

RK611
RK06, RK07

RK6 DR CPT PROG
CZR6QC0

AH-B162C-MC
FICHE 1 OF 2

NOV 1980
COPYRIGHT © 77-80
MADE IN USA



A large grid of approximately 15 columns and 20 rows of microfiche frames. Each frame contains a small, high-contrast image of a document page, likely a technical drawing or a data table. The text within the frames is too small to be legible, but the overall layout suggests a comprehensive set of data or instructions.

RK611
RK06 RK07

RK6 DR CPT PROG
CZR60C0

AH-B162C-MC
FICHE 2 OF 2

NOV 1980
COPYRIGHT © 77-80
MADE IN USA



.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-B161C-MC
PRODUCT NAME: CZR6QC0 RK6 DR CPT PROG
DATE: JUNE 1980
MAINTAINER: DIAGNOSTIC GROUP

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.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.
DIGITAL.

COPYRIGHT (C) 1977,1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

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

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

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

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

1.0 INTRODUCTION

THE PURPOSE OF THIS PROGRAM IS TO VERIFY THE COMPATIBILITY OF UP TO 16 RK06 OR RK07 DRIVES WHICH MAY RESIDE ON ONE OR MORE RK611/RK06-07 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.

NOTE: RK06 AND RK07 DRIVES CANNOT BE MIXED FOR COMPATIBILITY TESTING.

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.

167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE RK06-07 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
RK611 CONTROLLER
1 TO 8 RK06 OR RK07 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 CZR6A-CZR6E AND CZR6K, THE DRIVE DIAGNOSTICS CZR6H-CZR6J, AND THE SUBSYSTEM VERIFICATION PROGRAMS CZR6M-CZR6N. IN ADDITION, THE CARTRIDGE MUST BE FORMATTED IN 20 OR 22 SECTOR FORMAT USING THE PACK FORMATTER CZR6L, 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.

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

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.

270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325

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

326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380

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.

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
434
435
436

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:

CZR6QB0 - RK06-RK07 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:

DRIVE TYPE 6<CR> FOR RK06, 7<CR> FOR RK07:

THE OPERATOR MUST TYPE EITHER A 6<CR> OR A 7<CR>

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:

CZR6QB0 - RK06-RK07 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) :

DRIVE TYPE 6<CR> FOR RK06, 7<CR> FOR RK07:

THE OPERATOR MUST TYPE IN 6<CR> OR 7<CR>

SUBSYS A DRIVE(S) =

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

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
490
491
492

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:

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:

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
546
547
548

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

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 :

CZR6QB0 - RK06-RK07 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 PDP-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:

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
602
603
604

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:

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.

605
606
607
608
609
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

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 **

(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

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.

661
662
663
664
665
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
706
707
708
709
710
711
712
713
714
715
716

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

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.

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
762
763
764
765
766
767
768
769
770
771
772

9.2 DESCRIPTION OF PASS 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

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
804
805
806
807
808
809
810
811
812
813

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.

814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869

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

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
919
920
921
922
923
924
925

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

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

- | | |
|-----|---|
| 926 | 20. DRIVE OFF TRACK |
| 927 | |
| 928 | 21. DRIVE TIMING ERROR |
| 929 | |
| 930 | |
| 931 | |
| 932 | 22. DATA LATE ERROR |
| 933 | |
| 934 | 23. CONTROLLER TIMEOUT ERROR |
| 935 | |
| 936 | 24. OPERATION INCOMPLETE ERROR |
| 937 | |
| 938 | 25. HEADER VRC ERROR |
| 939 | |
| 940 | 26. DATA CHECK ERROR |
| 941 | |
| 942 | 27. WRITE CHECK ERROR |
| 943 | |
| 944 | 28. DATA MISCOMPARE |
| 945 | |
| 946 | 29. NO DRIVE RESPONSE - UFE AND NXD |
| 947 | |
| 948 | 30. DRIVE ERROR WILL NOT CLEAR |
| 949 | |
| 950 | 31. DRIVE STATUS CHANGE WILL NOT CLEAR |
| 951 | |
| 952 | 32. ATTENTION BUT NO STATUS CHANGE OR FAULT |
| 953 | |
| 954 | 33. ATTENTION BUT DRIVE NOT AVAILABLE |
| 955 | |
| 956 | 34. ERROR WHILE GATHERING DRIVE STATUS |
| 957 | |
| 958 | 35. MULTIPLE DRIVE SELECT |
| 959 | |
| 960 | 36. HEADER COMPARE ERROR |
| 961 | |
| 962 | 37. ERROR IN RECALIBRATE FOR RECOVERY |
| 963 | |
| 964 | 38. CLEAR CONTROLLER DID NOT CLEAR ERROR |
| 965 | |
| 966 | 39. NO ATTENTION IN ATTENTION SUMMARY REGISTER |
| 967 | |
| 968 | 40. UNSOLICITED ATTENTION |
| 969 | |
| 970 | 41. UNEXPECTED DATA TYPE ERROR |
| 971 | |
| 972 | 42. ATTENTION DID NOT RESET WITH CLEAR |
| 973 | |
| 974 | 43. SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION |
| 975 | |
| 976 | 44. DATA LATE WHEN UNLOADING HEADER |
| 977 | |
| 978 | 45. CONTROLLER ERROR WHEN DRIVER SERVICING |
| 979 | |
| 980 | 46. RETRY UNSUCCESSFUL |
| 981 | |

47. BAD SECTOR ERROR ON SECTOR NOT LISTED BAD

982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018

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
000000	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

1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058

TABLE A

BASIC READ/WRITE TEST SECTORS

ADDRESS OF SECTOR ON EACH SURFACE (IN OCTAL)

DRIVE NO. -----	CYLINDER ----- RK06/RK07 -----	SECTOR -----
0	620/1444	0
1	620/1444	2
2	620/1444	4
3	620/1444	6
4	620/1444	10
5	620/1444	12
6	620/1444	14
7	620/1444	16
10	622/1446	0
11	622/1446	2
12	622/1446	4
13	622/1446	6
14	622/1446	10
15	622/1446	12
16	622/1446	14
17	622/1446	16

1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093

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

1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130

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	RK06
2 - 100-177	100-117	160-177	3	RK06
3 - 200-277	200-217	260-277	7	RK06
4 - 300-377	300-317	360-377	13	RK06
5 - 400-477	400-417	460-477	17	RK06
6 - 500-577	500-517	560-577	22	RK06
7 - 600-632	600-617	---	0	RK06
1 - CYL 0-177	CYL 0-17	CYL 160-177	SECT 0	RK07
2 - 200-377	200-217	360-377	3	RK07
3 - 400-577	400-417	560-577	7	RK07
4 - 600-777	600-617	760-777	13	RK07
5 - 1000-1177	1000-1017	1160-1177	17	RK07
6 - 1200-1377	1200-1217	1360-1377	22	RK07
7 - 1400-1456	1400-1417	---	0	RK07

1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170

TABLE D

 BASIC CYLINDER BLOCK LAYOUT EXAMPLE

(SHOWN FOR RK06, USE CYL 160-177 FOR RK07)

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	3	4	5	6	7	10	11	12	13	14	15	16	17	0	1	2
63	6	7	10	11	12	13	14	15	16	17	0	1	2	3	4	5
64	12	13	14	15	16	17	0	1	2	3	4	5	6	7	10	11
65	17	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16
66	5	6	7	10	11	12	13	14	15	16	17	0	1	2	3	4
67	14	15	16	17	0	1	2	3	4	5	6	7	10	11	12	13
70	4	5	6	7	10	11	12	13	14	15	16	17	0	1	2	3
71	15	16	17	0	1	2	3	4	5	6	7	10	11	12	13	14
72	7	10	11	12	13	14	15	16	17	0	1	2	3	4	5	6
73	2	3	4	5	6	7	10	11	12	13	14	15	16	17	0	1
74	16	17	0	1	2	3	4	5	6	7	10	11	12	13	14	15
75	13	14	15	16	17	0	1	2	3	4	5	6	7	10	11	12
76	11	12	13	14	15	16	17	0	1	2	3	4	5	6	7	10
77	10	11	12	13	14	15	16	17	0	1	2	3	4	5	6	7

1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209

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

NOTE: THE ABOVE TABLE GIVES CYLINDER NUMBERS FOR THE RK06.
 REFER FOR TABLE C TO OBTAIN CORRESPONDING CYLINDER
 NUMBERS FOR AN RK07.

1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248

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
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

NOTE: THE ABOVE TABLE GIVES CYLINDER NUMBERS FOR THE RK06.
REFER TO TABLE C TO OBTAIN CORRESPONDING CYLINDER
NUMBERS FOR THE RK07.

1249
1250
1251
1252
1253
1254
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

TABLE G

PSEUDO-RANDOM DATA PATTERN

<u>WORD #</u>	<u>DATA (OCTAL)</u>
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

a

1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340

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

*** REV 003 ***

.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

000001

\$TN=1
.TITLE CZR6QC0 RK6 DR CPT PROG
*COPYRIGHT (C) 1977,1980
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*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

160000

163000

\$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

001100

*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

000011

*MISCELLANEOUS DEFINITIONS
HT= 11 ;;CODE FOR HORIZONTAL TAB

1341	000012	LF=	12	::CODE FOR LINE FEED
1342	000015	CR=	15	::CODE FOR CARRIAGE RETURN
1343	000200	CRLF=	200	::CODE FOR CARRIAGE RETURN-LINE FEED
1344	177776	PS=	177776	::PROCESSOR STATUS WORD
1345		.EQUIV	PS,PSW	
1346	177774	STKLMT=	177774	::STACK LIMIT REGISTER
1347	177772	PIRQ=	177772	::PROGRAM INTERRUPT REQUEST REGISTER
1348	177570	DSWR=	177570	::HARDWARE SWITCH REGISTER
1349	177570	DDISP=	177570	::HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

1351		R0=	%0	::GENERAL REGISTER
1352	000000	R1=	%1	::GENERAL REGISTER
1353	000001	R2=	%2	::GENERAL REGISTER
1354	000002	R3=	%3	::GENERAL REGISTER
1355	000003	R4=	%4	::GENERAL REGISTER
1356	000004	R5=	%5	::GENERAL REGISTER
1357	000005	R6=	%6	::GENERAL REGISTER
1358	000006	R7=	%7	::GENERAL REGISTER
1359	000007	SP=	%6	::STACK POINTER
1360	000006	PC=	%7	::PROGRAM COUNTER
1361	000007			

;*PRIORITY LEVEL DEFINITIONS

1362		PR0=	0	::PRIORITY LEVEL 0
1363		PR1=	40	::PRIORITY LEVEL 1
1364	000000	PR2=	100	::PRIORITY LEVEL 2
1365	000040	PR3=	140	::PRIORITY LEVEL 3
1366	000100	PR4=	200	::PRIORITY LEVEL 4
1367	000140	PR5=	240	::PRIORITY LEVEL 5
1368	000200	PR6=	300	::PRIORITY LEVEL 6
1369	000240	PR7=	340	::PRIORITY LEVEL 7
1370	000300			
1371	000340			

;'SWITCH REGISTER' SWITCH DEFINITIONS

1372		SW15=	100000	
1373		SW14=	40000	
1374	100000	SW13=	20000	
1375	040000	SW12=	10000	
1376	020000	SW11=	4000	
1377	010000	SW10=	2000	
1378	004000	SW09=	1000	
1379	002000	SW08=	400	
1380	001000	SW07=	200	
1381	000400	SW06=	100	
1382	000200	SW05=	40	
1383	000100	SW04=	20	
1384	000040	SW03=	10	
1385	000020	SW02=	4	
1386	000010	SW01=	2	
1387	000004	SW00=	1	
1388	000002	.EQUIV	SW09,SW9	
1389	000001	.EQUIV	SW08,SW8	
1390		.EQUIV	SW07,SW7	
1391		.EQUIV	SW06,SW6	
1392		.EQUIV	SW05,SW5	
1393		.EQUIV	SW04,SW4	
1394		.EQUIV	SW03,SW3	
1395				
1396				

```
1397 .EQUIV SW02,SW2
1398 .EQUIV SW01,SW1
1399 .EQUIV SW00,SW0
1400
1401 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1402 100000 BIT15= 100000
1403 040000 BIT14= 40000
1404 020000 BIT13= 20000
1405 010000 BIT12= 10000
1406 004000 BIT11= 4000
1407 002000 BIT10= 2000
1408 001000 BIT09= 1000
1409 000400 BIT08= 400
1410 000200 BIT07= 200
1411 000100 BIT06= 100
1412 000040 BIT05= 40
1413 000020 BIT04= 20
1414 000010 BIT03= 10
1415 000004 BIT02= 4
1416 000002 BIT01= 2
1417 000001 BIT00= 1
1418 .EQUIV BIT09,BIT9
1419 .EQUIV BIT08,BIT8
1420 .EQUIV BIT07,BIT7
1421 .EQUIV BIT06,BIT6
1422 .EQUIV BIT05,BIT5
1423 .EQUIV BIT04,BIT4
1424 .EQUIV BIT03,BIT3
1425 .EQUIV BIT02,BIT2
1426 .EQUIV BIT01,BIT1
1427 .EQUIV BIT00,BIT0
1428
1429 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1430 000004 ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
1431 000010 RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
1432 000014 TBITVEC=14 ;;"T" BIT
1433 000014 TRTVEC= 14 ;;TRACE TRAP
1434 000014 BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
1435 000020 IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
1436 000024 PWRVEC= 24 ;;POWER FAIL
1437 000030 EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
1438 000034 TRAPVEC=34 ;;"TRAP" TRAP
1439 000060 TKVEC= 60 ;;TTY KEYBOARD VECTOR
1440 000064 TPVEC= 64 ;;TTY PRINTER VECTOR
1441 000240 PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR
1442
1443 .SBTTL TRAP CATCHER
1444
1445 000000 .=0
1446 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1447 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1448 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1449 000174 .=174
1450 000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
1451 000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
1452 .SBTTL STARTING ADDRESS(ES)
```



```

1453 000200 000137 012146      JMP      @#DFSTRT      ;; JUMP TO STARTING ADDRESS OF PROGRAM
1454      000204      . =204
1455 000204 000137 012154      JMP      @#P1STRT
1456      000220      . =220
1457 000220 000137 012172      JMP      @#P2STRT
1458
1459 000224 000700      ..LOW:  .WORD 700
1460 000226 001100      ..HIGH: .WORD 1100
1461
1462      .SBTTL ACT11 HOOKS
1463
1464      ;;*****
1465      :HOOKS REQUIRED BY ACT11
1466      $SVPC=      ;SAVE PC
1467      . =46
1468 000046 017234      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
1469      . =52
1470 000052 140000      .WORD 140000      ;;2)SET LOC.52 TO 140000
1471      .=$SVPC      ;; RESTORE PC
1472      . =1000
1473      .SBTTL APT PARAMETER BLOCK
1474
1475      ;;*****
1476      :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1477      ;;*****
1478      . $X=      ;;SAVE CURRENT LOCATION
1479      . =24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1480 000024 000200      200      ;;FOR APT START UP
1481      . =44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
1482 000044 001000      $APTHDR    ;;POINT TO APT HEADER BLOCK
1483      . = $X      ;;RESET LOCATION COUNTER
1484      ;;*****
1485      :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1486      :INTERFACE SPEC.
1487
1488 001000      $APTHD:
1489 001000 000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1490 001002 001330      $MBADR: .WORD $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
1491 001004 001320      $TSTM:  .WORD 1320    ;;RUN TIM OF LONGEST TEST
1492 001006 001546      $PASTM: .WORD 1546    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
1493 001010 001546      $UNITM: .WORD 1546    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
1494 001012 000030      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
1495      120210      AVECT1=120210      ;RKVEC=210, RKPRI=5
1496      177440      ABASE=177440      ;RKBAS ADRS
1497      000377      ADEVM=000377      ;SET DEVICES 0-7 IN MAP
  
```

```

1498          .SBTTL COMMON TAGS
1499
1500          ::*****
1501          ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
1502          ::*USED IN THE PROGRAM.
1503
1504          001100          .=1100
1505          001100          $CMTAG:          ;;START OF COMMON TAGS
1506          001100          000000          .WORD          0          ;;CONTAINS THE TEST NUMBER
1507          001102          000          $TSTNM: .BYTE          0          ;;CONTAINS ERROR FLAG
1508          001103          000          $ERFLG: .BYTE          0          ;;CONTAINS SUBTEST ITERATION COUNT
1509          001104          000000          $ICNT:  .WORD          0          ;;CONTAINS SCOPE LOOP ADDRESS
1510          001106          000000          $LPADR: .WORD          0          ;;CONTAINS SCOPE RETURN FOR ERRORS
1511          001110          000000          $LPERR: .WORD          0          ;;CONTAINS TOTAL ERRORS DETECTED
1512          001112          000000          $ERTTL: .WORD          0          ;;CONTAINS ITEM CONTROL BYTE
1513          001114          000          $ITEMB: .BYTE          0          ;;CONTAINS MAX. ERRORS PER TEST
1514          001115          001          $ERMAX: .BYTE          1          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
1515          001116          000000          $ERRPC: .WORD          0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
1516          001120          000000          $GDADR: .WORD          0          ;;CONTAINS ADDRESS OF 'BAD' DATA
1517          001122          000000          $BDADR: .WORD          0          ;;CONTAINS 'GOOD' DATA
1518          001124          000000          $GDDAT: .WORD          0          ;;CONTAINS 'BAD' DATA
1519          001126          000000          $BDDAT: .WORD          0          ;;RESERVED--NOT TO BE USED
1520          001130          000000          .WORD          0
1521          001132          000000          .WORD          0
1522          001134          000          $AUTOB: .BYTE          0          ;;AUTOMATIC MODE INDICATOR
1523          001135          000          $INTAG: .BYTE          0          ;;INTERRUPT MODE INDICATOR
1524          001136          000000          .WORD          0
1525          001140          177570          $SWR:   .WORD          DSWR          ;;ADDRESS OF SWITCH REGISTER
1526          001142          177570          $DISPLAY: .WORD          DDISP          ;;ADDRESS OF DISPLAY REGISTER
1527          001144          177560          $TKS:   177560          ;;TTY KBD STATUS
1528          001146          177562          $TKB:   177562          ;;TTY KBD BUFFER
1529          001150          177564          $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
1530          001152          177566          $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
1531          001154          000          $NULL:  .BYTE          0          ;;CONTAINS NULL CHARACTER FOR FILLS
1532          001155          002          $FILLS: .BYTE          2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1533          001156          012          $FILLC: .BYTE          12         ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
1534          001157          000          $TPFLG: .BYTE          0          ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
1535          001160          000000          $REGAD: .WORD          0          ;;CONTAINS THE ADDRESS FROM
1536          001162          000000          $REG0:  .WORD          0          ;;WHICH ($REG0) WAS OBTAINED
1537          001164          000000          $REG1:  .WORD          0          ;;CONTAINS (($REGAD)+0)
1538          001166          000000          $REG2:  .WORD          0          ;;CONTAINS (($REGAD)+2)
1539          001170          000000          $REG3:  .WORD          0          ;;CONTAINS (($REGAD)+4)
1540          001172          000000          $REG4:  .WORD          0          ;;CONTAINS (($REGAD)+6)
1541          001174          000000          $REG5:  .WORD          0          ;;CONTAINS (($REGAD)+10)
1542          001176          000000          $REG6:  .WORD          0          ;;CONTAINS (($REGAD)+12)
1543          001200          000000          $REG7:  .WORD          0          ;;CONTAINS (($REGAD)+14)
1544          001202          000000          $REG10: .WORD          0          ;;CONTAINS (($REGAD)+16)
1545          001204          000000          $REG11: .WORD          0          ;;CONTAINS (($REGAD)+20)
1546          001206          000000          $REG12: .WORD          0          ;;CONTAINS (($REGAD)+22)
1547          001210          000000          $REG13: .WORD          0          ;;CONTAINS (($REGAD)+24)
1548          001212          000000          $REG14: .WORD          0          ;;CONTAINS (($REGAD)+26)
1549          001214          000000          $REG15: .WORD          0          ;;CONTAINS (($REGAD)+30)
1550          001216          000000          $REG16: .WORD          0          ;;CONTAINS (($REGAD)+32)
1551          001220          000000          $REG17: .WORD          0          ;;CONTAINS (($REGAD)+34)
1552          001222          000000          $REG20: .WORD          0          ;;CONTAINS (($REGAD)+36)
1553

