =	020000	2514#	5858	6597	7103	7269	7505						

SW04 = 000020	1353#	1363			
SW05 = 000040	1352#	1362			
SW06 = 000100	1351#	1361			
SW07 = 000200	1350#	1360			
SW08 = 000400	1349#	1359			
SW09 = 001000	1348#	1358			
SW1 = 000002	1366#				
SW10 = 002000	1347#				
SW11 = 004000	1346#				
SW12 = 010000	1345#				
SW13 = 020000	1344#	6225			
SW14 = 040000	1343#				
SW15 = 100000	1342#				
SW2 = 000004	1365#				
SW3 = 000010	1364#				
SW4 = 000020	1363#				
SW5 = 000040	1362#				
SW6 = 000100	1361#				
SW7 = 000200	1360#				
SW8 = 000400	1359#				
SW9 = 001000	1358#				
SYSDRV 007653	3212#	3549#	3550		
S.ACLO= 000100	2569#				
S.BRAM= 000100	2584#				
S.BRKE= 040000	2606#				
S.CART= 000400	2586#				
S.DCLO= 010000	2576#				
S.DIB = 002000	2602#				
S.DOOR= 000200	2585#				
S.DRA = 000040	2555#				
S.DROT= 020000	2577#				
S.DRY = 000200	2557#	4851			
S.DSC = 040000	2564#	7175	7323	7340	7403
S.F.T = 000200	2570#	7318			
S.FORM= 001000	2559#				
S.FWD = 002000	2588#				
S.HDFL= 000200	2599#				
S.HDMM= 000040	2583#				
S.ICYL= 000040	2568#				
S.ILF = 000400	2571#				
S.LIND= 020000	2605#				
S.LOAD= 010000	2590#				
S.MHD = 000400	2600#				
S.MMOV= 010000	2604#				
S.OFF = 002000	2560#				
S.PAR = 001000	2572#	7178	7655		
S.PIP = 020000	2563#	7186	7381		
S.PLO = 004000	2603#				
S.REV = 004000	2589#				
S.RTZ = 020000	2591#				
S.SECT= 000020	2596#				
S.SKI = 002000	2573#				
S.SPIN= 010000	2562#				
S.SPLS= 010000	2575#				
S.SPOK= 001000	2587#				
S.TYPE= 000400	2558#				