```

```
1554 001224 000000 $REG21: .WORD 0 ;;CONTAINS (($REGAD)+42)
1555 001226 000000 $REG22: .WORD 0 ;;CONTAINS (($REGAD)+44)
1556 001230 000000 $REG23: .WORD 0 ;;CONTAINS (($REGAD)+46)
1557 001232 000000 $REG24: .WORD 0 ;;CONTAINS (($REGAD)+50)
1558 001234 000000 $REG25: .WORD 0 ;;CONTAINS (($REGAD)+52)
1559 001236 000000 $REG26: .WORD 0 ;;CONTAINS (($REGAD)+54)
1560 001240 000000 $REG27: .WORD 0 ;;CONTAINS (($REGAD)+56)
1561 001242 000000 $REG30: .WORD 0 ;;CONTAINS (($REGAD)+60)
1562 001244 000000 $REG31: .WORD 0 ;;CONTAINS (($REGAD)+62)
1563 001246 000000 $REG32: .WORD 0 ;;CONTAINS (($REGAD)+64)
1564 001250 000000 $REG33: .WORD 0 ;;CONTAINS (($REGAD)+66)
1565 001252 000000 $REG34: .WORD 0 ;;CONTAINS (($REGAD)+70)
1566 001254 000000 $REG35: .WORD 0 ;;CONTAINS (($REGAD)+72)
1567 001256 000000 $REG36: .WORD 0 ;;CONTAINS (($REGAD)+74)
1568 001260 000000 $REG37: .WORD 0 ;;CONTAINS (($REGAD)+76)
1569 001262 000000 $TMP0: .WORD 0 ;;USER DEFINED
1570 001264 000000 $TMP1: .WORD 0 ;;USER DEFINED
1571 001266 000000 $TMP2: .WORD 0 ;;USER DEFINED
1572 001270 000000 $TMP3: .WORD 0 ;;USER DEFINED
1573 001272 000000 $TMP4: .WORD 0 ;;USER DEFINED
1574 001274 000000 $TMP5: .WORD 0 ;;USER DEFINED
1575 001276 000000 $TMP6: .WORD 0 ;;USER DEFINED
1576 001300 000000 $TMP7: .WORD 0 ;;USER DEFINED
1577 001302 000000 $TMP10: .WORD 0 ;;USER DEFINED
1578 001304 000000 $TMP11: .WORD 0 ;;USER DEFINED
1579 001306 000000 $TMP12: .WORD 0 ;;USER DEFINED
1580 001310 000000 $TMP13: .WORD 0 ;;USER DEFINED
1581 001312 000000 $TMP14: .WORD 0 ;;USER DEFINED
1582 001314 000000 $TMP15: .WORD 0 ;;USER DEFINED
1583 001316 000000 $ESCAPE:0 ;;ESCAPE ON ERROR ADDRESS
1584 001320 177607 000377 $BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
1585 001324 077 $QUES: .ASCII /?/ ;;QUESTION MARK
1586 001325 015 $CRLF: .ASCII <15> ;;CARRIAGE RETURN
1587 001326 000012 $LF: .ASCIZ <12> ;;LINE FEED
1588 *****
1589 .SBTTL APT MAILBOX-ETABLE
1590
1591 *****
1592 .EVEN
1593 001330 $MAIL: ;;APT MAILBOX
1594 001330 000000 $MSGTY: .WORD AMSGTY ;;MESSAGE TYPE CODE
1595 001332 000000 $FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
1596 001334 000000 $TESTN: .WORD ATESTN ;;TEST NUMBER
1597 001336 000000 $PASS: .WORD APASS ;;PASS COUNT
1598 001340 000000 $DEVCT: .WORD ADEVCT ;;DEVICE COUNT
1599 001342 000000 $UNIT: .WORD AUNIT ;;I/O UNIT NUMBER
1600 001344 000000 $MSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS
1601 001346 000000 $MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
1602 001350 $ETABLE: ;;APT ENVIRONMENT TABLE
1603 001350 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
1604 001351 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
1605 001352 000000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
1606 001354 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
1607 001356 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
1608 ;;
1609 ;; BITS 15-11=CPU TYPE
;; 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
```

1610			.*			11/70=06,PDQ=07,Q=10
1611			.*			BIT 10=REAL TIME CLOCK
1612			.*			BIT 9=FLOATING POINT PROCESSOR
1613			.*			BIT 8=MEMORY MANAGEMENT
1614	001360	000	.*	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
1615	001361	000	.*	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
1616			.*			MEM.TYPE BYTE -- (HIGH BYTE)
1617			.*			900 NSEC CORE=001
1618			.*			300 NSEC BIPOLAR=002
1619			.*			500 NSEC MOS=003
1620	001362	000000	.*	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
1621			.*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
1622	001364	000	.*	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
1623	001365	000	.*	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
1624	001366	000000	.*	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
1625	001370	000	.*	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
1626	001371	000	.*	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
1627	001372	000000	.*	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
1628	001374	000	.*	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
1629	001375	000	.*	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
1630	001376	000000	.*	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
1631	001400	120210	.*	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
1632	001402	000000	.*	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
1633	001404	177440	.*	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
1634	001406	000377	.*	\$DEVN: .WORD	ADEVN	::DEVICE MAP
1635	001410		.*	\$ETEND:		
1636			.*	.MEXIT		

1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692

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

\$ERRTB:
 ;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

001410
001410 045405
001412 050500
001414 052264
001416 052374

001420 045431
001422 050500
001424 052264
001426 052374

001430 045455
001432 050500
001434 052264
001436 052374

001440 045500
001442 050500
001444 052264
001446 052374

001450 045521
001452 050500
001454 052264
001456 052424

001460 045540
001462 050500
001464 052264
001466 052460

001470 045564
001472 050500
001474 052264

1693	001476	052460	DF03	
1694				
1695			:ERROR 10	
1696	001500	045620	EM10	:AC LOW
1697	001502	050500	DH100	
1698	001504	052264	DT100	
1699	001506	052424	DF02	
1700				
1701			:ERROR 11	
1702	001510	045627	EM11	:SPEED LOSS
1703	001512	050500	DH100	
1704	001514	052264	DT100	
1705	001516	052424	DF02	
1706				
1707			:ERROR 12	
1708	001520	045642	EM12	:ILLEGAL FUNCTION
1709	001522	050500	DH100	
1710	001524	052264	DT100	
1711	001526	052424	DF02	
1712				
1713			:ERROR 13	
1714	001530	045663	EM13	:PROGRAMMING ERROR
1715	001532	050500	DH100	
1716	001534	052264	DT100	
1717	001536	052374	DF01	
1718				
1719			:ERROR 14	:NON-EXISTANT FUNCTION
1720	001540	045705	EM14	
1721	001542	050500	DH100	
1722	001544	052264	DT100	
1723	001546	052424	DF02	
1724				
1725			:ERROR 15	
1726	001550	045733	EM15	:DRIVE TYPE ERROR
1727	001552	050500	DH100	
1728	001554	052264	DT100	
1729	001556	052424	DF02	
1730				
1731			:ERROR 16	
1732	001560	045754	EM16	:FORMAT ERROR
1733	001562	050500	DH100	
1734	001564	052264	DT100	
1735	001566	052424	DF02	
1736				
1737			:ERROR 17	
1738	001570	045771	EM17	:WRITE LOCK ERROR
1739	001572	050500	DH100	
1740	001574	052264	DT100	
1741	001576	052424	DF02	
1742				
1743			:ERROR 20	
1744	001600	046012	EM20	:DRIVE UNSAFE
1745	001602	050500	DH100	
1746	001604	052264	DT100	
1747	001606	052424	DF02	
1748				

1749			:ERROR 21	
1750	001610	046027	EM21	:SEEK INCOMPLETE
1751	001612	050500	DH100	
1752	001614	052264	DT100	
1753	001616	052424	DF02	
1754				
1755			:ERROR 22	
1756	001620	046047	EM22	:CYLINDER OVERFLOW
1757	001622	050500	DH100	
1758	001624	052264	DT100	
1759	001626	052424	DF02	
1760				
1761			:ERROR 23	
1762	001630	046071	EM23	:ILLEGAL CYLINDER
1763	001632	050500	DH100	
1764	001634	052264	DT100	
1765	001636	052424	DF02	
1766				
1767			:ERROR 24	
1768	001640	046122	EM24	:DRIVE OFF TRACK
1769	001642	050500	DH100	
1770	001644	052264	DT100	
1771	001646	052424	DF02	
1772				
1773			:ERROR 25	
1774	001650	046142	EM25	:DRIVE TIMING ERROR
1775	001652	050500	DH100	
1776	001654	052264	DT100	
1777	001656	052424	DF02	
1778				
1779			:ERROR 26	
1780	001660	046165	EM26	:DATA LATE
1781	001662	050500	DH100	
1782	001664	052264	DT100	
1783	001666	052424	DF02	
1784				
1785			:ERROR 27	
1786	001670	046177	EM27	:CONTROLLER TIMEOUT
1787	001672	050500	DH100	
1788	001674	052264	DT100	
1789	001676	052424	DF02	
1790				
1791			:ERROR 30	
1792	001700	046222	EM30	:OPERATION INCOMPLETE
1793	001702	050500	DH100	
1794	001704	052264	DT100	
1795	001706	052534	DF05	
1796				
1797			:ERROR 31	
1798	001710	046247	EM31	:HEADER VRC ERROR
1799	001712	050500	DH100	
1800	001714	052264	DT100	
1801	001716	052534	DF05	
1802				
1803			:ERROR 32	
1804	001720	046270	EM32	:DATA CHECK ERROR

1805	001722	050500	DH100	
1806	001724	052264	DT100	
1807	001726	052570	DF07	
1808				
1809			:ERROR 33	
1810	001730	046311	EM33	:WRITE CHECK ERROR
1811	001732	050500	DH100	
1812	001734	052264	DT100	
1813	001736	052624	DF10	
1814				
1815			:ERROR 34	
1816	001740	046333	EM34	:DATA MISCOMPARE(S)
1817	001742	050500	DH100	
1818	001744	052264	DT100	
1819	001746	052520	DF04	
1820				
1821			:ERROR 35	
1822	001750	046353	EM35	:NO DRIVE RESPONSE-UFE & NXD
1823	001752	050500	DH100	
1824	001754	052264	DT100	
1825	001756	052374	DF01	
1826				
1827			:ERROR 36	
1828	001760	046407	EM36	:DRIVE ERROR WILL NOT CLEAR
1829	001762	000000	0	
1830	001764	000000	0	
1831	001766	000000	0	
1832				
1833			:ERROR 37	
1834	001770	046442	EM37	:DRIVE STATUS CHANGE WILL NOT CLEAR
1835	001772	000000	0	
1836	001774	000000	0	
1837	001776	000000	0	
1838				
1839			:ERROR 40	
1840	002000	046505	EM40	:ATTENTION BUT NO STATUS CHANGE OR FAULT
1841	002002	050500	DH100	
1842	002004	052264	DT100	
1843	002006	052424	DF02	
1844				
1845			:ERROR 41	
1846	002010	046551	EM41	:ATTENTION BUT DRIVE NOT AVAILABLE
1847	002012	050500	DH100	
1848	002014	052264	DT100	
1849	002016	052424	DF02	
1850				
1851			:ERROR 42	
1852	002020	046607	EM42	:ATTENTION WHEN NOT EXPECTED
1853	002022	050500	DH100	
1854	002024	052264	DT100	
1855	002026	052424	DF02	
1856				
1857			:ERROR 43	
1858	002030	046637	EM43	:ERROR WHILE GATHERING DRIVE STATUS
1859	002032	050500	DH100	
1860	002034	052264	DT100	

1861	002036	052670	DF12	
1862				
1863			:ERROR 44	
1864	002040	047114	EM63	:CLEAR CONTROLLER DID NOT CLEAR ERROR
1865	002042	050500	DH100	
1866	002044	052264	DT100	
1867	002046	052670	DF12	
1868				
1869			:ERROR 45	
1870	002050	047161	EM64	:NO ATTENTION IN ATTENTION SUMMARY REG
1871	002052	050500	DH100	
1872	002054	052264	DT100	
1873	002056	052670	DF12	
1874				
1875			:ERROR 46	
1876	002060	047224	EM65	:UNSOLICITED ATTENTION
1877	002062	050500	DH100	
1878	002064	052264	DT100	
1879	002066	052670	DF12	
1880				
1881			:ERROR 47	
1882	002070	047252	EM66	:UNEXPECTED DATA TYPE ERROR
1883	002072	050500	DH100	
1884	002074	052264	DT100	
1885	002076	052670	DF12	
1886				
1887			:ERROR 50	
1888	002100	047305	EM67	:ATTENTION DID NOT RESET WITH CLEAR
1889	002102	050500	DH100	
1890	002104	052264	DT100	
1891	002106	052670	DF12	
1892				
1893			:ERROR 51	
1894	002110	047344	EM70	:SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION
1895	002112	050500	DH100	
1896	002114	052264	DT100	
1897	002116	052670	DF12	
1898				
1899			:ERROR 52	
1900	002120	046674	EM52	:MULTIPLE DRIVE SELECT
1901	002122	050500	DH100	
1902	002124	052264	DT100	
1903	002126	052670	DF12	
1904				
1905			:ERROR 53	
1906	002130	046722	EM53	:ABREVIATED HCE ERROR
1907	002132	050500	DH100	
1908	002134	052264	DT100	
1909	002136	052720	DF13	
1910				
1911			:ERROR 54	
1912	002140	046222	EM30	:OPERATION INCOMPLETE ERROR
1913	002142	050500	DH100	
1914	002144	052264	DT100	
1915	002146	052750	DF14	
1916				

1917			:ERROR 55	
1918	002150	046247	EM31	:ABREVIATED HVRC ERROR
1919	002152	050500	DH100	
1920	002154	052264	DT100	
1921	002156	052720	DF13	
1922				
1923			:ERROR 56	
1924	002160	046747	EM56	:2 TIMEOUT ERROR
1925	002162	050500	DH100	
1926	002164	052264	DT100	
1927	002166	053000	DF15	
1928				
1929			:ERROR 57	:2ND LEVEL IN SUBSYSTEM TIMEOUT
1930	002170	046747	EM56	
1931	002172	050500	DH100	
1932	002174	052264	DT100	
1933	002176	053044	DF16	
1934				
1935			:ERROR 60	
1936	002200	046766	EM60	:ERROR IN RECAL FOR RECOVERY
1937	002202	000000	0	
1938	002204	000000	0	
1939	002206	000000	0	
1940				
1941			:ERROR 61	
1942	002210	047022	EM61	:ABORT MESSAGE
1943	002212	000000	0	
1944	002214	000000	0	
1945	002216	000000	0	
1946				
1947			:ERROR 62	
1948	002220	047070	EM62	:CYLINDER MISCOMPARE
1949	002222	050500	DH100	
1950	002224	052264	DT100	
1951	002226	053110	DF17	
1952				
1953			:ERROR 63	:DATA ERROR WORDS
1954	002230	000000	0	
1955	002232	000000	0	
1956	002234	052356	DT602	
1957	002236	053220	DF25	
1958				
1959			:ERROR 64	
1960	002240	047114	EM63	:CLEAR CONTROLLER DID NOT CLEAR ERROR
1961	002242	050500	DH100	
1962	002244	052264	DT100	
1963	002246	052424	DF02	
1964				
1965			:ERROR 65	
1966	002250	047161	EM64	:NO ATTENTION IN ATTENTION SUMMARY REG
1967	002252	050500	DH100	
1968	002254	052264	DT100	
1969	002256	052424	DF02	
1970				
1971			:ERROR 66	
1972	002260	047224	EM65	:UNSOLICITED ATTENTION

1973	002262	050500	DH100	
1974	002264	052264	DT100	
1975	002266	052424	DF02	
1976				
1977			:ERROR 67	
1978	002270	047252	EM66	:UNEXPECTED DATA TYPE ERROR
1979	002272	050500	DH100	
1980	002274	052264	DT100	
1981	002276	052424	DF02	
1982				
1983			:ERROR 70	
1984	002300	047305	EM67	:ATTENTION DID NOT RESET WITH CLEAR
1985	002302	050500	DH100	
1986	002304	052264	DT100	
1987	002306	052424	DF02	
1988				
1989			:ERROR 71	
1990	002310	047344	EM70	:SUBSYSTEM CLEAR DID NOT CLEAR ATT
1991	002312	050500	DH100	
1992	002314	052264	DT100	
1993	002316	052424	DF02	
1994				
1995			:ERROR 72	
1996	002320	047413	EM71	:DATA LATE WHEN UNLOADING HEADER
1997	002322	050500	DH100	
1998	002324	052264	DT100	
1999	002326	052424	DF02	
2000				
2001			:ERROR 73	
2002	002330	047453	EM72	:CONTROLLER ERROR DURING DRIVER SERVICE
2003	002332	050500	DH100	
2004	002334	052264	DT100	
2005	002336	052424	DF02	
2006				
2007			:ERROR 74	
2008	002340	047522	EM73	:DRIVE DETECTED PARITY ERROR
2009	002342	050500	DH100	
2010	002344	052264	DT100	
2011	002346	052424	DF02	
2012				
2013			:ERROR 75	
2014	002350	047556	EM74	:UNDEFINED ERROR
2015	002352	050500	DH100	
2016	002354	052264	DT100	
2017	002356	052424	DF02	
2018				
2019			:ERROR 76	
2020	002360	047576	EM75	:MARKING SECTOR BAD MESSAGE
2021	002362	000000	0	
2022	002364	000000	0	
2023	002366	000000	0	
2024				
2025			:ERROR 77	
2026	002370	047626	EM76	:BAD DATA VERIFICATION WITH READ
2027	002372	051465	DH605	
2028	002374	052350	DT601	

2029	002376	053144	DF21	
2030				
2031			:ERROR 100	
2032	002400	047710	EM77	;RETRY SUCCESSFUL MESSAGE
2033	002402	000000	0	
2034	002404	052350	DT601	
2035	002406	053204	DF23	
2036				
2037			:ERROR 101	
2038	002410	047710	EM77	;ANOTHER RETRY SUCCESSFUL MESSAGE
2039	002412	000000	0	
2040	002414	052350	DT601	
2041	002416	053204	DF23	
2042				
2043			:ERROR 102	
2044	002420	047731	EM100	;RETRY UNSUCCESSFUL MESSAGE
2045	002422	000000	0	
2046	002424	052350	DT601	
2047	002426	053204	DF23	
2048				
2049			:ERROR 103	
2050	002430	047754	EM101	;NO VALID HEADERS IN TRACK JUST READ
2051	002432	051447	DH6042	
2052	002434	052350	DT601	
2053	002436	053214	DF24	
2054				
2055			:ERROR 104	
2056	002440	050032	EM102	;BSE ERROR ON SECTOR NOT LISTED AS BAD
2057	002442	050500	DH100	
2058	002444	052264	DT100	
2059	002446	052424	DF02	
2060				
2061			:ERROR 105	
2062	002450	050104	EM103	;TIMED-OUT ON READ HEADER
2063	002452	050500	DH100	
2064	002454	052264	DT100	
2065	002456	052460	DF03	
2066				
2067			:ERROR 106	
2068	002460	050132	EM104	;TIMED-OUT ON SEEK
2069	002462	050500	DH100	
2070	002464	052264	DT100	
2071	002466	052460	DF03	
2072				
2073			:ERROR 107	
2074	002470	050154	EM105	;DRIVE SIEZED BY OTHER PORT
2075	002472	050500	DH100	
2076	002474	052264	DT100	
2077	002476	052424	DF02	
2078				
2079			:ERROR 110	
2080	002500	050207	EM106	; 'DATA MISCMPR WHILE BAI SET'
2081	002502	050500	DH100	
2082	002504	052264	DT100	
2083	002506	053234	DF27	
2084				