\$ROOCT= ***** U	8838															
\$POSZ = 000000	8606#															
\$ QAO 001160	1503#															
\$ Q0 001162	1505#	6388	9312													
\$REG1 001164	1506#	9312														
\$REG10 001202	1513#	5559#	5560#	5561	5592*	5595	5744*	5745*	5746	5779*	5782	6613#	6643#			
	6682*	9316	9324													
\$REG11 001204	1514#	5593#	5780#	6614*	6644*	6683#	9316	9324								
\$ G12 001206	1515#	5594#	5781#	6615*	6645*	6676*	6678*	9316	9324							
\$ G13 001210	1516#	5596#	5598#	5600*	5601*	5783*	5785*	5787*	5788*	9316	9324					
\$ G14 001212	1517#	5595#	5597#	5599*	5782*	5784*	5786*	9319	9324							
\$ G15 001214	1518#	5602#	5603#	5789*	5790*	9319	9324									
\$ G16 001216	1519#	9320	9324													
\$ G17 001220	1520#	9320														
\$ G2 001166	1507#	9312														
\$ G20 001222	1521#	9320														
\$ G21 001224	1522#	9320														
\$ G22 001226	1523#	9320														
\$ G23 001230	1524#	9320														
\$ G24 001232	1525#	9320														
\$ G25 001234	1526#	9320														
\$ G26 001236	1527#	6259	6419													
\$ G27 001240	1528#															
\$REG3 001170	1508#	9312														
\$REG30 001242	1529#															
\$REG31 001244	1530#															
\$REG32 001246	1531#															
\$REG33 001250	1532#															
\$REG34 001252	1533#															
\$REG35 001254	1534#															
\$REG36 001256	1535#	6398*	9315													
\$REG37 001260	1536#	6399*	9315													
\$REG4 001172	1509#															
\$REG5 001174	1510#	5170	5574*	5582	5588*	5761*	5769	5775*	6207*	6208*	6395	6777*	6778*			
	6815	6887*	6909*	6921*	6935	6938	9316	9323								
\$REG6 001176	1511#	5193	5196	5575*	5584	5589*	5762*	5771	5776*	6888*	6891*	6904	6906*			
	6908*	6928*	6929*	6930*	6931*	6934*	6936	6937	9316	9323						
\$REG7 001200	1512#	5178	5576*	5586	5590*	5763*	5773	5777*	6889*	6890*	6894	6897*	6901*			
	6903*	6935*	6937*	6939*	9316	9323										
\$RESRE 044416	8787#	8839														
\$R2A = ***** U	8840															
\$SAVRE 044360	8771#	8838														
\$SCOPE 043742	3452	8666#														
\$SETUP= 000037	3429#	3451	3452	3454	3456	3458	3460	3461	3463	4329	8532	8557	8564			
	8574	8612	8667													
\$STUP = 177777	3429#															
\$SUPRS 042524	4261	4288	5334	5361	8360#											
\$SVLAD 044042	8676	8682	8688#													
\$SVPC = 000230	1434#	1439														
\$SMR = 163000	1284#	1286#	1291	1292	1293	1294	1295	1296	1297	1551	1552	3461	3463			
	3464	3730	3813	3853	3895	3940	3974	4046	4123	4220	4324	4330	4340			
	4346	4347	8523	8524	8525	8526	8527	8535	8542	8554	8557	8569	8660			
	8661	8662	8663	8667	8679	8681	8682	8683	8688	8691	8694	8697				
\$SAREG 001352	1573#	3484														
\$SAPRK= 000000	8663															
\$TESTN 001334	1564#	3790*	3813*	3853*	3895*	3940*	3974*	4046*	4123*	4220*	8689*					