2085			:ERROR 111	
2086	002510	050242	EM107	;'NO NEM WHEN EXPECTED''
2087	002512	000000	0	
2088	002514	000000	0	
2089	002516	000000	0	
2090				
2091			:ERROR 112	
2092	002520	050307	EM110	;'INTRPT WHEN CNTRLR NOT READY''
2093	002522	050500	DH100	
2094	002524	052264	DT100	
2095	002526	052374	DF01	
2096				
2097			:ERROR 113	
2098	002530	050342	EM111	;'NO ATT'N ON SEEK''
2099	002532	050500	DH100	
2100	002534	052264	DT100	
2101	002536	052424	DF02	
2102				
2103			:ERROR 114	
2104	002540	050363	EM112	;'DRIVE'S CYLINDER INCORRECT
2105	002542	051674	DH702	
2106	002544	052324	DT202	
2107	002546	053224	DF26	
2108				
2109			:ERROR 115	
2110	002550	000000	0	;'TYPE ADRS OF DATA MISCOMPARE(S)
2111	002552	000000	0	
2112	002554	052350	DT601	
2113	002556	052650	DF11	
2114				
2115			:ERROR 116	
2116	002560	046333	EM34	;'DATA MISCOMPARE (11/70)
2117	002562	051011	DH103	
2118	002564	000000	0	
2119	002566	053134	DF20	
2120				
2121			:ERROR 117	
2122	002570	000000	0	;'PART OF DATA MISCOMPARE
2123	002572	000000	0	
2124	002574	052264	DT100	
2125	002576	053154	DF22	
2126				
2127			:ERROR 120	
2128	002600	050416	EM113	;'ABORT- CAN'T READ BSF
2129	002602	050500	DH100	
2130	002604	052264	DT100	
2131	002606	052374	DF01	
2132				
2133			:ERROR 121	
2134	002610	050444	EM114	;'KT11 FAILURE
2135	002612	050500	DH100	
2136	002614	052264	DT100	
2137	002616	053260	DF30	
2138				
2139			:ERROR 122	
2140	002620	050461	EM115	;'MEM PARITY ERROR

2141	002622	050500	DH100	
2142	002624	052264	DT100	
2143	002626	053304	DF31	
2144				
2145			:ERROR 123	
2146	002630	045455	EM3	:NED ON SIZING UNDER APT
2147	002632	000000	0	
2148	002634	000000	0	
2149	002636	000000	0	
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
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

.SBTTL BIT ASSIGNMENTS IN THE RK611 REGISTERS

: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
: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
:RKDC (DESIRED CYLINDER)							
15	14	13	12	11	10	9	8
NU	NU	NU	NU	NU	NU	DC9	DC8
READ ONLY							
:RKDA (DESIRED TRACK AND SECTOR)							
15	14	13	12	11	10	9	8
UN	UN	UN	UN	UN	TA2	TA1	TA0
READ ONLY							
:RKDC (DESIRED CYLINDER)							
15	14	13	12	11	10	9	8
UN	UN	UN	SA4	SA3	SA2	SA1	SA0
READ ONLY							

2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
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

: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!

2275
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

: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	!	FMTE	!	DRPAR	!	ILF	!	SKI	!	ILC	!
READ ONLY															
:RKDS (DRIVE STATUS)															
15	14	13	12	11	10	9	8								
SVAL	!	DSC	!	PIF	!	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	!	MEWD	!	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															

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

```

: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 ! EPO12! EPO11! EPO10! EPO9 ! EPO8 !
:                               READ ONLY
:-----

: 7 6 5 4 3 2 1 0
-----
EPO7 ! EPO6 ! EPO5 ! EPO4 ! EPO3 ! EPO2 ! EPO1 ! EPO0 !
:                               READ ONLY
:-----

:DRIVE STATUS INFORMATION

:LINE A MESSAGE 00
: 15 14 13 12 11 10 9 8
-----
: PAR ! DSC ! PIP ! SPON ! WRLK ! OFFON! FMT ! DRTP !
:-----

: 7 6 5 4 3 2 1 0
-----
:DRDY ! VV ! DRA ! NU ! NU ! DRIVE SELECT CODE !
:-----
    
```

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

:LINE B MESSAGE 00							
15	14	13	12	11	10	9	8
PAR	UN	DROT	DCLO	WLE	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					
:-----							

```

2390 ;LINE A MESSAGE 10
2391 ; 15 14 13 12 11 10 9 8
2392 -----
2393 ; PAR ! NU ! NU ! CYLINDER DIFF/OFFSET VALUE !
2394 -----
2395 ; 7 6 5 4 3 2 1 0
2396 -----
2397 ; CYLINDER DIFF/OFFSET VALUE! NU ! DRIVE SELECT CODE !
2398 -----
2399 ;
2400 ;LINE B MESSAGE 10
2401 ; 15 14 13 12 11 10 9 8
2402 -----
2403 ; PAR ! ALIGN! NU ! CYLINDER ADDRESS
2404 ; ! SIGN !
2405 -----
2406 ; 7 6 5 4 3 2 1 0
2407 -----
2408 ; CYLINDER ADDRESS ! UN ! UN ! 1 ! 0 !
2409 -----
2410 ;
2411 ;LINE A MESSAGE 11
2412 ; 15 14 13 12 11 10 9 8
2413 -----
2414 ; PAR ! DRIVE SERIAL NUMBER
2415 -----
2416 ; 7 6 5 4 3 2 1 0
2417 -----
2418 ; DRIVE SERIAL NUMBER ! DRIVE SELECT CODE !
2419 -----
2420 ;
2421 ;LINE B MESSAGE 11
2422 ; 15 14 13 12 11 10 9 8
2423 -----
2424 ; PAR ! NU ! NU ! DECODED HEAD ! SECTOR!
2425 ; ! ! ! ADDRESS ! COUNT !
2426 -----
2427 ; 7 6 5 4 3 2 1 0
2428 -----
2429 ; SECTOR COUNT ! UN ! UN ! 1 ! 1 !
2430 -----
2431 ;
2432 ;
2433 ;
2434 ;
2435 ;
2436 ;
2437 ;
2438 ;
2439 ;
2440 ;

```

.SBTTL RK06 CONTROLLER REGISTER DEFINITION

```

2441 000000 RKCS1= 0 ;CONTROL AND STATUS REGISTER 1
2442 000002 RKWC= 2 ;WORD COUNT REGISTER
2443 000004 RKBA= 4 ;BUS ADDRESS REGISTER
2444 000006 RKDA= 6 ;DESIRED TRACK SECTOR REGISTER
2445 000010 RKCS2= 10 ;CONTROL AND STATUS REGISTER 2

```

2446	000012	RKDS= 12	:DRIVE STATUS REGISTER
2447	000014	RKER= 14	:ERROR REGISTER
2448	000016	RKASOF= 16	:ATTENTION SUMMARY AND OFFSET REGISTER
2449	000020	RKDC= 20	:DESIRED CYLINDER REGISTER
2450	000020	RKDCYL= 20	:DESIRED CYLINDER REGISTER
2451	000024	RKDB= 24	:DATA BUFFER
2452	000026	RKMR1= 26	:MAINTENANCE REGISTER 1
2453	000034	RKMR2= 34	:MAINTENANCE REGISTER 2
2454	000036	RKMR3= 36	:MAINTENANCE REGISTER 3
2455	000030	RKPOS= 30	:ECC POSITION INFORMATION
2456	000030	RKECPS= 30	:ECC POSITION INFORMATION
2457	000032	RKPAT= 32	:ECC PATTERN INFORMATION
2458	000032	RKECPT= 32	:ECC PATTERN INFORMATION
2459			
2460		.SBTTL DRIVE COMMANDS	
2461			
2462	000101	SELDRV= 101	:SELECT DRIVE
2463	000103	PACK= 103	:PACK ACKNOWLEDGE
2464	000105	CLEAR= 105	:DRIVE CLEAR
2465	000107	UNLOAD= 107	:UNLOAD
2466	000111	SRTSPL= 111	:START SPINDLE
2467	000113	RECAL= 113	:RECALIBRATE
2468	000115	OFFSET= 115	:OFFSET
2469	000117	SEEK= 117	:SEEK
2470	000121	RDDATA= 121	:READ DATA
2471	000123	WRDATA= 123	:WRITE DATA
2472	000125	RDHEAD= 125	:READ HEADER
2473	000127	WRHEAD= 127	:WRITE HEADER AND DATA
2474	000131	WRTCHK= 131	:WRITE CHECK
2475			
2476		:	THE FOLLOWING ARE NOT DRIVE COMMANDS BUT ARE USED BY THE DRIVER
2477		:	TO SIMULATE A SPECIFIC DESIRED OPERATION
2478			
2479	000140	RELEAS= 140	:RELEASE DRIVE
2480	000141	RDSTAT= 141	:GET ALL STATUS FROM DRIVE
2481	000164	RDALHD= 164	:READ ALL HEADERS
2482	000176	CONCLR= 176	:CONTROLLER CLEAR (BIT 15 OF CS1)
2483	000177	SUBCLR= 177	:SUBSYSTEM CLEAR (BIT 5 OF CS2)
2484	000300	INTR= 300	:GENERATE INTERRUPT TO CPU
2485			
2486		:	DRIVER ISSUED SERVICE COMMANDS
2487			
2488	000001	DR.SEL= 001	:DRIVE SELECT
2489	000005	DR.CLR= 005	:DRIVE CLEAR
2490			
2491		.SBTTL CONTROL AND STATUS REGISTER 1 BITS	
2492			
2493	000001	GO= BIT0	:GO BIT
2494	000100	IE= BIT6	:INTERRUPT ENABLE
2495	000200	RDY= BIT7	:CONTROLLER READY
2496	000400	BA16= BIT8	:BUS ADDRESS BIT 16
2497	001000	BA17= BIT9	:BUS ADDRESS BIT 17
2498	002000	CDT= BIT10	:CONTROLLER DRIVE TYPE (0=RK06,1=RK07)
2499	004000	CTO= BIT11	:CONTROLLER TIMED OUT WAITING FOR
2500			: DRIVE RESPONSE
2501	010000	CFMT= BIT12	:CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)

2502	020000	SPAR= BIT13	:DRIVE BUS PARITY ERROR DETECTED BY CONTROLLER
2503	040000	DI= BIT14	:DRIVE INTERRUPT
2504	100000	CERR= BIT15	:CONTROLLER ERROR
2505	100000	CCLR= BIT15	:CONTROLLER CLEAR
2506			
2507		:	THESE BIT DEFINITIONS ARE USED FOR ADDRESS
2508		:	THE HIGH BYTE OF RKCS1
2509			
2510	000001	B.BA16= BIT0	:BUS ADDRESS BIT 16
2511	000002	B.BA17= BIT1	:BUS ADDRESS BIT 17
2512	000004	B.CDT= BIT2	:CONTROLLER DRIVE TYPE (0=RK06,1=RK07)
2513	000020	B.CFMT= BIT4	:CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)
2514			

.SBTTL CONTROL AND STATUS REGISTER 2 BITS

2517	000007	DRVMSK= 7	:MASK FOR DRIVE SELECTION CODE
2518	000010	DESL= BIT3	:DESELECT OR RELEASE DRIVE IN BITS 0-2
2519	000010	RLS= BIT3	:DESELECT OR RELEASE DRIVE IN BITS 0-2
2520	000020	BAI= BIT4	:BUS ADDRESS INCREMENT INHIBIT
2521	000040	CLR= BIT5	:CLEAR CONTROLLER AND ALL DRIVES
2522	000040	SCLR= BIT5	:CLEAR CONTROLLER AND ALL DRIVES
2523	000100	IR= BIT6	:INPUT READY
2524	000200	OR= BIT7	:OUTPUT READY
2525	000400	UFE= BIT8	:UNIT FIELD ERROR
2526	001000	MDS= BIT9	:MULTIPLE DRIVE SELECT
2527	002000	PGE= BIT10	:PROGRAMMING ERROR
2528	004000	NEM= BIT11	:NON-EXISTENT MEMORY
2529	010000	NED= BIT12	:NON-EXISTENT DRIVE
2530	020000	UPE= BIT13	:UNIBUS PARITY ERROR
2531	040000	WCE= BIT14	:WRITE CHECK ERROR
2532	100000	DLT= BIT15	:DATA LATE ERROR
2533			

.SBTTL ERROR REGISTER BIT DEFINITION

2536	000001	ILC= BIT0	:ILLEGAL FUNCTION CODE
2537		*ILF= BIT0	:ILLEGAL FUNCTION CODE
2538	000002	SKI= BIT1	:SEEK INCOMPLETE
2539	000004	ILF= BIT2	:ILLEGAL DRIVE FUNCTION
2540	000004	NXF= BIT2	:ILLEGAL DRIVE FUNCTION
2541	000010	DRPAR= BIT3	:DRIVE DETECTED DRIVE BUS PARITY ERROR
2542	000020	FMTE= BIT4	:FORMAT ERROR
2543	000040	DTYE= BIT5	:DRIVE TYPE ERROR
2544	000100	ECH= BIT6	:ECC HARD
2545	000200	BSE= BIT7	:BAD SECTOR ERROR
2546	000400	HCRC= BIT8	:HEADER CRC ERROR
2547	000400	HVRC= BIT8	:HEADER VRC ERROR
2548	001000	COE= BIT9	:CYLINDER ADDRESS OVERFLOW ERROR
2549	002000	IDAE= BIT10	:INVALID DISK ADDRESS ERROR
2550	004000	WLE= BIT11	:WRITE LOCK ERROR
2551	010000	DTE= BIT12	:DRIVE TIMING ERROR
2552	020000	OPI= BIT13	:OPERATION (SEARCH) INCOMPLETE
2553	040000	UNS= BIT14	:DRIVE UNSAFE
2554	100000	DCK= BIT15	:DATA CHECK
2555			

.SBTTL STATUS REGISTER BIT DEFINITION

2556
2557

```

2558      000001      DRA=   BIT0      ;DRIVE AVAILABLE (CONTROLLER IS SET IF
2559                                     ; THIS BIT IS RESET)
2560      000004      OFST=  BIT2      ;DRIVE OFFSET
2561      000010      ACLO=  BIT3      ;AC LOW
2562      000020      SPDLSS= BIT4      ;SPEED LOSS
2563      000020      DCLO=  BIT4      ;DC LOW
2564      000040      DROT=  BIT5      ;DRIVE OFF TRACK
2565      000100      VV=    BIT6      ;VOLUME VALID
2566      000200      DRY=   BIT7      ;DRIVE READY
2567      000200      DRDY=  BIT7      ;DRIVE READY
2568      000400      DDT=   BIT8      ;DRIVE TYPE (0=RK06,1=RK07)
2569      004000      WRL=   BIT11     ;WRITE LOCK
2570      020000      PIP=   BIT13     ;POSITIONING IN PROGRESS
2571      040000      DSC=   BIT14     ;DRIVE STATUS CHANGE
2572      100000      SVAL=  BIT15     ;STATUS VALID
2573
2574      .SBTTL MAINTENANCE REGISTER 1 BIT DEFINITION
2575
2576      000017      MESMSK= 17      ;MESSAGE MASK
2577
2578      000020      PAT=   BIT4      ;FORCE EVEN PARITY ON DRIVE BUS MESSAGE LINES
2579      000040      DMD=   BIT5      ;DIAGNOSTIC MODE
2580      000100      MSP=   BIT6      ;MAINTENANCE SECTOR PULSE
2581      000200      MIND=  BIT7      ;MAINTENANCE INDEX
2582      000400      MCLK=  BIT8      ;MAINTENANCE CLOCK
2583      001000      MERD=  BIT9      ;MAINTENANCE ENCODED READ DATA
2584      002000      MEWD=  BIT10     ;MAINTENANCE ENCODED WRITE DATA
2585      004000      PCA=   BIT11     ;PRECOMPENSATION ADVANCE
2586      010000      PCD=   BIT12     ;PRECOMPENSATION DELAY
2587      020000      ECCW=  BIT13     ;ECC WORD IS BEING READ OR WRITTEN
2588      040000      WRTGAT= BIT14     ;WRITE GATE
2589      100000      RDGATE= BIT15     ;READ GATE
2590
2591      .SBTTL DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE A
2592
2593      000040      S.DRA=  BIT5      ;DRIVE AVAILIABLE
2594      000100      S.VV=   BIT6      ;VOLUME VALID
2595      000200      S.DRY=  BIT7      ;DRIVE READY
2596      000400      S.TYPE= BIT8      ;DRIVE TYPE
2597      001000      S.FORM= BIT9      ;DRIVE FORMAT
2598      002000      S.OFF=  BIT10     ;OFFSET
2599      004000      S.WRL=  BIT11     ;WRITE LOCK
2600      010000      S.SPIN= BIT12     ;SPINDLE ON
2601      020000      S.PIP=  BIT13     ;POSITIONING IN PROGRESS
2602      040000      S.DSC=  BIT14     ;DRIVE STATUS CHANGE
2603
2604      .SBTTL DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE B
2605
2606      000040      S.ICYL= BIT5      ;ILLEGAL CYLINDER ADDRESS
2607      000100      S.ACLO= BIT6      ;AC LOW
2608      000200      S.FLT=  BIT7      ;DRIVE FAULT
2609      000400      S.ILF=  BIT8      ;ILLEGAL FUNCTION
2610      001000      S.PAR=  BIT9      ;DRIVE DETECTED DRIVE BUS PARITY ERROR
2611      002000      S.SKI=  BIT10     ;SEEK INCOMPLETE
2612      004000      S.WLE=  BIT11     ;WRITE LOCK ERROR
2613      010000      S.SPLS= BIT12     ;SPEED LOSS
  
```

2614	010000	S.DCLO= BIT12	:DC LOW
2615	020000	S.DROT= BIT13	:DRIVE OFF TRACK
2616	040000	S.UNS= BIT14	:DRIVE UNSAFE
2617			
2618		.SBTTL DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE A	
2619			
2620	000020	S.XDOK= BIT4	:TRANSDUCER OK
2621	000040	S.HDHM= BIT5	:HEADS HOME
2622	000100	S.BRHM= BIT6	:BRUSHES HOME
2623	000200	S.DOOR= BIT7	:DOOR INTERLOCKED
2624	000400	S.CART= BIT8	:CARTRAGE INTERLOCK
2625	001000	S.SPOK= BIT9	:SPEED OK
2626	002000	S.FWD= BIT10	:FORWARD
2627	004000	S.REV= BIT11	:REVERSE
2628	010000	S.LOAD= BIT12	:HEADS LOADING
2629	020000	S.RTZ= BIT13	:RETURN TO ZERO
2630	040000	S.UNLD= BIT14	:HEADS UNLOADING
2631			
2632		.SBTTL DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE B	
2633			
2634	000020	S.SECT= BIT4	:SECTOR ERROR
2635	000040	S.WCLK= BIT5	:WRITE CLOCK AND NO WRITE GATE
2636	000100	S.WGAT= BIT6	:WRITE GATE AND NO TRANSISTIONS
2637	000200	S.HDFL= BIT7	:HEAD FAULT
2638	000400	S.MHD= BIT8	:MULTIPLE HEAD SELECT
2639	001000	S.XERR= BIT9	:INDEX ERROR
2640	002000	S.DIB= BIT10	:DIBIT ERROR
2641	004000	S.PLO= BIT11	:PLO ERROR
2642	010000	S.NMOV= BIT12	:SEEK AND NO MOTION
2643	020000	S.LIMD= BIT13	:LIMIT DETECT ON SEEK
2644	040000	S.BRKE= BIT14	:SERVO-BRAKE
2645			
2646		.SBTTL COMMON MASKS	
2647			
2648	000007	M.DRV= 7	:DRIVE CODE
2649	100000	M.PAR= BIT15	:PARITY
2650	000003	M.ID= 3	:BYTE ID
2651	017760	M.CDIF= 17760	:CYLINDER DIFFERENCE/OFFSET
2652	017760	M.CADD= 17760	:CYLINDER ADDRESS
2653	077770	M.SER= 77770	:DRIVE SERIAL NUMBER
2654	000760	M.SECT= 760	:SECTOR COUNT
2655	007000	M.HEAD= 7000	:HEAD DECODE

2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711

.SBTTL PARAMETER BLOCK ALLOCATION

```

*****
* 1  : COMMAND          ! DRIVE NO.          : 0
* 3  : CYLINDER ADDRESS :                : 2
* 5  : TRACK            ! SECTOR            : 4
* 7  : BA16-17,FORMAT,DRV TYPE! OFFSET      : 6
* 11 : BUS ADDRESS (LOW 16 BITS) :          : 10
* 13 : WORD COUNT (2'S COMPLEMENT) :        : 12
* 15 : PROGRAM DRIVE STATUS INFORMATION :      : 14
* 17 : COMMAND AND STATUS REGISTER 1 :      : 16
* 21 : COMMAND AND STATUS REGISTER 2 :      : 20
* 23 : WORD COUNT REGISTER :          : 22
* 25 : BUS ADDRESS REGISTER :          : 24
* 27 : DESIRED TRACK AND SECTOR :        : 26
* 31 : DESIRED CYLINDER :                : 30
* 33 : ATTENTION SUMMARY AND DRIVE OFFSET :      : 32
* 35 : ERROR REGISTER :                : 34
* 37 : STATUS REGISTER :                : 36
* 41 : MESSAGE LINE A STATUS BYTE 00 :      : 40
* 43 : MESSAGE LINE B STATUS BYTE 00 :      : 42
* 45 : MESSAGE LINE A STATUS BYTE 01 :      : 44
* 47 : MESSAGE LINE B STATUS BYTE 01 :      : 46
* 51 : MESSAGE LINE A STATUS BYTE 10 :      : 50
* 53 : MESSAGE LINE B STATUS BYTE 10 :      : 52
* 55 : MESSAGE LINE A STATUS BYTE 11 :      : 54
* 57 : MESSAGE LINE B STATUS BYTE 11 :      : 56
* 61 : ECC POSITION INFORMATION :          : 60
* 63 : ECC PATTERN INFORMATION :          : 62
*****
  
```

.SBTTL PARAMETERS PASSED TO THE DRIVER

: THE FOLLOWING DEFINITIONS ARE USED TO PASS PARAMETERS
 : TO THE RK06/RK07 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.CS1H= 7 ;RKCS1 BITS 8-15
000007 P.BAHI= 7 ;BUS ADDRESS (BITS 16 AND 17)
000010 P.BALO= 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 DRVPDT= BIT2 ;DRIVE POSITIONED FOR DATA TRANSFER
000010 UEXATT= BIT3 ;UNEXPECTED ATTENTION
000020 DRVHRD= BIT4 ;DRIVE HAS HARD ERROR
000040 DRVDSC= BIT5 ;DRIVE STATUS CHANGE DID NOT CLEAR
  
```

2712	000100	CMDTO= BIT6	:NO TERMINATION TO COMMAND FOR AT
2713			: LEAST 1 SECOND
2714	000200	W.WCK= BIT7	:WRITE FOR WRITE WRITE CHECK
2715	000400	NOCHK= BIT8	:NO CHECK, DO NOT SET INTERRUPT ENABLE
2716	001000	PBSVAL= BIT9	:PARAMETER STATUS WORDS VALID
2717			: (SET WHEN ERROR TERMINATION OR
2718			: READ STATUS COMMAND)
2719	002000	DRPDRV= BIT10	:DROP DRIVE FROM TEST SEQUENCE
2720	004000	NODSC= BIT11	:ATTENTION SET BUT DCS AND FAULT RESET
2721	010000	DRVSZD= BIT12	:DRIVE SEIZED BY OTHER PORT
2722	020000	E.UNLD= BIT13	:DRIVE UNLOADED DUE TO ERROR
2723	040000	Q.INIT= BIT14	:PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
2724	100000	DTBAII= BIT15	:INHIBIT BUS ADDRESS INCREMENT

.SBTTL PARAMETERS PASSED FROM DRIVER TO PROGRAM

: THE FOLLOWING DEFINITIONS ARE USED FOR REGISTER RETURNS
 : FROM THE DRIVER TO THE CALLING PROGRAM

2730			
2731	000016	P.CS1= 16	:COMMAND AND STATUS REGISTER 1
2732	000020	P.CS2= 20	:COMMAND AND STATUS REGISTER 2
2733	000022	P.WCR= 22	:WORD COUNT REGISTER
2734	000024	P.BAR= 24	:BUS ADDRESS REGISTER
2735	000026	P.DTS= 26	:DESIRED TRACK SECTOR REGISTER
2736	000030	P.DCYL= 30	:DESIRED CYLINDER REGISTER
2737	000032	P.ASOF= 32	:ATTENTION SUMMARY/OFFSET REGISTER
2738	000034	P.ER= 34	:ERROR REGISTER
2739	000036	P.DS= 36	:STATUS REGISTER
2740	000040	P.A00= 40	:MESSAGE A STATUS BYTE 00
2741	000042	P.B00= 42	:MESSAGE B STATUS BYTE 00
2742	000044	P.A01= 44	:MESSAGE A STATUS BYTE 01
2743	000046	P.B01= 46	:MESSAGE B STATUS BYTE 01
2744	000050	P.A10= 50	:MESSGGE A STATUS BYTE 10
2745	000052	P.B10= 52	:MESSAGE B STATUS BYTE 10
2746	000054	P.A11= 54	:MESSAGE A STATUS BYTE 11
2747	000056	P.B11= 56	:MESSAGE B STATUS BYTE 11
2748	000060	P.EPOS= 60	:ECC POSITION INFORMATION
2749	000062	P.EPAT= 62	:ECC PATTERN INFORMATION

.SBTTL PARAMETER BLOCK 0 FOR DRIVE

2750			
2751			
2752			
2753	002640	000	PARMO: .BYTE 0 :DRIVE NUMBER
2754	002641	000	.BYTE 0 :COMMAND
2755	002642	000000	.WORD 0 :CYLINDER ADDRESS
2756	002644	000	.BYTE 0 :SECTOR ADDRESS
2757	002645	000	.BYTE 0 :TRACK ADDRESS
2758	002646	000	.BYTE 0 :OFFSET VALUE
2759	002647	000	.BYTE 0 :BUS ADDRESS (BITS 16 AND 17)
2760	002650	000000	.WORD 0 :BUS ADDRESS (BITS 0 - 15)
2761	002652	000000	.WORD 0 :WORD COUNT (2'S COMPLEMENT)
2762	002654	000000	.WORD 0 :PROGRAM DRIVE STATUS INFORMATION
2763	002656	000000	.WORD 0 :COMMAND AND STATUS REGISTER 1
2764	002660	000000	.WORD 0 :COMMAND AND STATUS REGISTER 2
2765	002662	000000	.WORD 0 :WORD COUNT REGISTER
2766	002664	000000	.WORD 0 :BUS ADDRESS REGISTER
2767	002666	000000	.WORD 0 :DESIRED TRACK AND SECTOR REGISTER

2768	002670	000000	.WORD	0	:DESIRED CYLINDER REGISTER
2769	002672	000000	.WORD	0	:ATTENTION SUMMARY/OFFSET REGISTER
2770	002674	000000	.WORD	0	:ERROR REGISTER
2771	002676	000000	.WORD	0	:STATUS REGISTER
2772	002700	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 00
2773	002702	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 00
2774	002704	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 01
2775	002706	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 01
2776	002710	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 10
2777	002712	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 10
2778	002714	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 11
2779	002716	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 11
2780	002720	000000	.WORD	0	:ECC POSITION INFORMATION
2781	002722	000000	.WORD	0	:ECC PATTERN INFORMATION

.SBTTL PARAMETER BLOCK 1 FOR DRIVE

2785	002724	000	PARM1: .BYTE	0	:DRIVE NUMBER
2786	002725	000	.BYTE	0	:COMMAND
2787	002726	000000	.WORD	0	:CYLINDER ADDRESS
2788	002730	000	.BYTE	0	:SECTOR ADDRESS
2789	002731	000	.BYTE	0	:TRACK ADDRESS
2790	002732	000	.BYTE	0	:OFFSET VALUE
2791	002733	000	.BYTE	0	:BUS ADDRESS (BITS 16 AND 17)
2792	002734	000000	.WORD	0	:BUS ADDRESS (BITS 0 - 15)
2793	002736	000000	.WORD	0	:WORD COUNT (2'S COMPLEMENT)
2794	002740	000000	.WORD	0	:PROGRAM DRIVE STATUS INFORMATION
2795	002742	000000	.WORD	0	:COMMAND AND STATUS REGISTER 1
2796	002744	000000	.WORD	0	:COMMAND AND STATUS REGISTER 2
2797	002746	000000	.WORD	0	:WORD COUNT REGISTER
2798	002750	000000	.WORD	0	:BUS ADDRESS REGISTER
2799	002752	000000	.WORD	0	:DESIRED TRACK AND SECTOR REGISTER
2800	002754	000000	.WORD	0	:DESIRED CYLINDER REGISTER
2801	002756	000000	.WORD	0	:ATTENTION SUMMARY/OFFSET REGISTER
2802	002760	000000	.WORD	0	:ERROR REGISTER
2803	002762	000000	.WORD	0	:STATUS REGISTER
2804	002764	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 00
2805	002766	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 00
2806	002770	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 01
2807	002772	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 01
2808	002774	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 10
2809	002776	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 10
2810	003000	000000	.WORD	0	:MESSAGE LINE A STATUS BYTE 11
2811	003002	000000	.WORD	0	:MESSAGE LINE B STATUS BYTE 11
2812	003004	000000	.WORD	0	:ECC POSITION INFORMATION
2813	003006	000000	.WORD	0	:ECC PATTERN INFORMATION

.SBTTL TEMPORARY CONTROLLER REGISTER STORAGE

2817	003010	000000	T.CS1: .WORD	0	:TEMPORARY STORAGE FOR COMMAND AND STATUS REGISTER 1
2819	003012	000000	T.CS2: .WORD	0	:TEMPORARY STORAGE FOR COMMAND AND STATUS REGISTER 2
2821	003014	000000	T.WCR: .WORD	0	:TEMPORARY STORAGE FOR WORD COUNT REGISTER
2822	003016	000000	T.BA: .WORD	0	:TEMPORARY STORAGE FOR BUS ADDRESS REGISTER
2823	003020	000000	T.DA: .WORD	0	:TEMPORARY STORAGE FOR DISK TRACK AND SECTOR

2824	003022	000000	T.DC: .WORD	0	: TEMPORARY STORAGE FOR DRIVE CYLINDER
2825	003024	000000	T.ASOF: .WORD	0	: TEMPORARY STORAGE FOR ATTENTION SUMMARY
2826					: AND OFFSET
2827	003026	000000	T.ER: .WORD	0	: TEMPORARY STORAGE FOR ERROR REGISTER
2828	003030	000000	T.DS: .WORD	0	: TEMPORARY STORAGE FOR DRIVE STATUS REGISTER
2829	003032	000000	T.MR1: .WORD	0	: TEMPORARY STORAGE FOR MAINTENANCE REGISTER 1
2830	003034	000000	T.MR2: .WORD	0	: TEMPORARY STORAGE FOR MAINTENANCE REGISTER 2
2831	003036	000000	T.MR3: .WORD	0	: TEMPORARY STORAGE FOR MAINTENANCE REGISTER 3
2832	003040	000000	T.POS: .WORD	0	: TEMPORARY STORAGE FOR ECC POSITION
2833	003042	000000	T.PAT: .WORD	0	: TEMPORARY STORAGE FOR ECC PATTERN
2834	003044	000000	T.DB: .WORD	0	: TEMPORARY STORAGE FOR DATA BUFFER REGISTER
2835					
2836			.SBTTL DRIVER PARAMETERS		
2837					
2838	003046	177440	RKBAS: .WORD	177440	: ADDRESS OF RK611 UNIBUS ADDRESS BLOCK
2839	003050	000210	RKVEC: .WORD	210	: ADDRESS OF R611 VECTOR
2840	003052	000240	RKPRI: .WORD	PR5	: RK611 INTERRUPT PRIORITY
2841	003054	030250	A.NORM: ERRFRE		: ADDRESS OF NORMAL RETURN FROM DRIVER
2842	003056	031472	A.ABNL: ERRHDL		: ADDRESS OF ABNORMAL RETURN FROM DRIVER
2843	003060	030756	A.CONT: CONERR		: ADDRESS OF CONTROLLER ERROR RETURN
2844	003062	000000	E.CONT: .WORD	0	: CONTROLLER ERROR STATUS
2845					: THIS LOCATION IS CLEARED WHEN EVERY COMMAND
2846					: IS INITIATED. IF A CONTROLLER ERROR
2847					: OCCURS THE FOLLOWING BIT ASSIGNMENT IS
2848					: USED:
2849					
2850		000001	E.CCLF= BIT0		: CLEAR CONTROLLER DID NOT CLEAR ERROR
2851		000002	E.NOAT= BIT1		: NO ATTENTION IN ATTENTION SUMMARY REG
2852		000004	E.UATT= BIT2		: UNSOLICITED ATTENTION (SEQUENTIAL ONLY)
2853		000010	E.UDAT= BIT3		: UNEXPECTED DATA TYPE ERROR
2854		000020	E.CLAT= BIT4		: ATTENTION DID NOT RESET WITH CLEAR
2855		000040	E.SCLR= BIT5		: SUBSYSTEM CLEAR DID NOT CLEAR DRIVE
2856					: ATTENTION
2857		000100	E.ILLD= BIT6		: ILLEGAL DRIVER COMMAND
2858		000400	E.DLT= BIT8		: DATA LATE WHEN UNLOADING HEADER
2859		001000	E.CERR= BIT9		: CONTROLLER ERROR DURING DRIVER SERVICING
2860		002000	E.DPAR= BIT10		: DRIVE DETECTED PARITY ERROR
2861		040000	E.CMTO= BIT14		: CONTROLLER COMMAND TIME OUT (QUEUED ONLY)
2862		100000	E.MDS= BIT15		: MULTIPLE DRIVE SELECT
2863					
2864	003064	000000	O.WAIT: .WORD	0	: PARAMETER BLOCK OF THE DRIVE
2865					: WAITING FOR COMMAND COMPLETION
2866	003066	000400	W.MTIM: .WORD	400	: LOOP COUNTER FOR MILLISECOND SCAN OF DRIVE
2867	003070	000400	W.MILI: .WORD	400	: 16 MILLISECOND TIME FOR PROGRAM
2868					
2869					
2870					
2871					
2872					
2873					
2874					
2875					
2876					
2877					
2878					
2879					

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