DO1

MO-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM

MACY11 27(1006) 12-APR-77 08:48 PAGE 211

SEQ 0209

DZR60A.P11 11-APR-77 14:38 CROSS REFERENCE TABLE -- USER SYMBOLS

STKB	001146	1496#	4632	4674#	8572	8589	8595								
STKS	001144	1495#	4643#	4676#	8572	8587	8593								
STMP0	001262	1537#													
STMP1	001264	1538#	6072#	6073#	6081										
STMP10	001302	1545#	6085#	6086#	6087										
STMP11	001304	1546#	4160#	4162#	4166	4180#	4182#	4186	4250#	4265#	4280#	4292#	4477#	4482#	
		4487#	4496	4508#	4509#	4559#	4564	4571#	4572#	5095#	5101				
STMP12	001306	1547#	4166#	4167#	4168	4170#	4171	4186#	4187#	4188	4190#	4191	4230#	4234	
		4272#	4480#	4490#	4513#	4560#	4573#	5096#	5102						
STMP13	001310	1548#	4248#	4263#	4264	4478#	4483#	4488#	4499	4511#	4512#	5097#	5103		
STMP14	001312	1549#													
STMP15	001314	1550#													
STMP2	001266	1539#	6068#	6096											
STMP3	001270	1540#	6069#	6097											
STMP4	001272	1541#	6070#	6098											
STMP5	001274	1542#													
STMP6	001276	1543#													
STMP7	001300	1544#													
STN =	000012	1273#	1284	3777	3790#	3800	3813#	3843	3853#	3875	3895#	3926	3940#	3949	
		3974#	4026	4046#	4095	4123#	4199	4220#							
STMPWR	042440	8308	8309	8329#											
STP8	001152	1498#	8146#	8157											
STPFLG	001157	1502#	8095	8157											
STPS	001150	1497#	8144	8157											
STRAP	044454	3456	8807#												
STRAP2	044476	8818#	8829												
STRAP =	000012	8822#	8831#	8832#	8833#	8834#	8835#	8837	8838#	8839#	8840#	8841#			
STRAPD	044510	8812	8829#												
STSTM	001004	1459#													
STSTMH	001102	1475#	3492#	3540#	3641#	3706#	3907#	4304#	4329#	5650#	6247	8534	8569	8659	
		8688#	8689	8694	8697										
STYPBN=	##### U	8835													
STYPDS	043012	8462#	8834												
STYPE	041510	8095#	8728	8822	8830										
STYPEC	041722	8125	8132	8139	8144#	8145									
STYPEX	041770	8150	8152	8155#											
STYPOC	042610	8403#	8831												
STYPON	042624	8402	8405#	8833											
STYPOS	042564	8398#	8832												
SUNIT	001342	1567#													
SUNITH	001010	1461#													
SUSR	001354	1574#													
SVECT1	001400	1599#	3517												
SVECT2	001402	1600#													
SXSTR	043752	8670#													
SSET4=	000000	4340#													
S.S.SM=	##### U	2828	7014	7249	7440	7451	7489	7803	7844	7856	7910	7932	7950	7977	
		7986													
SOFILL	043007	8399#	8403#	8413	8448#										
S4OCAT=	##### U	8544	8667												
.	= 054236	1413#	1417#	1422#	1424#	1434	1435#	1437#	1439#	1440#	1446	1447#	1449#	1451#	
		1472#	1556	2914#	2915#	2920#	2936#	2943#	2944#	2946#	2951#	3012#	3013#	3014#	
		3015#	3016#	3017#	3018#	3019#	3020#	3021#	3022#	3023#	3024#	3025#	3026#	3027#	
		3073#	3076#	3079#	3084#	3087#	3090#	3095#	3098#	3101#	3106#	3109#	3112#	3148#	
		3151#	3154#	3159#	3162#	3165#	3449	3463	3464	4347	8157	8229	8287#	8349#	
		8516#	8569	8572	8612	8620	8637#	8697	8749#	9591#					

E01

MD-11-DZR60-A - RK06 DRIVE COMPATIBILITY PROGRAM
DZR60A.P11 11-APR-77 14:38

MACY11 27(1006) 12-APR-77 08:48 PAGE 212
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0210

.SASTA= ***** U	8701	8704
.SX = 001000	14468	1451
..HIGH 000226	14288	
..LOW 000224	14278	

SSOECBN	18														
SSOCTBN	18	7986													
SSSCHRE	1466#	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518
	1519	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533
	1534	1535	1536												
SSSCHTH	1466#	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550
SSSESCA	1410#														
SSNEWT	1410#	3777	3800	3843	3875	3926	3949	4026	4095	4199					
SSSET	8822#	8831	8832	8833	8834	8837	8838	8839	8840						
SSSETH	3481#														
SSSKIP	1410#														
.EQUAT	1259#	1300													
.HEADE	1259#	1274													
.SETUP	1259#	3429													
.SWRHI	1259#	1287													
.SWRLO	1259#	1298#													
.SACTI	1259#	1430													
.SAPT8	1557#														
.SAPTH	1259#	1441													
.SAPTY	1259#	8697													
.SCATC	1259#	1411													
.SCHTA	1259#	1466													
.SOB20	1259#	8288													
.SOB20	1259#	8249													
.SEOP	1259#	4320													
.SERRO	1259#	8517													
.SPOWE	1259#														
.SRAND	1259#	8042													
.SREAO	1259#	8569													
.SSAVE	1259#	8754													
.SSCOP	1259#	8654													
.SSIZE	1259#														
.SSUPR	1259#	8350													
.STRAP	1259#	8799													
.STYPD	1259#	8450													
.STYPE	1259#	8078													
.STYPO	1259#	8373													

. ABS. 054236 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DZR60A, DZR60A, SEQ/SOL/CRF/NL: TOC/DOC=DRIV10.P11/EQ: QNEWSW, DZR60A.P11

RUN-TIME: 31 27 2 SECONDS
RUN-TIME RATIO: 218/61=3.5
CORE USED: 48K (96 PAGES)

DOCUMENT PAGES: 212