```

2880 003072 000300      W.SEC: .WORD 300      ;SECOND COUNT COUNT FOR ALL COMMANDS
2881                                ; EXCEPT START SPINDLE
2882 003074 003000      W.BSEC: .WORD 3000    ;8 SECOND FOR DRIVE CYCLE DOWN
2883 003076 030000      W.MIN: .WORD 30000   ;MINUTE TIME FOR START SPINDLE
2884 003100 000000      HDR.AD: .WORD 0      ;ADDRESS USED FOR READ ALL HEADERS
2885 003102 000000      HDR.CT: .WORD 0      ;NUMBER OF HEADERS LEFT TO READ FOR READ
2886                                ; ALL HEADERS
2887 003104 000          I.ISRL: .BYTE 0      ;INTERRUPT OR RELEASED COMMAND ISSUED
2888 003105 002 004 010 H.HEAD: .BYTE 2,4,10 ;HEAD DECODES
2889 003110 000          W.TIME: .BYTE 0      ;DRIVES BEING WATCH-DOG TIMED
2890
2891      .SBTTL INTERRUPT MASKS
2892
2893 003111 000          INTMSK: .BYTE 0      ;INTERRUPT MASKS FOR DRIVE IN PARAMETER BLOCK
2894
2895      ; INTERRUPT MASK TABLE
2896
2897 003112 001          I.DRV: .BYTE 1      ;INTERRUPT MASK FOR DRIVE 0
2898 003113 002          .BYTE 2      ;INTERRUPT MASK FOR DRIVE 1
2899 003114 004          .BYTE 4      ;INTERRUPT MASK FOR DRIVE 2
2900 003115 010          .BYTE 10     ;INTERRUPT MASK FOR DRIVE 3
2901 003116 020          .BYTE 20     ;INTERRUPT MASK FOR DRIVE 4
2902 003117 040          .BYTE 40     ;INTERRUPT MASK FOR DRIVE 5
2903 003120 100          .BYTE 100    ;INTERRUPT MASK FOR DRIVE 6
2904 003121 200          .BYTE 200    ;INTERRUPT MASK FOR DRIVE 7
2905
2906      .SBTTL PARAMETER BLOCK TABLE
2907
2908 003122 002640      PBLKT: PARM0      ;ADDRESS OF PARAMETER BLOCK GIVEN WITH
2909                                ;DRIVE CALL. MUST BE LOADED INTO PBLKT
2910
2911
2912      .SBTTL TIME FOR WATCH-DOG TIMER
2913
2914 003124 000000      W.DRV: .WORD 0      ;TIME FOR INSTRUCTION IN PARAMETER BLOCK
2915      .SBTTL PROGRAM SPECIFIC RESERVED LOCATIONS
2916 003126 000          MDFLAG: .BYTE 0      ;FLAG TO INDIC. DEFLT OR PARAM MODE
2917 003127 000          TSTING: .BYTE 0      ;CURRENTLY RUNNING TESTS IF = 1
2918 003130 000          DERCNT: .BYTE 0      ;DATA ERROR COUNT
2919 003131 000          OPCOMP: .BYTE 0      ;OPERATION COMPLETE FLAG
2920 003132 000          DONE: .BYTE 0      ;DONE SWITCH
2921 003133 000          TYPFMT: .BYTE 0      ;DRIVE TYPE & FORMAT CONTROL
2922 003134 000          FORMAT: .BYTE 0      ;DRIVE FORMAT IN BIT 4 OF BYTE
2923 003135 000          ERRCNT: .BYTE 0      ;ERROR COUNT
2924 003136 004          ERRLMT: .BYTE 4      ;ERROR LIMIT
2925 003137 000          DRVERS: .BYTE 0      ;ERROR COUNT FOR CURRENT DRIVE
2926 003140 000          OPCONT: .BYTE 0      ;OPERATION CONTROL SWITCHES
2927 003141 000          DRNAFG: .BYTE 0      ;=1 INDICATES DRIVE SIEZED BY OTHER PORT
2928 003142 000          REISSU: .BYTE 0      ;DUAL-ACC FLAG TO RE-ISSUE COMMAND
2929 003143 000          TSTTYP: .BYTE 0      ;OFFSET TEST IDENTIFIER
2930 003144 000          DCEFLG: .BYTE 0      ;DATA CHECK ERROR FLAG
2931 003145 000          WCEFLG: .BYTE 0      ;WRITE CHECK ERROR FLAG
2932 003146 000          DLTF LG: .BYTE 0      ;DATA LATE ERROR FLAG
2933 003147 000          DIF100: .BYTE 0      ;"CYL DIF = 100/200" FLAG
2934 003150 000          NORTRY: .BYTE 0      ;"NO-RETRY" FLAG
2935 003151 000          MEMABT: .BYTE 0      ;SET BYTE = 1 FOR NO ABORT
  
```


2992
 2993
 2994
 2995
 2996
 2997 005566
 2998 005566 000620 000000
 2999 005572 000620 000002
 3000 005576 000620 000004
 3001 005602 000620 000006
 3002 005606 000620 000010
 3003 005612 000620 000012
 3004 005616 000620 000014
 3005 005622 000620 000016
 3006 005626 000622 000000
 3007 005632 000622 000002
 3008 005636 000622 000004
 3009 005642 000622 000006
 3010 005646 000622 000010
 3011 005652 000622 000012
 3012 005656 000622 000014
 3013 005662 000622 000016
 3014
 3015
 3016

:TABLE A - BASIC READ/WRITE TEST SECTORS FOR RK06

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

3017 005666 001444 000000
 3018 005672 001444 000002
 3019 005676 001444 000004
 3020 005702 001444 000006
 3021 005706 001444 000010
 3022 005712 001444 000012
 3023 005716 001444 000014
 3024 005722 001444 000016
 3025 005726 001446 000000
 3026 005732 001446 000002
 3027 005736 001446 000004
 3028 005742 001446 000006
 3029 005746 001446 000010
 3030 005752 001446 000012
 3031 005756 001446 000014
 3032 005762 001446 000016
 3033
 3034
 3035

:TABLE A - CORRESPONDING CYL NUMBERS FOR THE RK07

TABL7A:
 .WORD 1444,0 :DRV 0
 .WORD 1444,2 : 1
 .WORD 1444,4 : 2
 .WORD 1444,6 : 3
 .WORD 1444,10 : 4
 .WORD 1444,12 : 5
 .WORD 1444,14 : 6
 .WORD 1444,16 : 7
 .WORD 1446,0 : 10
 .WORD 1446,2 : 11
 .WORD 1446,4 : 12
 .WORD 1446,6 : 13
 .WORD 1446,10 : 14
 .WORD 1446,12 : 15
 .WORD 1446,14 : 16
 .WORD 1446,16 : 17

3036 005766
 3037 005766 072307
 3038 005766 072307
 3039 005770 135143
 3040 005772 135143
 3041 005774 156461
 3042 005774 156461
 3043 005776 167230
 3044 005776 167230
 3045 005776 073514
 3046 006000 073514
 3047 006000 035646
 006002 016723
 006004 016723
 006006 107351
 006008 107351
 006010 143564
 006012 143564
 006014 061672
 006016 061672

:TABLE B - WORST CASE DATA PATTERN

TABLEB:
 072307
 135143
 156461
 167230
 073514
 035646
 016723
 107351
 143564
 061672

3048	006012	030735	030735
3049	006014	114356	114356
3050	006016	046167	046167
3051	006020	123073	123073
3052	006022	151453	151453
3053	006024	164616	164616

3054
3055
3056

:TABLE OF STARTING CYLINDERS FOR ALL OVWRT AND COMPAT CYL BLKS FOR RK06

3058	006026	000000	000060
3059	006032	000100	000160
3060	006036	000200	000260
3061	006042	000300	000360
3062	006046	000400	000460
3063	006052	000500	000560
3064	006056	000600	

TABLEC:	.WORD	0,60	:	FOR CURRENT ZONE	1
	.WORD	100,160	:		2
	.WORD	200,260	:		3
	.WORD	300,360	:		4
	.WORD	400,460	:		5
	.WORD	500,560	:		6
	.WORD	600	:		7

3065
3066
3067

:TABLE C CORRESPONDING CYL NUMBERS FOR RK07

3068	006060	000000	000160
3069	006064	000200	000360
3070	006070	000400	000560
3071	006074	000600	000760
3072	006100	001000	001160
3073	006104	001200	001360
3074	006110	001400	

TABL7C:	.WORD	0,160	:	CURRENT ZONE	1
	.WORD	200,360	:		2
	.WORD	400,560	:		3
	.WORD	600,760	:		4
	.WORD	1000,1160	:		5
	.WORD	1200,1360	:		6
	.WORD	1400	:		7

3075
3076
3077

:CYLINDER BLOCK LAYOUT TABLE

3078			
3079	006112	000020	
3080	006132	000020	
3081	006152	000020	
3082	006172	000020	
3083	006212	000020	
3084	006232	000020	
3085	006252	000020	
3086	006272	000020	
3087	006312	000020	
3088	006332	000020	
3089	006352	000020	
3090	006372	000020	
3091	006412	000020	
3092	006432	000020	
3093	006452	000020	
3094	006472	000020	

TABLED:	.BLKB	16.	:	CYLINDER	0	IN BLK
	.BLKB	16.	:		1	
	.BLKB	16.	:		2	
	.BLKB	16.	:		3	
	.BLKB	16.	:		4	
	.BLKB	16.	:		5	
	.BLKB	16.	:		6	
	.BLKB	16.	:		7	
	.BLKB	16.	:		10	
	.BLKB	16.	:		11	
	.BLKB	16.	:		12	
	.BLKB	16.	:		13	
	.BLKB	16.	:		14	
	.BLKB	16.	:		15	
	.BLKB	16.	:		16	
	.BLKB	16.	:		17	

3095
3096
3097

:TABLE G - PSEUDO-RANDOM DATA PATTERN

3098			
3099	006512		
3100	006512	040135	
3101	006514	177070	
3102	006516	070414	
3103	006520	064531	

TABLEG:		040135
		177070
		070414
		064531

3160
3161 ;TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 0 FOR NEG OFST
3162 007112 000020 NCSCRO: .BLKW 16.
3163
3164 ;TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 1 FOR NEG OFST
3165 007152 000020 NCSCR1: .BLKW 16.
3166
3167 ;TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 2 FOR NEG OFST
3168 007212 000020 NCSCR2: .BLKW 16.
3169
3170
3171
3172 ;TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 0 FOR POS OFST
3173 007252 000020 PCSCRO: .BLKW 16.
3174
3175 ;TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 1 FOR POS OFST
3176 007312 000020 PCSCR1: .BLKW 16.
3177
3178 ;TABLE OF DATA COMPATIBILITY TEST SCORES FOR TRACK 2 FOR POS OFST
3179 007352 000020 PCSCR2: .BLKW 16.
3180
3181
3182
3183
3184 ;SELF-TEST SCORE FOR TRACK 0 FOR NEG OFST
3185 007412 000000 NSSCRO: .WORD 0
3186
3187 ;SELF-TEST SCORE FOR TRACK 1 FOR NEG OFST
3188 007414 000000 NSSCR1: .WORD 0
3189
3190 ;SELF-TEST SCORE FOR TRACK 2 FOR NEG OFST
3191 007416 000000 NSSCR2: .WORD 0
3192
3193
3194
3195 ;SELF-TEST SCORE FOR TRACK 0 FOR POS OFST
3196 007420 000000 PSSCRO: .WORD 0
3197
3198 ;SELF-TEST SCORE FOR TRACK 1 FOR POS OFST
3199 007422 000000 PSSCR1: .WORD 0
3200
3201 ;SELF-TEST SCORE FOR TRACK 2 FOR POS OFST
3202 007424 000000 PSSCR2: .WORD 0
3203
3204
3205
3206 ;AVERAGE OF OVRWRT SCORES OVER ALL SURFACES
3207 007426 000000 OVR AVG: .WORD 0
3208
3209 ;AVERAGE OF READ SCORES OVER ALL SURFACES
3210 007430 000000 COMAVG: .WORD 0
3211
3212
3213
3214 ;SCRATCH TABLE OF SCORES FOR TRACK 0 FOR NEG OFST
3215 007432 000020 NSCORO: .BLKB 16.

```

3216
3217
3218 007452 000020 ;SCRATCH TABLE OF SCORES FOR TRACK 1 FOR NEG OFST
3219 NSCOR1: .BLKB 16.
3220
3221 007472 000020 ;SCRATCH TABLE OF SCORES FOR TRACK 2 FOR NEG OFST
3222 NSCOR2: .BLKB 16.
3223
3224
3225
3226 007512 000020 ;SCRATCH TABLE OF SCORES FOR TRACK 0 FOR POS OFST
3227 PSCOR0: .BLKB 16.
3228
3229 007532 000020 ;SCRATCH TABLE OF SCORES FOR TRACK 1 FOR POS OFST
3230 PSCOR1: .BLKB 16.
3231
3232 007552 000020 ;SCRATCH TABLE OF SCORES FOR TRACK 2 FOR POS OFST
3233 PSCOR2: .BLKB 16.
3234
3235
3236 000002 LSTTRK=2 ;LAST TRACK = 2
3237 000002 BSERR=BIT1 ;BSE ERROR
3238 000004 HVR CER=BIT2 ;HVRC ERROR
3239 000010 OPIERR=BIT3 ;OPI ERROR
3240 000020 DCKERR=BIT4 ;DATA CHECK ERROR
3241 000040 ECCNC=BIT5 ;ECC NON-CORRECTABLE
3242 000100 WCERR=BIT6 ;WRITE CHECK ERROR
3243 000200 ABORT=BIT7 ;ABORT
3244 000400 LEV2ER=BIT8 ;LEVEL TWO ERROR
3245 001000 BADSEC=BIT9 ;BAD SECTOR FLAG
3246 002000 TWOTOS=BIT10 ;TWO TIME OUTS
3247 004000 RCLREQ=BIT11 ;RECALIBRATE REQUIRED
3248 010000 DRNAVL=BIT12 ;DRIVE NOT RELSD BY OTHER PORT
3249 100000 ANYDER=BIT15 ;ANY ERROR DETECTED FLAG
3250
3251 007572 005015 041412 051132 DZR6Q: .ASCIZ <15><12><12>/CZR6QC0 RK6 DR CPT PROG/<15><12><12>
3252 007600 050466 030103 051040
3253 007606 033113 042040 020122
3254 007614 050103 020124 051120
3255 007622 043517 005015 000012
3256 007630 051012 030113 026466 RKBADR: .ASCIZ <12>/RK06-07 BUS ADR = /
3257 007636 033460 041040 051525
3258 007644 040440 051104 036440
3259 007652 000040
3260 007654 045522 033060 030055 RKVADR: .ASCIZ /RK06-07 VEC ADR = /
3261 007662 020067 042526 020103
3262 007670 042101 020122 020075
3263 007676 000
3264 007677 122 030113 026466 RKPRTY: .ASCIZ /RK06-07 PRIORITY =/
3265 007704 033460 050040 044522
3266 007712 051117 052111 020131
3267 007720 000075
3268 007722 052012 050131 020105 TYP SYS: .ASCIZ <12>/TYPE NAME OF NEXT SUBSYS TO TEST (A-P) : /
3269 007730 040516 042515 047440
3270 007736 020106 042516 052130
3271 007744 051440 041125 054523
    
```

3272	007752	020123	047524	052040	
3273	007760	051505	020124	040450	
3274	007766	050055	020051	020072	
3275	007774	000			
3276	007775	012	051104	053111	ASKTYP: .ASCIZ <12>/DRIVE TYPE 6<CR> FOR RK06, 7<CR> FOR RK07: /
3277	010002	020105	054524	042520	
3278	010010	020040	036066	051103	
3279	010016	020076	047506	020122	
3280	010024	045522	033060	020054	
3281	010032	036067	051103	020076	
3282	010040	047506	020122	045522	
3283	010046	033460	020072	000	
3284	010053	123	041125	054523	SYSDRV: .ASCIZ /SUBSYS DRIVE(S) = /
3285	010060	020123	020040	051104	
3286	010066	053111	024105	024523	
3287	010074	036440	000040		
3288	010100	044527	046114	052040	WILTST: .ASCIZ /WILL TEST DRIVE(S) /
3289	010106	051505	020124	051104	
3290	010114	053111	024105	024523	
3291	010122	000040			
3292	010124	047440	020116	052523	ONSUBS: .ASCIZ / ON SUBSYS /<15><12><12>
3293	010132	051502	051531	020040	
3294	010140	005015	000012		
3295	010144	051511	052040	042510	ANOTHR: .ASCIZ /IS THERE ANOTHER SUBSYS (Y OR N <CR>) ? /
3296	010152	042522	040440	047516	
3297	010160	044124	051105	051440	
3298	010166	041125	054523	020123	
3299	010174	054450	047440	020122	
3300	010202	020116	041474	037122	
3301	010210	020051	020077	000	
3302	010215	012	047515	042522	DROVFL: .ASCIZ <12>/MORE THAN 16 DRIVES ENTERED !/<15><12><12>
3303	010222	052040	040510	020116	
3304	010230	033061	042040	044522	
3305	010236	042526	020123	047105	
3306	010244	042524	042522	020104	
3307	010252	006441	005012	000	
3308	010257	012	025052	051440	SRPASS: .ASCIZ <12>/** STARTING PASS **/<15><12>
3309	010264	040524	052122	047111	
3310	010272	020107	040520	051523	
3311	010300	020040	025040	006452	
3312	010306	000012			
3313	010310	046412	052517	052116	MNTPAK: .ASCIZ <12>/MOUNT PACK ON DRIVE AND LOAD./<15><12>
3314	010316	050040	041501	020113	
3315	010324	047117	042040	044522	
3316	010332	042526	020040	020040	
3317	010340	047101	020104	047514	
3318	010346	042101	006456	000012	
3319	010354	052412	046116	040517	DISPAK: .ASCIZ <12>/UNLOAD DRIVE AND REMOVE PACK./<15><12>
3320	010362	020104	051104	053111	
3321	010370	020105	020040	040440	
3322	010376	042116	051040	046505	
3323	010404	053117	020105	040520	
3324	010412	045503	006456	000012	
3325	010420	051412	040524	052122	STR204: .ASCIZ <12>/START AT ADR 204 FOR PASS 1 ON SUBSYS ./<15><12><12>
3326	010426	040440	020124	042101	
3327	010434	020122	030062	020064	

3328	010442	047506	020122	040520	
3329	010450	051523	030440	047440	
3330	010456	020116	052523	051502	
3331	010464	051531	020040	006456	
3332	010472	005012	000		
3333	010475	012	052123	051101	STR220: .ASCIZ <12>/START AT ADR 220 FOR PASS 2 ON SUBSYS ./<15><12><12>
3334	010502	020124	052101	040440	
3335	010510	051104	031040	030062	
3336	010516	043040	051117	050040	
3337	010524	051501	020123	020062	
3338	010532	047117	051440	041125	
3339	010540	054523	020123	027040	
3340	010546	005015	000012		
3341	010552	054524	042520	051040	TYPRWN: .ASCIZ /TYPE R<CR> WHEN /
3342	010560	041474	037122	053440	
3343	010566	042510	020116	000	
3344	010573	104	044522	042526	DRIRDY: .ASCIZ /DRIVE READY : /
3345	010600	051040	040505	054504	
3346	010606	035040	000040		
3347	010612	047504	042516	035040	DRIDON: .ASCIZ /DONE : /
3348	010620	000040			
3349	010622	047516	042040	044522	NODRLS: .ASCIZ /NO DRIVES LISTED FOR THIS SUBSYS !/<15><12>
3350	010630	042526	020123	044514	
3351	010636	052123	042105	043040	
3352	010644	051117	052040	044510	
3353	010652	020123	052523	051502	
3354	010660	051531	020440	005015	
3355	010666	000			
3356	010667	015	051412	047503	RESLTS: .ASCII <15><12>/SCORES FOR DRIVE (0-12 DEC.) :/<15><12><12>
3357	010674	042522	020123	047506	
3358	010702	020122	051104	053111	
3359	010710	020105	020040	020040	
3360	010716	030050	030455	020062	
3361	010724	042504	027103	020051	
3362	010732	006472	005012		
3363	010736	051124	041501	004513	.ASCII /TRACK DRIVE OVRWRT OVRWRT READ READ/<15><12>
3364	010744	051104	053111	004505	
3365	010752	053117	053522	052122	
3366	010760	047411	051126	051127	
3367	010766	004524	042522	042101	
3368	010774	051011	040505	006504	
3369	011002	012			
3370	011003	116	046525	051011	.ASCII /NUM READ SCORE SCORE SCORE SCORE/<15><12>
3371	011010	040505	004504	041523	
3372	011016	051117	004505	041523	
3373	011024	051117	004505	041523	
3374	011032	051117	004505	041523	
3375	011040	051117	006505	012	
3376	011045	011	047411	051506	.ASCIZ / OFST- OFST+ OFST- OFST+/<15><12><12>
3377	011052	026524	047411	051506	
3378	011060	025524	047411	051506	
3379	011066	026524	047411	051506	
3380	011074	025524	005015	000012	
3381	011102	047440	051126	051127	OAVERG: .ASCIZ / OVRWRT AVG = /
3382	011110	020124	053101	020107	
3383	011116	020075	000		

3384	011121	040	042522	042101	RAVERG: .ASCIZ / READ AVG = /
3385	011126	040440	043526	020040	
3386	011134	036440	000040		
3387	011140	005015	025012	020052	ENDTST: .ASCIZ <15><12><12>/** END OF TESTING **/<15><12>
3388	011146	042440	042116	047440	
3389	011154	020106	042524	052123	
3390	011162	047111	020107	025040	
3391	011170	006452	000012		
3392	011174	020040	000011		TRKBUF: .ASCIZ / /
3393	011200	020040	004440	000	DRVBUF: .ASCIZ / /
3394	011205	123	046105	004506	SELF: .ASCIZ /SELF /
3395	011212	000			
3396	011213	011	000		TAB: .ASCIZ / /
3397	011215	123	051127	036440	SWRMSG: .ASCIZ /SWR = /
3398	011222	000040			
3399	011224	020040	042516	020127	NEWMSG: .ASCIZ / NEW = /
3400	011232	020075	000		
3401	011235	015	042012	044522	DRVSEQ: .ASCIZ <15><12>/DRIVE(S) = /
3402	011242	042526	051450	020051	
3403	011250	020075	000		
3404	011253	104	044522	042526	BADDRV: .ASCIZ /DRIVE /
3405	011260	020040	000040		
3406	011264	047516	026516	054105	NXDRIV: .ASCIZ /NON-EXISTENT/<15><12>
3407	011272	051511	042524	052116	
3408	011300	005015	000		
3409	011303	116	052117	051040	NTREDY: .ASCIZ /NOT READY/<15><12>
3410	011310	040505	054504	005015	
3411	011316	000			
3412	011317	127	044522	042524	WRTLOK: .ASCIZ /WRITE-LOCKED/<15><12>
3413	011324	046055	041517	042513	
3414	011332	006504	000012		
3415	011336	047514	042101	042105	ALNPAK: .ASCIZ /LOADED WITH ALIGN PACK/<15><12>
3416	011344	053440	052111	020110	
3417	011352	046101	043511	020116	
3418	011360	040520	045503	005015	
3419	011366	000			
3420	011367	111	020106	054130	REPLPK: .ASCIZ /IF XXDP PACK ON DRV 0, TYPE 'Y <CR>', & REPLACE IT : /
3421	011374	050104	050040	041501	
3422	011402	020113	047117	042040	
3423	011410	053122	030040	020054	
3424	011416	054524	042520	021040	
3425	011424	020131	041474	037122	
3426	011432	026042	023040	020040	
3427	011440	042522	046120	041501	
3428	011446	020105	052111	035040	
3429	011454	000040			
3430	011456	005015	025012	020052	NODRTS: .ASCII <15><12><12>/** NO DRIVES TO TEST/<15><12>
3431	011464	047516	042040	044522	
3432	011472	042526	020123	047524	
3433	011500	052040	051505	006524	
3434	011506	012			
3435	011507	120	042522	051523	CNTRDY: .ASCIZ /PRESS 'CONT' WHEN RDY/<15><12>
3436	011514	021040	047503	052116	
3437	011522	020042	044127	047105	
3438	011530	051040	054504	005015	
3439	011536	000			

CZR6QCO RK6 DR CPT PROG MACY11 30A(1052) 02-JUL-80 09:52 PAGE 71
CZR6QC.P11 02-JUL-80 09:50

PROGRAM SPECIFIC RESERVED LOCATIONS

SEQ 0069

3440	011537	110	046101	020124	HLTRQD: .ASCIZ /HALT REQUESTED/<15><12>
3441	011544	042522	052521	051505	
3442	011552	042524	006504	000012	
3443	011560	051104	053111	020105	DROXDP: .ASCIZ /DRIVE 0 IS LOAD MEDIUM/<15><12>
3444	011566	020060	051511	046040	
3445	011574	040517	020104	042515	
3446	011602	044504	046525	005015	
3447	011610	000			
3448	011611	015	025012	020052	DROPDR: .ASCIZ <15><12>/** DROPPING DRIVE - 20 ERRORS/<15><12>
3449	011616	042040	047522	050120	
3450	011624	047111	020107	051104	
3451	011632	053111	020105	020055	
3452	011640	030062	042440	051122	
3453	011646	051117	006523	000012	
3454	011654	005015	052012	051505	TSTDRN: .ASCII <15><12><12>/TESTING DRIVE /
3455	011662	044524	043516	042040	
3456	011670	044522	042526	040	
3457	011675	040	005015	000	DRVNO: .ASCIZ / /<15><12>
3458	011701	104	044522	042526	DRIV: .ASCIZ /DRIVE /
3459	011706	000040			
3460	011710	040503	052122	020056	CART: .ASCIZ /CART. /
3461	011716	000			
3462	011717	123	051105	020056	SERNM: .ASCIZ /SER. NO. /
3463	011724	047516	020056	000040	
3464	011732	005015	040506	052103	FACTBS: .ASCIZ <15><12>/FACTORY /
3465	011740	051117	020131	000	
3466	011745	012	047523	052106	SOFTBS: .ASCII <12>/SOFTWARE /
3467	011752	040527	042522	040	
3468	011757	102	042101	051440	BDSECT: .ASCIZ /BAD SECTORS :/<15><12>
3469	011764	041505	047524	051522	
3470	011772	035040	005015	000	
3471	011777	040	047040	047117	NOFALS: .ASCIZ / NONE/
3472	012004	000105			
3473					
3474	012006	025052	020040	040503	BAD632: .ASCIZ /** CANNOT READ BAD SECTOR TRACK!/<15><12>
3475	012014	047116	052117	051040	
3476	012022	040505	020104	040502	
3477	012030	020104	042523	052103	
3478	012036	051117	052040	040522	
3479	012044	045503	006441	000012	
3480	012052	041536	005015	000	CNTRLC: .ASCIZ /^C/<15><12>
3481	012057	136	006532	000012	CNTRLZ: .ASCIZ /^Z/<15><12>
3482	012064	051136	005015	000	CNTRLR: .ASCIZ /^R/<15><12>
3483	012071	136	006525	000012	CNTRLU: .ASCIZ /^U/<15><12>
3484	012076	043536	005015	000	CNTRLG: .ASCIZ /^G/<15><12>
3485	012103	015	005012	000	CR2LF: .ASCIZ <15><12><12>
3486	012107	134	000		BKSLSH: .ASCIZ /\ /
3487	012111	054	000		COMMA: .ASCIZ /, /
3488	012113	040	040		SPACE6: .ASCII / /
3489	012115	040			SPACE4: .ASCII / /
3490	012116	040			SPACE3: .ASCII / /
3491	012117	040			SPACE2: .ASCII / /
3492	012120	000040			SPACE1: .ASCIZ / /
3493					.EVEN
3494	012122	020040	000075		PRMBUF: .ASCIZ / =/
3495	012126	047527	042122	000040	WORDSP: .ASCIZ /WORD /

```

3496 012134 036440 000040      EQUALS: .ASCIZ / = /
3497 012140 020052 000      PROMPT: .ASCIZ /* /
3498 012143 076 000040      PRMPSP: .ASCIZ /> /
3499
3500      .EVEN
3501
3502
3503
3504
3505 012146 105037 003126      DFSTRT: CLRB      MDFLAG      ;SET FLAG FOR 200 START
3506 012152 000403      BR      P1SET      ;CONTINUE
3507
3508 012154 112737 000001 003126      P1STRT: MOV      #1,MDFLAG      ;SET FLAG FOR 204 START
3509 012162 112737 000061 003155      P1SET:  MOV      #'1,PASSNO      ;SET PASS NO. = 1
3510 012170 000406      BR      CMSTRT      ;CONTINUE
3511
3512 012172 112737 000002 003126      P2STRT: MOV      #2,MDFLAG      ;SET FLAG FOR 220 START
3513 012200 112737 000062 003155      MOV      #'2,PASSNO      ;SET PASS NO. = 2
3514
3515 012206 012737 000340 177776      CMSTRT: MOV      #PR7,@#PS      ;BLOCK ALL INTERRUPTS
3516      .SBTTL INITIALIZE THE COMMON TAGS
3517      ;;CLEAR THE COMMON TAGS (%CMTAG) AREA
3518 012214 012706 001100      MOV      #CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
3519 012220 005026      CLR      (R6)+      ;;CLEAR MEMORY LOCATION
3520 012222 022706 001140      CMP      #SWR,R6 ;;DONE?
3521 012226 001374      BNE      -6      ;;LOOP BACK IF NO
3522 012230 012706 001100      MOV      #STACK,SP      ;;SETUP THE STACK POINTER
3523      ;;INITIALIZE A FEW VECTORS
3524 012234 012737 044574 000020      MOV      #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
3525 012242 012737 000340 000022      MOV      #340,@#IOTVEC+2 ;;LEVEL 7
3526 012250 012737 044070 000030      MOV      #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
3527 012256 012737 000340 000032      MOV      #340,@#EMTVEC+2 ;;LEVEL 7
3528 012264 012737 045306 000034      MOV      #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
3529 012272 012737 000340 000036      MOV      #340,@#TRAPVEC+2;LEVEL 7
3530 012300 012737 044440 000024      MOV      #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
3531 012306 012737 000340 000026      MOV      #340,@#PWRVEC+2 ;;LEVEL 7
3532 012314 013737 017220 017212      MOV      SENDCT,$EOPCT      ;;SETUP END-OF-PROGRAM COUNTER
3533 012322 005037 001316      CLR      $ESCAPE      ;;CLEAR THE ESCAPE ON ERROR ADDRESS
3534 012326 112737 000001 001115      MOV      #1,$ERMAX      ;;ALLOW ONE ERROR PER TEST
3535 012334 012737 012334 001106      MOV      #,$SLPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
3536 012342 012737 012342 001110      MOV      #,$SLPERR      ;;SETUP THE ERROR LOOP ADDRESS
3537      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3538      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
3539 012350 013746 000004      MOV      @ERRVEC,-(SP)      ;;SAVE ERROR VECTOR
3540 012354 012737 012410 000004      MOV      #64$,@ERRVEC      ;;SET UP ERROR VECTOR
3541 012362 012737 177570 001140      MOV      #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
3542 012370 012737 177570 001142      MOV      #DDISP,DISPLAY      ;;AND A HARDWARE DISPLAY REGISTER
3543 012376 022777 177777 166534      CMP      #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
3544 012404 001012      BNE      66$      ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3545      ;;AND THE HARDWARE SWR IS NOT = -1
3546 012406 000403      BR      65$      ;;BRANCH IF NO TIMEOUT
3547 012410 012716 012416      64$:  MOV      #65$, (SP)      ;;SET UP FOR TRAP RETURN
3548 012414 000002      RTI
3549 012416 012737 000176 001140      65$:  MOV      #SWREG,SWR      ;;POINT TO SOFTWARE SWR
3550 012424 012737 000174 001142      MOV      #DISPREG,DISPLAY
3551 012432 012637 000004      66$:  MOV      (SP)+,@ERRVEC      ;;RESTORE ERROR VECTOR
    
```



```

3552
3553 012436 005037 001336          CLR    $PASS          ;;CLEAR PASS COUNT
3554 012442 132737 000200 001351  BITB   #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
3555 012450 001403                BEQ    67$           ;;YES,USE NON-APT SWITCH
3556 012452 012737 001352 001140  MOV    #$$SWREG,$SWR  ;;NO,USE APT SWITCH REGISTER
3557 012460                67$:
3558 012460 000005                RESET          ;CLEAR THE UNIBUS
3559 012462 012737 000006 000004  MOV    #6,@#ERRVEC   ;SET TIME-OUT VECTOR
3560 012470 005037 000006          CLR    @#ERRVEC+2
3561 012474 105037 003143          CLRB   TSTTYP        ;CLEAR OFFSET TEST IDENTIFIER
3562 012500 105037 003137          CLRB   DRVERS        ;CLEAR ERROR COUNT FOR CURRENT DRIVE
3563 012504 105037 003135          CLRB   ERRCNT        ;CLEAR ERROR RETRY COUNT
3564 012510 105037 001102          CLRB   $TSTNM        ;SET TEST NO. = 0
3565 012514 112737 000001 003152  MOVB   #1,HLPOVL     ;SET HLP FILE OVLD INDICTR
3566 012522 104401 007572          TYPE   ,DZR6Q       ;TYPE PROGRAM I.D.
3567 012526 105737 003153          TSTB   NOTYPE        ;SEE IF OPERATOR NOTE SHOULD BE TYPED
3568 012532 001004                BNE    40$           ;BR IF NOT
3569 012534 104401 054330          TYPE   ,NOTMSG       ;TYPE OPERATOR NOTE
3570 012540 105237 003153          INCB   NOTYPE        ;SET FLAG FOR NEXT TIME
3571 012544 012737 020604 000060 40$:  MOV    #KBDHDL,@#TKVEC ;LOAD VECTOR FOR TTY KBD
3572 012552 012737 000200 000062  MOV    #PR4,@#TKVEC+2 ;SET KBD PRIORITY = 4
3573 012560 013701 003050          MOV    RKVEC,R1      ;ADDR. OF RK06 VECTOR STORAGE
3574 012564 012721 034710          MOV    #1,INTR,(R1)+ ;SET IT TO RK06 HANDLER
3575 012570 013711 003052          MOV    RKPRI,(R1)    ;SET RK06 PRIORITY
3576 012574 105037 003127          CLRB   TSTING        ;CLEAR 'RUNNING TESTS' FLAG
3577 012600 012737 000000 177776  MOV    #PRO,@#PS     ;ALLOW ALL INTERRUPTS AGAIN
3578 012606 012701 005162          MOV    #PRVCMO,R1   ;ZERO OUT PREVIOUS COMMAND
3579 012612 012700 000006          MOV    #6,R0        ;SET COUNTER
3580 012616 005021                42$:  CLR    (R1)+         ;ZERO A WORD
3581 012620 005300                DEC    R0
3582 012622 001375                BNE    42$           ;BR IF NOT DONE YET
3583 012624 005037 005432          CLR    STALLS        ;DON'T ALLOW STALLS YET
3584 012630 004737 021106          JSR    PC,GTSWRG     ;OPEN SOFTWARE SWR FOR MODIFICATION
3585 012634 012737 176543 042336 44$:  MOV    #176543,$HINUM ;INIT. PSEUDO-RANDOM NOS.
3586 012642 012737 123456 042340  MOV    #123456,$LONUM
3587 012650 004737 024154          JSR    PC,GETTYP
3588 012654 105737 003126          TSTB   MDFLAG        ;SEE IF DEFAULT MODE
3589 012660 001034                BNE    ALLSYS        ;BR IF NOT DEFAULT MODE
3590 012662 013700 001400          MOV    $VECT1,R0    ;GET APT RK06 VECTOR AND PRTY
3591 012666 013737 001404 003046  MOV    $BASE,RKBAS   ;GET APT RK06 BASE ADDRESS
3592 012674 132737 000200 001351  BITB   #BIT7,$ENVM   ;SEE IF NO SIZING
3593 012702 001005                BNE    18$           ;BR IF NO SIZING
3594 012704 012700 120210          MOV    #AVECT1,R0   ;GET DEFAULT VECTOR AND PRIORITY
3595 012710 012737 177440 003046  MOV    #ABASE,RKBAS  ;GET DEFAULT BASE ADDRESS
3596 012716 110037 003050 18$:  MOVB   R0,RKVEC      ;STORE VECTOR
3597 012722 105037 003051          CLRB   RKVEC+1      ;CLEAR HI BYTE
3598 012726 000300                SWAB   R0            ;GET PRTY INTO BITS 5-7
3599 012730 042700 177437          BIC    #177437,R0   ;CLEAR OTHER BITS
3600 012734 010037 003052          MOV    R0,RKPRI     ;STORE RK06 PRIORITY
3601 012740 112737 000101 005510  MOVB   #'A,SUBSYS    ;SET SUBSYSTEM NAME = A
3602 012746 000137 013744          JMP    DFLDR        ;GO CHECK ALL DRIVES
3603
3604
3605
3606
3607

```

 :*

```

3608 .SBTTL TEST 0 PARAMETER INPUT ROUTINES
3609 :*
3610 :*****
3611 :READ AND STORE ALL SUBSYSTEMS AND DRIVES TO TEST
3612 012752 104401 001325 ALLSYS: TYPE ,SCLRF ;TYPE <CR>,<LF>
3613 012756 105037 001102 CLR B $STNM ;SET TEST NO. = 0
3614 012762 112703 000101 MOV B #'A,R3 ;INIT SUBSYS INDICATOR
3615 012766 012701 005512 MOV #DRVLST,R1 ;GET DRIVE LIST ADDRESS
3616 012772 012700 000021 MOV #17.,R0 ;CLEAR OUT THE DRIVE LIST
3617 012776 005021 2$: CLR (R1)+
3618 013000 005300 DEC R0
3619 013002 001375 BNE 2$
3620 013004 012704 005512 MOV #DRVLST,R4 ;GET STARTING ADRS OF DRIVE LIST
3621 :ASK FOR DRIVES TO TEST ON THIS SUBSYSTEM
3622 013010 110337 010062 ALLSY1: MOV B R3,SYSDRV+7 ;SET SUBSYS NAME FOR PRINTOUT
3623 013014 104401 010053 6$: TYPE ,SYSDRV ;ASK FOR DRIVES ON THIS SUBSYSTEM
3624 013020 004737 023402 JSR PC,RDCHRS ;READ INPUT STRING
3625 013024 012752 ALLSYS ;(^C) RETURN ADDRESS
3626 013026 012752 ALLSYS ;(^Z) RETURN ADDRESS
3627 013030 013014 6$ ;(^U) RETURN ADDRESS
3628 013032 005700 TST R0 ;SEE IF NULL INPUT
3629 013034 001767 BEQ 6$ ;BR TO ASK AGAIN
3630 013036 022700 000017 CMP #15.,R0 ;SEE IF TOO MANY CHARS TYPED
3631 013042 002005 BGE 12$ ;BR IF NOT TOO MANY
3632 013044 104401 005212 10$: TYPE ,BUFFO ;ECHO BAD INPUT
3633 013050 104401 001324 TYPE ,SQUES ;TYPE <?> AND <CR>,<LF>
3634 013054 000757 BR 6$ ;GO ASK AGAIN
3635 013056 005001 12$: CLR R1 ;CLEAR CHAR POINTER
3636 013060 126127 005212 000060 14$: CMPB BUFFO(R1),#'0 ;SEE IF DRIVE NO. < 0
3637 013066 103766 BLO 10$ ;BR TO ECHO BAD INPUT
3638 013070 126127 005212 000067 CMPB BUFFO(R1),#'7 ;SEE IF CHAR > 7
3639 013076 101362 BHI 10$ ;BR TO ECHO BAD INPUT
3640 013100 005201 INC R1 ;INCR CHAR POINTER
3641 013102 020100 CMP R1,R0 ;SEE IF MORE CHARS TO CHECK
3642 013104 001406 BEQ 16$ ;BR IF ALL CHARS CHECKED
3643 013106 126127 005212 000054 CMPB BUFFO(R1),#',' ;SEE IF THIS IS A COMMA
3644 013114 001353 BNE 10$ ;BR IF NOT A COMMA
3645 013116 005201 INC R1 ;INCR CHAR POINTER
3646 013120 000757 BR 14$ ;BR TO CONTINUE CHECKING CHARS
3647 :PUT DRIVES FOR THIS SUBSYSTEM INTO SCRATCH LIST
3648 013122 012701 005556 16$: MOV #SCLST,R1 ;GET SCRATCH DRIVE LIST ADRS
3649 013126 005021 CLR (R1)+ ;CLEAR OUT SCRATCH DRIVE LIST
3650 013130 005021 CLR (R1)+
3651 013132 005021 CLR (R1)+
3652 013134 005011 CLR (R1)
3653 013136 005000 CLR R0 ;INIT CHAR POINTER
3654 013140 116001 005212 18$: MOV B BUFFO(R0),R1 ;GET A DRIVE NO.
3655 013144 042701 000060 BIC #'0,R1 ;STRIP ASCII BITS
3656 013150 116061 005212 005556 MOV B BUFFO(R0),SCLST(R1) ;SET ASCII DRV NO.
3657 013156 062700 000002 ADD #2,R0 ;POINT TO NEXT DRIVE NO.
3658 013162 105760 005211 TSTB BUFFO-(R0) ;SEE IF NULL CHAR YET
3659 013166 001364 BNE 18$ ;BR IF NOT DONE YET
3660 :LOAD DRIVES FROM SCRATCH LIST INTO TOTAL DRIVE LIST
3661 013170 005000 ALLSY3: CLR R0 ;INIT SCRATCH DRV LIST POINTER
3662 013172 116014 005556 ALLSY2: MOV B SCLST(R0),(R4) ;MOVE THIS DRIVE INTO ACTUAL LIST
3663 013176 001411 BEQ 22$ ;BR IF DRIVE NOT MARKED
  
```

```

3664 013200 005204          INC      R4          ;INCR DRIVE LIST POINTER
3665 013202 110324          MOVVB   R3,(R4)+     ;PUT SUBSYS LETTER INTO HI BYTE
3666 013204 020427 005554    CMP     R4,#DRVLST+34. ;SEE IF DRIVE LIST OVERFLOW
3667 013210 103404          BLO    22$          ;BR IF NO OVERFLOW YET
3668 013212 104401 010215    TYPE   ,DROVFL      ;TYPE 'MORE THAN 16 DRIVES ENTERED !'
3669 013216 000137 012752    JMP    ALLSYS       ;GO START OVER, ASKING FOR DRIVES
3670 013222 005200          INC     R0          ;INCR SCRATCH DRIVE LIST POINTER
3671 013224 020027 000010    22$:   CMP     R0,#8.   ;SEE IF DONE WITH THIS SUBSYS
3672 013230 103760          BLO    ALLSY2       ;BR IF NOT YET
3673          ;ECHO DRIVES TO TEST ON THIS SUBSYSTEM
3674 013232 104401 010100    TYPE   ,WILTST      ;TYPE 'WILL TEST DRIVE(S) '
3675 013236 005037 005444    CLR    SCRACH       ;INIT CHAR OUTPUT BUF
3676 013242 005000          CLR    R0          ;INIT SCRATCH LIST POINTER
3677 013244 005001          CLR    R1          ;INIT COUNT OF LISTED DRIVES FOR SUBSYS
3678 013246 116037 005556 005444 23$:   MOVVB   SCRLST(R0),SCRACH ;SEE IF THIS DRIVE LISTED
3679 013254 001407          BEQ    26$          ;BR IF DRIVE NOT LISTED
3680 013256 005701          TST    R1          ;SEE IF FIRST TIME HERE
3681 013260 001402          BEQ    24$          ;BR IF FIRST TIME
3682 013262 104401 012111    TYPE   ,COMMA       ;TYPE A COMMA
3683 013266 104401 005444    24$:   TYPE   ,SCRACH     ;TYPE A DRIVE NO.
3684 013272 005201          INC     R1          ;INCR. COUNT OF LISTED DRIVES FOR THIS SUBSYS
3685 013274 005200          INC     R0          ;INCR LIST POINTER
3686 013276 020027 000010    26$:   CMP     R0,#8.   ;SEE IF DONE CHECKING LIST YET
3687 013302 002761          BLT    23$          ;BR IF NOT DONE YET
3688 013304 110337 010137    MOVVB   R3,ONSUBS+11. ;SET SUBSYS NO. IN MSG
3689 013310 104401 010124    TYPE   ,ONSUBS      ;TYPE 'ON SUBSYSTEM '
3690 013314 005203          INC     R3          ;INCR SUBSYSTEM NAME
3691 013316 020327 000120    CMP     R3,#'P      ;SEE IF NAME > P
3692 013322 101035          BHI    ASKSYS       ;BR IF YES
3693 013324 105737 003126    TSTB   MDFLAG       ;SEE IF 200 START
3694 013330 001002          BNE    28$          ;BR IF NOT
3695 013332 000137 013704    JMP    DRVTST       ;START PASS 1
3696          ;ASK IF THERE IS ANOTHER SUBSYSTEM TO SPECIFY
3697 013336 104401 010144    28$:   TYPE   ,ANOTHR     ;ASK FOR ANOTHER SUBSYS
3698 013342 004737 023402    JSR    PC,RDCHRS    ;READ RESPONSE
3699 013346 012752          ALLSYS              ;(^C) RETURN ADDRESS
3700 013350 012752          ALLSYS              ;(^Z) RETURN ADDRESS
3701 013352 013336          28$                ;(^U) RETURN ADDRESS
3702 013354 005700          TST    R0          ;SEE IF NULL INPUT
3703 013356 001417          BEQ    ASKSYS       ;BR IF NULL INPUT, TO ASSUME <N>
3704 013360 022737 000131 005212    CMP     #'Y,BUFFO   ;SEE IF <Y> TYPED
3705 013366 001002          BNE    30$          ;BR IF NOT <Y>
3706 013370 000137 013010          JMP    ALLSY1       ;GO ASK FOR DRIVES ON NEXT SUBSYS
3707 013374 022737 000116 005212 30$:   CMP     #'N,BUFFO   ;SEE IF <N> TYPED
3708 013402 001405          BEQ    ASKSYS       ;BR IF <N>
3709 013404 104401 005212    TYPE   ,BUFFO       ;ECHO BAD INPUT
3710 013410 104401 001324    TYPE   ,SQUES       ;TYPE <?> AND <CR>,<LF>
3711 013414 000750          BR     28$          ;GO ASK AGAIN
3712
3713          ;204 OR 220 START - GET NAME OF SUBSYSTEM TO TEST NEXT
3714 013416 105037 001102    ASKSYS: CLR    $TSTNM   ;SET TEST NO. = 0
3715 013422 104401 007722    TYPE   ,TYPYSYS    ;ASK FOR NEXT SUBSYS TO TEST
3716 013426 004737 023402    JSR    PC,RDCHRS    ;READ RESPONSE
3717 013432 012752          ALLSYS              ;(^C) RETURN ADDRESS
3718 013434 013416          ASKSYS              ;(^Z) RETURN ADDRESS
3719 013436 013416          ASKSYS              ;(^U) RETURN ADDRESS
    
```

```

3720 013440 005700          TST      RO          ;SEE IF NULL INPUT
3721 013442 001765          BEQ      ASKSYS      ;BR IF NULL, TO ASK AGAIN
3722 013444 023727 005212 000101  CMP      BUFFO,#'A   ;SEE IF A CHAR < A TYPED
3723 013452 103005          BHIS     6$          ;BR IF NOT
3724 013454 104401 005212 4$:    TYPE     ,BUFFO     ;ECHO BAD INPUT
3725 013460 104401 001324          TYPE     ,SQUES     ;TYPE <?> AND <CR>,<LF>
3726 013464 000754          BR       ASKSYS      ;GO ASK AGAIN
3727 013466 023727 005212 000120 6$:    CMP      BUFFO,#'P   ;SEE IF A CHAR > P TYPED
3728 013474 101367          BHI     4$          ;BR IF YES
3729 013476 113737 005212 005510  MOVB     BUFFO,SUBSYS ;STORE CURRENT SUBSYSTEM NAME
3730 013504 012700 005513          MOV      #DRVLST+1,RO ;SET DRIVE LIST POINTER
3731 013510 122037 005510 8$:    CMPB    (RO)+,SUBSYS ;SEE IF A DRIVE LISTED FOR THIS SUBSYS
3732 013514 001410          BEQ     9$          ;BR IF YES
3733 013516 005200          INC     RO          ;INCR POINTER
3734 013520 020027 005553          CMP     RO,#DRVLST+33. ;SEE IF WHOLE LIST CHECKED YET
3735 013524 103771          BLO     8$          ;BR IF NOT YET
3736 013526 104401 010622          TYPE     ,NODRLS    ;TYPE 'NO DRIVES LISTED FOR SUBSYS''
3737 013532 000137 013416          JMP     ASKSYS      ;GO ASK FOR A SUBSYS AGAIN
3738 013536 162700 000002 9$:    SUB     #2,RO       ;POINT TO LOW BYTE OF LIST ENTRY
3739 013542 010037 005554          MOV     RO,DRVPTTR  ;STORE POINTER
3740                                ;OPEN RK06 UNIBUS ADDRESS FOR MODIFICATION
3741 013546 013746 003046          MOV     RKBAS,-(SP)  ;PUT OLD VALUE ON STACK
3742 013552 104401 007630          TYPE     ,RKBADR    ;TYPE 'RK06 BUS ADR = ''
3743 013556 004737 020776          JSR     PC,GETPRM   ;TYPE OLD, GET NEW RKBAS VALUE
3744 013562 012637 003046          MOV     (SP)+,RKBAS ;STORE NEW VALUE
3745                                ;OPEN RK06 VECTOR ADDRESS FOR MODIFICATION
3746 013566 013746 003050          MOV     RKVEC,-(SP) ;PUT OLD VALUE ON STACK
3747 013572 104401 007654          TYPE     ,RKVADR    ;TYPE 'RK06 VEC ADR = ''
3748 013576 004737 020776          JSR     PC,GETPRM   ;TYPE OLD, GET NEW RKVEC VALUE
3749 013602 012637 003050          MOV     (SP)+,RKVEC ;STORE NEW VALUE
3750                                ;OPEN RK06 INTERRUPT HANDLER PRIORITY LEVEL FOR MODIFICATION
3751 013606 013746 003052 11$:   MOV     RKPRI,-(SP)  ;GET OLD VALUE OF PRIORITY
3752 013612 006316          ASL     (SP)        ;GET IT INTO BITS 0-2
3753 013614 006316          ASL     (SP)
3754 013616 006316          ASL     (SP)
3755 013620 000316          SWAB    (SP)
3756 013622 104401 007677          TYPE     ,RKPRTY    ;TYPE 'RK06 PRIORITY = ''
3757 013626 004737 020776          JSR     PC,GETPRM   ;TYPE OLD, GET NEW RKPRI VALUE
3758 013632 012600          MOV     (SP)+,RO
3759 013634 020027 000004          CMP     RO,#4       ;SEE IF AT LEAST LEVEL 4
3760 013640 002414          BLT     12$        ;BR IF NEW VALUE TOO SMALL
3761 013642 020027 000007          CMP     RO,#7       ;SEE IF LEVEL 7 OR LESS
3762 013646 003011          BGT     12$        ;BR IF NEW VALUE TOO LARGE
3763 013650 000300          SWAB    RO         ;GET PRIORITY INTO BITS 5-7
3764 013652 006200          ASR     RO
3765 013654 006200          ASR     RO
3766 013656 006200          ASR     RO
3767 013660 010037 003052          MOV     RO,RKPRI    ;STORE NEW VALUE
3768 013664 104401 001325          TYPE     ,SCLF
3769 013670 000405          BR      DRVTST
3770 013672 104401 005212 12$:   TYPE     ,BUFFO     ;ECHO BAD INPUT
3771 013676 104401 001324          TYPE     ,SQUES
3772 013702 000741          BR      11$        ;GO ASK AGAIN
3773
3774 013704 113737 003155 010301 DRVTST: MOVB    PASSNO,SRPASS+18. ;SET PASS NO. FOR PRINTOUT
3775 013712 104401 010257          TYPE     ,SRPASS    ;TYPE 'STARTING PASS X''

```

```

3776 013716 122737 000061 003155      CMPB  #'1,PASSNO      ;SEE IF PASS 1
3777 013724 001002                BNE   2$              ;BR IF NOT PASS 1
3778 013726 000137 014316                JMP   TST1            ;START PASS 1
3779 013732 112737 000004 001102 2$:  MOVB  #4,$TSTNM      ;SET TEST NUMBER = 4
3780 013740 000137 015116                JMP   TST5            ;START PASS 2
3781
3782
3783      ;ADR 200 START - COMPILE SCRATCH LIST OF DRIVES TO SIZE
3784 013744 005000      DFLTDR: CLR   RO      ;CLEAR DRIVE NUMBER
3785 013746 013703 001406      MOV   $DEVN,R3      ;GET APT DEVICE MAP
3786 013752 132737 000200 001351      BITB  #BIT7,$ENVM   ;SEE IF NO SIZING
3787 013760 001002                BNE   1$              ;BR IF NO SIZING
3788 013762 012703 000377      MOV   #377,R3      ;SET UP FOR SIZING
3789 013766 012701 000001 1$:  MOV   #BIT0,R1      ;SET BIT POINTER
3790 013772 105060 005556 2$:  CLRB  SCRLST(RO)   ;INITIALIZE DRIVE ENTRY
3791 013776 030103                BIT   R1,R3          ;SEE IF THIS DRIVE IS REQUESTED
3792 014000 001405                BEQ   4$              ;BR IF NOT
3793 014002 110060 005556      MOVB  RO,SCRLST(RO) ;MARK DRIVE IN SCRATCH LIST
3794 014006 152760 000060 005556      BISB  #'0,SCRLST(RO);CONVERT NO. TO ASCII
3795 014014 006301 4$:  ASL   R1              ;SHIFT BIT POINTER
3796 014016 005200                INC   RO              ;INCREMENT DRIVE NUMBER
3797 014020 022700 000010      CMP   #10,RO        ;SEE IF DONE MARKING ALL DRIVES
3798 014024 001362                BNE   2$              ;BR IF NOT DONE YET
3799 014026 132737 000200 001351      BITB  #BIT7,$ENVM   ;SEE IF PROGRAM SHOULD SIZE
3800 014034 001416                BEQ   8$              ;BR IF YES
3801 014036 012704 005512 10$:  MOV   #DRVLIST,R4   ;GET DRIVE LIST ADDRESS
3802 014042 113703 005510      MOVB  SUBSYS,R3     ;GET SUBSYS NAME IN R3
3803 014046 010401                MOV   R4,R1          ;GET DRIVE LIST ADRS
3804 014050 010437 005554      MOV   R4,DRVPTR     ;SET POINTER TO FIRST DRIVE TO TEST
3805 014054 012700 000021      MOV   #17,RO        ;SET COUNTER
3806 014060 005021 11$:  CLR   (R1)+          ;CLEAR A DRIVE LIST WORD
3807 014062 005300                DEC   RO              ;DECREMENT COUNTER
3808 014064 001375                BNE   11$            ;BR IF NOT DONE YET
3809 014066 000137 013170      JMP   ALLSY3        ;GO MOVE SCRATCH LIST INTO LIST
3810      ;REMOVE NON-EXISTENT DRIVES FROM SCRATCH LIST
3811 014072 005000 8$:  CLR   RO              ;CLEAR DRIVE NUMBER
3812 014074 105760 005556 12$:  TSTB  SCRLST(RO)   ;SEE IF THIS DRIVE IS MARKED IN SCRATCH LIST
3813 014100 001475                BEQ   14$            ;BR IF NOT MARKED
3814 014102 010037 005430      MOV   RO,DRIVE      ;SET DRIVE NUMBER FOR SCNDRV
3815 014106 004737 021140      JSR   PC,INITSS     ;CLEAR DRIVER AND INIT S.S.
3816 014112 042712 000100 3$:  BIC   #IE,(R2)      ;DISABLE RK06 INTERRUPT
3817 014116 113737 005430 000010      MOVB  DRIVE,RKCS2   ;SEE IF RK07
3818 014124 105737 003133      TSTB  TYPFMT        ;BR IF YES
3819 014130 001003                BNE   5$              ;ELSE SET GO BIT FOR RK06 IN RKCS1
3820 014132 012712 000001      MOV   #GO,(R2)
3821 014136 000402                BR   7$
3822 014140 012712 002001 5$:  MOV   #<CDT!GO>,(R2);ELSE SET GO FOR RK07 IN RKCS1
3823 014144 005037 005444 7$:  CLR   SCRACH        ;CLEAR STALL CTR
3824 014150 005237 005444 15$:  INC   SCRACH        ;INCREMENT STALL COUNTER
3825 014154 001375                BNE   15$            ;STALL FOR SEVERAL MILLI-SEC
3826 014156 032762 001000 000010      BIT   #MDS,RKCS2(R2);SEE IF MDS ERROR
3827 014164 001417                BEQ   18$            ;BR IF NOT
3828 014166 112765 000101 000001      MOVB  #SELDRV,P.CMND(R5);SET CMND FOR ERROR REPORT
3829 014174 011265 000016      MOV   (R2),P.CS1(R5);GET RKCS1 FOR REPORT
3830 014200 004737 037632      JSR   PC,I.CST1     ;GET OTHER RK611 REGS FOR REPORT
3831 014204 004737 031212      JSR   PC,REPSUP     ;SET UP ERROR REPORT
  
```

```

3832 014210 104052          ERROR 52          :REPORT MDS ERROR
3833 014212 052737 000200 005424  BIS          #ABORT,RECODE    :SET ABORT FLAG
3834 014220 000137 033324          JMP          ALLTRM :GO ABORT TESTING
3835 014224 004737 021140          JSR          PC,INITSS :INIT THE S.S.
3836 014230 112765 000141 000001  MOVB        #RDSTAT,P.CMND(R5) :SET READ STATUS COMMAND
3837 014236 012737 020450 003056  MOV          #NEDHDL,A.ABNL :SET NED ABNORMAL RETURN ADR
3838 014244 012737 000377 005476  MOV          #377,NEWON :INIT ONLINE INDICATOR
3839 014252 004737 030124          JSR          PC,DRVCAL :READ DRIVE STATUS
3840 014256 012737 031472 003056  MOV          #ERRHDL,A.ABNL :RESTORE ABNORMAL RETURN ADDRESS
3841 014264 022737 000377 005476  CMP          #377,NEWON :SEE IF THIS DRIVE EXISTS
3842 014272 001006          BNE          16$      :BR IF NO
3843 014274 005200          14$: INC          RO      :RETURN HERE IF DRIVE IS USEABLE
3844 014276 022700 000010  CMP          #10,RO   :SEE IF DONE CHECKING LIST
3845 014302 001274          BNE          12$      :BR IF NOT DONE YET
3846 014304 000137 014036  JMP          10$      :GO BACK TO LIST SELECTED DRIVES
3847 014310 105060 005556  16$: CLRB        SCRLST(RO) :REMOVE INVALID DRIVE FROM LIST
3848 014314 000767          BR          14$      :CONTINUE CHECKING THE LIST

```

3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887

 :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.

 TST1: 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
JSR          PC,MTCART

```

 :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.

```

3888
3889 014346 000004
3890 014350 012737 000002 001334
3891 014356 004737 022020
3892 014362 004737 021140
3893 014366 004737 020744
3894 014372 112737 000001 003127
3895
3896 014400 004737 024460
3897
3898 014404 113700 003154
3899 014410 006300
3900 014412 006300
3901 014414 105737 003133
3902 014420 001011
3903 014422 016065 005566 000002
3904 014430 062700 000002
3905 014434 116065 005566 000004
3906 014442 000410
3907
3908 014444 016065 005666 000002 1$:
3909 014452 062700 000002
3910 014456 116065 005666 000004
3911
3912 014464 012765 053330 000010 2$:
3913 014472 012765 177400 000012
3914
3915 014500 112765 000123 000001 6$:
3916 014506 004737 030124
3917
3918 014512 112765 000131 000001
3919 014520 004737 030124
3920
3921 014524 105265 000005
3922 014530 126527 000005 000002
3923 014536 101760
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937 014540 000004
3938 014542 012737 000003 001334
3939 014550 004737 022020
3940 014554 004737 021140
3941 014560 004737 020744
3942 014564 004737 022620
3943 014570 005000

```

```

:*****
TST2:  SCOPE
      MOV      #2,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
      JSR      PC,SETUP      ;;SET UP FOR LOOP ON ERROR
      JSR      PC,INITSS     ;;INITIALIZE DRIVER PARAMS AND SUB-SYS
      JSR      PC,PREPKB     ;;PREPARE FOR POSSIBLE KBD INPUT
      MOVB     #1,TSTING     ;;ALLOW TTY ESCAPE
;LOAD RWBUF WITH DATA PATTERN
      JSR      PC,LODSEC
;SET UP PARAMETERS
      MOVB     LOGDRV,R0      ;GET LOGICAL DRIVE NUMBER
      ASL     R0              ;CONVERT IT TO INDEX
      ASL     R0
      TSTB     TYPFMT        ;SEE IF RK07
      BNE     1$            ;BR IF YES
      MOV     TABLEA(R0),P.CYLN(R5) ;GET CYL # FOR THIS DRIVE
      ADD     #2,R0          ;INCREMENT INDEX
      MOVB     TABLEA(R0),P.SECT(R5) ;GET SECTOR # FOR THIS DRIVE
      BR      2$
1$:   MOV     TABL7A(R0),P.CYLN(R5) ;GET RK07 TABLE
      ADD     #2,R0
      MOVB     TABL7A(R0),P.SECT(R5) ;GET SECTOR # FOR THE DRV
2$:   MOV     #RWBUF,P.BALO(R5) ;SET BUS ADDRESS
      MOV     #-400,P.WC(R5) ;SET WORD COUNT FOR 1 SECTOR
;WRITE THE DATA
6$:   MOVB     #WRDATA,P.CMND(R5) ;SET WRITE COMMAND
      JSR     PC,DRVCAL      ;WRITE THE SECTOR
;PERFORM WRITE CHECK OF SECTOR
      MOVB     #WRTCHK,P.CMND(R5) ;SET WRITE CHECK COMMAND
      JSR     PC,DRVCAL      ;PERFORM THE WRITE CHECK
;INCREMENT TRACK ADDRESS
      INCB     P.TRCK(R5) ;INCREMENT TRACK NO.
      CMPB     P.TRCK(R5),#2 ;SEE IF ALL SURFACES WRITTEN YET
      BLOS    6$           ;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
;* BE USED IN PASS 2.
:*****
TST3:  SCOPE
      MOV     #3,$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
      JSR     PC,INTBLD     ;LOAD INITIAL VALUES INTO TABLED
      CLR     R0            ;INITIALIZE TABLEC INDEX TO 0

```

```

3944 014572 105737 003133 4$: TSTB TYPFMT ;SEE IF RK07
3945 014576 001013 BNE 1$ ;BR IF YES
3946 014600 016065 006026 000002 MOV TABLEC(R0),P.CYLN(R5) ;GET START OVRWRT CYL FROM TABLEC
3947 014606 004737 023064 JSR PC,WRTBLK ;WRITE THIS OVRWRT CYL BLK ON ALL SURF'S
3948 014612 062700 000002 ADD #2,R0 ;INCREMENT TABLEC POINTER
3949 014616 016065 006026 000002 MOV TABLEC(R0),P.CYLN(R5) ;GET START COMPAT CYL FROM TABLEC
3950 014624 000412 BR 2$
3951
3952 014626 016065 006060 000002 1$: MOV TABL7C(R0),P.CYLN(R5)
3953 014634 004737 023064 JSR PC,WRTBLK
3954 014640 062700 000002 ADD #2,R0
3955 014644 016065 006060 000002 MOV TABL7C(R0),P.CYLN(R5)
3956 014652 004737 023064 2$: JSR PC,WRTBLK ;WRITE THIS COMPAT CYL BLK ON ALL SURF'S
3957 014656 062700 000002 ADD #2,R0 ;INCREMENT TABLEC POINTER
3958 014662 020027 000030 CMP R0,#30 ;SEE IF ON LAST CURRENT ZONE YET
3959 014666 002003 BGE 8$ ;BR IF YES
3960 014670 004737 023000 JSR PC,ROTBLD ;ROTATE TABLED SEVERAL SECTORS
3961 014674 000736 BR 4$ ;CONTINUE WRITING BLOCKS
3962 014676 004737 022620 8$: JSR PC,INTBLD ;RE-LOAD INITIAL VALUES INTO TABLED
3963 014702 105737 003133 TSTB TYPFMT ;SEE IF RK07
3964 014706 001004 BNE 3$ ;BR IF YES
3965 014710 016065 006026 000002 MOV TABLEC(R0),P.CYLN(R5) ;GET START OVRWRT CYL FROM TABLEC
3966 014716 000403 BR 5$
3967
3968 014720 016065 006060 000002 3$: MOV TABL7C(R0),P.CYLN(R5)
3969 014726 004737 023064 5$: JSR PC,WRTBLK ;WRITE THIS OVRWRT CYL BLK ON ALL SURF'S
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991

```

```

:*****
:*TEST 4 DISMOUNTING OF TEST CARTRIDGE IN PASS 1
:*
:* THE PROGRAM DIRECTS THE OPERATOR TO UNLOAD THE DRIVE CURRENTLY
:* UNDER TEST, AND TO REMOVE THE TEST PACK. THEN, THE PROGRAM
:* DOES ONE OF 4 THINGS :
:* (1) IF THERE IS ANOTHER DRIVE TO TEST ON THIS SUBSYSTEM, THE
:* PROGRAM JUMPS TO TEST 1 FOR THE NEXT DRIVE.
:* (2) IF THERE IS ONLY ONE SUBSYSTEM, (FROM 200 START), AND
:* IF PASS 1 HAS BEEN COMPLETED ON ALL DRIVES, THE PROGRAM STARTS
:* PASS 2 ON THE FIRST DRIVE, BY JUMPING TO TEST 5.
:* (3) IF THERE IS ANOTHER SUBSYSTEM TO PERFORM PASS 1 ON, THE PROGRAM
:* TELLS THE OPERATOR TO RESTART AT ADDRESS 204, AND THEN IT HALTS.
:* (4) IF THERE ARE NO MORE DRIVES TO TEST IN PASS 1 ON ANY
:* OTHER SUBSYSTEM, THE PROGRAM TELLS THE OPERATOR TO
:* RESTART AT ADDRESS 220 FOR PASS 2 ON THE NEXT SUBSYSTEM, AND
:* THEN IT HALTS.
:*****

```

```

3992 014732 000004 TST4: SCOPE
3993 014734 012737 000004 001334 MOV #4,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
3994 014742 004737 022020 JSR PC,SETUP ;:SET UP FOR LOOP ON ERROR
3995 014746 004737 021140 JSR PC,INITSS ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
3996 014752 004737 020744 JSR PC,PREPKB ;:PREPARE FOR POSSIBLE KBD INPUT
3997 014756 105037 003127 CLR B TSTING ;:DON'T ALLOW TTY ESCAPE
3998 014762 004737 022450 JSR PC,DMCART ;:DIRECT OPERATOR TO DISMOUNT PACK
3999 014766 062737 000002 005554 ADD #2,DRVPTN ;:INCREMENT DRIVE LIST POINTER

```



```

4000 014774 013700 005554      MOV      DRVPTR,RO
4001 015000 005710              TST      (RO)          ;SEE IF ALL DONE WITH PASS 1 YET
4002 015002 001420              BEQ      8$            ;BR IF DONE WITH PASS 1
4003 015004 126037 000001 005510  CMPB     1(RO),SUBSYS ;SEE IF DONE WITH THIS SUBSYS YET
4004 015012 001004              BNE      4$            ;BR IF DONE WITH THIS SUBSYS
4005 015014 105037 001102      CLRB     $TSTNM        ;CLEAR TEST NUMBER
4006 015020 000137 014316      JMP      TST1          ;DO PASS 1 ON NEXT DRIVE OF SUBSYS
4007 015024 116037 000001 010467 4$:      MOVB    1(RO),STR204+39. ;PUT SUBSYS NAME INTO MSG
4008 015032 104401 010420      5$:      TYPE     ,STR204      ;TYPE RESTART MSG FOR PASS 1 ON NEXT SUBSYS
4009 015036 000000              6$:      HALT                    ;HALT THE PROCESSOR SO THAT
4010 015040 000137 000204              JMP      204          ; OPERATOR CAN RESTART AT PROPER ADRS
4011 015044 012737 005512 005554 8$:      MOV      #DRVLIST,DRVPTR ;RE-INIT DRIVE LIST POINTER
4012 015052 105737 003126      TSTB    MDFLAG        ;SEE IF 200 START
4013 015056 001007              BNE      12$          ;BR IF NOT
4014 015060 112737 000062 010301  MOVB    #'2,SRPASS+18. ;SET PASS NO. = 2 FOR PRINTOUT
4015 015066 104401 010257      TYPE     ,SRPASS      ;TYPE "STARTING PASS 2"
4016 015072 000137 015116      JMP      TST5          ;GO START PASS 2
4017 015076 112737 000101 010544 12$:     MOVB    #'A,STR220+39. ;PUT SUBSYS NAME INTO MSG
4018 015104 104401 010475      TYPE     ,STR220      ;TYPE RESTART MSG FOR PASS 2 ON FIRST SUBSYS
4019 015110 000000              HALT                    ;HALT PROCESSOR SO THAT OPERATOR CAN RESTART
4020 015112 000137 000220              JMP      220          ; AT THE PROPER ADRS
    
```

4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055

 *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).

```

TST5:  SCOPE
4037 015116 000004              MOV      #5,$TESTN     ;:SET TEST NUMBER IN APT MAIL BOX
4038 015120 012737 000005 001334  JSR      PC,SETUP      ;:SET UP FOR LOOP ON ERROR
4039 015126 004737 022020      JSR      PC,INITSS    ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
4040 015132 004737 021140      JSR      PC,PREPKB    ;:PREPARE FOR POSSIBLE KBD INPUT
4041 015136 004737 020744              MOVB    #'2,PASSNO    ;:SET PASS NO. = 2
4042 015142 112737 000062 003155  JSR      PC,MT CART
4043 015150 004737 022066
    
```

 *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.

```

4056 :* THEN, EACH CYLINDER OVERWRITTEN IS READ BACK BY THIS DRIVE
4057 :* WITH THE FOLLOWING RANGE OF OFFSET VALUES : 100, 200, 300,
4058 :* 400,500,600,700,800,900,1000,1100,1200 MICRO-IN., IN BOTH
4059 :* THE (+) AND (-) DIRECTIONS.
4060 :* THE PROGRAM SCANS FOR DATA CHECK ERRORS DURING THIS READ, AND IF
4061 :* ONE OCCURS, THE PROGRAM DETERMINES WHICH DRIVE'S DATA HAD NOT
4062 :* BEEN CORRECTLY OVERWRITTEN, AND A SCORE FOR THAT DRIVE IS
4063 :* DECREMENTED. THEN, THE TRANSFER IS CONTINUED AT THE NEXT SECTOR,
4064 :* WITH THAT OFFSET VALUE. THE READS ARE DONE WITH ALL OF THE ABOVE
4065 :* OFFSETS APPLIED, AND A SEPARATE SCORE FOR EACH DRIVE IS KEPT,
4066 :* WHILE THE CURRENT DRIVE IS PERFORMING THE OVERWRITES.
4067 :* FOR EACH TRACK (0,1,2), SCORES ARE AVERAGED FOR EACH OFFSET
4068 :* DIRECTION, OVER ALL CYLINDERS TESTED.
4069 :*
4070 :*
    
```

```

*****
4071 015154 000004 TST6: SCOPE
4072 015156 012737 000006 001334 MOV #6,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4073 015164 004737 022020 JSR PC,SETUP ;SET UP FOR LOOP ON ERROR
4074 015170 004737 021140 JSR PC,INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
4075 015174 004737 020744 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
4076 015200 112737 000001 003127 MOVB #1,TSTING ;ALLOW TTY ESCAPE
4077 015206 105037 003147 CLRB DIF100 ;CLEAR "CYL DIF = 100/200" FLAG
4078
4079 015212 112765 000117 000001 ;OVERWRITE THE APPROPRIATE CYLS FOR THIS DRIVE
4080 015220 105065 000002 MOVB #SEEK,P.CMND(R5) ;SET SEEK COMMAND
4081 015224 004737 030124 CLRB P.CYLN(R5) ;SET CYL = 0
4082 015230 012703 000007 JSR PC,DRVCAL ;SEEK TO CYL 0
4083 015234 112765 000123 000001 MOV #7,R3 ;SET COUNTER = 7
4084 015242 113700 003154 MOVB #WRDATA,P.CMND(R5) ;SET WRITE DATA COMMAND
4085 015246 010065 000002 MOVB LOGDRV,RO ;GET LOGICAL DRIVE NUMBER
4086 015252 006300 MOV RO,P.CYLN(R5) ;SET STARTING OVERWRITE CYL FOR THIS DRIVE
4087 015254 062700 006512 ASL RO ;GET DATA PATTERN INDEX
4088 015260 011037 053330 ADD #TABLEG,RO ;GET PATTERN WORD ADRS
4089 015264 012765 053330 000010 MOV (RO),RWBUF ;PUT DATA WORD INTO BUFFER
4090 015272 052765 100000 000014 BIS #DTBAII,P.PRST(R5) ;SET BUS ADRS INCREMENT INHIBIT
4091 015300 012765 137000 000012 MOV #-16896.,P.WC(R5) ;SET 22-SECTOR WORD COUNT
4092 015306 105737 003134 TSTB FORMAT ;DETERMINE THE FORMAT
4093 015312 001403 BEQ 6$ ;BR IF 22 SECTORS
4094 015314 012765 142000 000012 MOV #-15360.,P.WC(R5) ;SET 20-SECTOR WORD COUNT
4095 015322 004737 027456 6$: JSR PC,SVPRMS ;SAVE PARAMETERS OF TRANSFER
4096 015326 004737 027546 JSR PC,TRNSFR ;WRITE THE DATA
4097 015332 063765 024370 000002 ADD HOLD3,P.CYLN(R5) ;INCR THE CYL NO.
4098 015340 005303 DEC R3 ;SEE IF DONE WITH ALL CYLS YET
4099 015342 001367 BNE 6$ ;BR IF NOT YET
4100 ;READ OVERWRITE CYLS WITH RANGE OF OFFSETS
4101 015344 112737 000001 003143 MOVB #1,TSTTYP ;SET INDICATOR FOR OVRWRT TEST
4102 015352 012700 006612 MOV #NOSCRO,RO ;GET STARTING ADRS OF OVRWRT SCORES
4103 015356 005020 10$: CLR (RO)+ ;INIT SCORE TO 0
4104 015360 020027 007112 CMP RO,#NOSCRO+192. ;SEE IF ALL SCORES CLEARED YET
4105 015364 103774 BLO 10$ ;BR IF NOT DONE YET
4106 015366 012703 000007 MOV #7,R3 ;SET CYL CNTR = 7
4107 015372 113700 003154 MOVB LOGDRV,RO ;GET LOGICAL DRIVE NO.
4108 015376 010065 000002 MOV RO,P.CYLN(R5) ;GET START OVRWRT CYL FOR THIS DRIVE
4109 015402 004737 022620 11$: JSR PC,INTBLD ;INIT TABLED
4110 015406 004737 017250 12$: JSR PC,REDOFS ;DO READS WITH OFFSETS ON THIS CYL
4111 015412 004737 017614 JSR PC,ADSCOR ;ADD SCRATCH SCORES TO SUM
    
```

```

4112 015416 063765 024370 000002 ADD HOLD3,P.CYLN(R5) ;INCR CYL NO.
4113 015424 112737 000001 003147 MOVB #1,DIF100 ;SET "CYL DIF = 100/200" FLAG
4114 015432 005303 DEC R3 ;DECR CYL CNTR
4115 015434 004737 023000 JSR PC,ROTBLD ;ROTATE TABLED FOR NEXT CURRENT ZONE
4116 015440 020327 000001 CMP R3,#1 ;SEE IF ON LAST ZONE YET
4117 015444 003360 BGT 12$ ;BR IF NOT YET
4118 015446 001755 BEQ 11$ ;BR IF ON LAST CYL
4119 015450 004737 017774 JSR PC,AVSCOR ;COMPUTE AVERAGE OVERWRITE SCORES
4120 015454 105037 003143 CLRB TSITYP ;CLEAR OVRWRT TEST INDICATOR
    
```

4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142

 :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 FOR THIS DRIVE ARE WRITTEN WITH THE DATA PATTERN SHOWN IN TABLE, 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 DHECK 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.

```

4143 015460 000004
4144 015462 012737 000007 001334
4145 015470 004737 022020
4146 015474 004737 021140
4147 015500 004737 020744
4148 015504 005037 007412
4149 015510 005037 007414
4150 015514 005037 007416
4151 015520 005037 007420
4152 015524 005037 007422
4153 015530 005037 007424
4154 015534 105037 003147
4155 015540 012703 000006
4156 015544 113700 003154
4157 015550 062700 000060
4158 015554 012765 053330 000010
4159 015562 004737 024460
4160 015566 012765 177400 000012
4161 015574 112765 000004 000004 3$:
4162 015602 105065 000005 4$:
4163 015606 105037 007432
4164 015612 105037 007452
4165 015616 105037 007472
4166 015622 105037 007512
4167 015626 105037 007532
    
```

```

*****
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 NSSCR0 ;:INIT SELF-TEST SCORES TO 0
CLRB NSSCR1
CLRB NSSCR2
CLRB PSSCR0
CLRB PSSCR1
CLRB PSSCR2
CLRB DIF100 ;:CLEAR "CYL DIF = 100/200" FLAG
MOVB #6,R3 ;:SET CYL COUNTER = 6
MOVB LOGDRV,RO ;:GET LOGICAL DRIVE NO.
ADD #60,RO ;:COMPUTE STARTING CYL NO.
MOV #RWBUF,P.BALO(R5) ;:SET BUS ADRS
JSR PC,LODSEC ;:LOAD R/W BUF WITH DATA PATTERN
MOV #-256.,P.WC(R5) ;:SET WORD COUNT FOR 1 SECTOR
MOVB #4,P.SECT(R5) ;:SET SECTOR = 4
CLRB P.TRCK(R5) ;:SET TRACK = 0
CLRB NSCOR0 ;:INIT SCRATCH SCORES TO 0
CLRB NSCOR1
CLRB NSCOR2
CLRB PSCOR0
CLRB PSCOR1
    
```

```

4168 015632 105037 007552          CLRB  PSCOR2
4169 015636 105037 003143          CLRB  TSTTYP          ;CLEAR OFFSET TEST INDICATOR
4170 015642 005065 000002          CLR   P.CYLN(R5)      ;SET CYL = 0
4171 015646 112765 000117 000001  MOVB  #SEEK,P.CMND(R5) ;SET SEEK COMMAND
4172 015654 004737 030124          JSR   PC,DRVCAL      ;PERFORM SEEK TO 0
4173 015660 010065 000002          MOV   R0,P.CYLN(R5)  ;SET CURRENT CYLINDER NUMBER
4174 015664 112765 000123 000001  MOVB  #WRDATA,P.CMND(R5);SET WRITE DATA COMMAND
4175 015672 004737 030124          JSR   PC,DRVCAL      ;WRITE A SECTOR
4176 015676 112737 000003 003143  MOVB  #3,TSTTYP      ;SET INDICATOR FOR SELF-TEST
4177 015704 004737 017250          JSR   PC,REDOFS     ;READ SECTOR WITH RANGE OF OFFSETS
4178 015710 105265 000005          INCB  P.TRCK(R5)     ;INCREMENT TRACK
4179 015714 126527 000005 000002  CMPB  P.TRCK(R5),#2  ;SEE IF ALL TRACKS DONE ON THIS CYL
4180 015722 101745          BLOS  6$            ;BR IF NOT YET
4181 015724 004737 017614          JSR   PC,ADSCOR     ;ADD SCRATCH SCORES TO SUMS
4182 015730 062765 000005 000004  ADD   #5,P.SECT(R5)  ;INCREMENT SECTOR BY 5
4183 015736 126527 000004 000023  CMPB  P.SECT(R5),#23 ;SEE IF ALL SECTORS DONE ON THIS TRACK
4184 015744 101716          BLOS  4$            ;BR IF NOT YET
4185 015746 063700 024370          ADD   HOLD3,R0      ;INCR CYL NO. BY 100/200
4186 015752 005303          DEC   R3            ;DECR CYL CNTR
4187 015754 001307          BNE   3$            ;BR IF NOT DONE WITH ALL CYLS YET
4188 015756 004737 017774          JSR   PC,AVSCOR     ;COMPUTE AVG SELF-TEST SCORES
4189 015762 105037 003143          CLRB  TSTTYP        ;CLEAR OVRWRT TEST INDICATOR
    
```

4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219

```

*****
: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 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 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.
    
```

```

4220 015766 000004          TST10: SCOPE
4221 015770 012737 000010 001334  MOV   #10,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
4222 015776 004737 022020          JSR   PC,SETUP      ;SET UP FOR LOOP ON ERROR
4223 016002 004737 021140          JSR   PC,INITSS     ;INITIALIZE DRIVER PARAMS AND SUB-SYS
    
```

4224	016006	004737	020744		JSR	PC,PREPKB	:PREPARE FOR POSSIBLE KBD INPUT
4225	016012	112737	000002	003143	MOVB	#2,TSTTYP	:SET INDIC FOR READ TEST
4226	016020	105037	003147		CLRB	DIF100	:CLEAR 'CYL DIF = 100/200' FLAG
4227	016024	012700	007112		MOV	#NCSCRO,R0	:GET ADRS OF COMPAT READ SCORES
4228	016030	005020		6\$:	CLR	(R0)+	:INIT A SCORE TO 0
4229	016032	020027	007412		CMP	R0,#NCSCRO+192.	:SEE IF ALL SCORES CLEARED YET
4230	016036	103774			BLO	6\$:BR IF NOT YET
4231	016040	012703	000006		MOV	#6,R3	:SET CYL BLK CNTR = 6
4232	016044	004737	022620		JSR	PC,INTBLD	:INIT TABLE D
4233	016050	013701	024372		MOV	HOLD4,R1	:SET START CYL = 60/160
4234	016054	012765	053330	000010	MOV	#RWBUF,P.BALO(R5)	:SET BUS ADRS
4235	016062	052765	100000	000014	BIS	#DTBAII,P.PRST(R5)	:SET BUS ADRS INCR INHIBIT
4236	016070	012765	137000	000012	MOV	#-16896.,P.WC(R5)	:SET 22-SECTOR WORD COUNT
4237	016076	105737	003134		TSTB	FORMAT	:TEST THE FORMAT
4238	016102	001403			BEQ	10\$:BR IF 22 SECTORS
4239	016104	012765	142000	000012	MOV	#-15360.,P.WC(R5)	:SET 20-SECTOR WORD COUNT
4240	016112	012700	000020	10\$:	MOV	#16.,R0	:INIT CYL CNTR = 16.
4241	016116	010165	000002		MOV	R1,P.CYLN(R5)	:SET CYL NO.
4242	016122	004737	017250	12\$:	JSR	PC,REDOFS	:DO READS WITH OFSTS ON THIS CYL
4243	016126	004737	017614		JSR	PC,ADSCOR	:ADD SCRATCH SCORES TO SUM
4244	016132	005265	000002		INC	P.CYLN(R5)	:INCR CYL NO.
4245	016136	005300			DEC	R0	:SEE IF DONE WITH THIS CYL BLK
4246	016140	001370			BNE	12\$:BR IF NOT YET
4247	016142	004737	023000		JSR	PC,ROTBLD	:ROTATE TABLED FOR NEXT CURRENT ZONE
4248	016146	063701	024370		ADD	HOLD3,R1	:INCR CYL BY 100/200
4249	016152	112737	000001	003147	MOVB	#1,DIF100	:SET 'CYL DIF = 100' FLAG
4250	016160	005303			DEC	R3	:SEE IF DONE WITH ALL CYL BLKS
4251	016162	001353			BNE	10\$:BR IF NOT YET
4252	016164	004737	017774		JSR	PC,AVSCOR	:COMPUTE AVERAGES OF READ SCORES
4253	016170	105037	003143		CLRB	TSTTYP	:CLEAR OFST TEST INDICATOR
4254					;ADJUST	COMPAT READ SCORES (- OFSTS)	BY SELF-TEST SCORES
4255	016174	012701	007112		MOV	#NCSCRO,R1	:SET COMPAT SCORE POINTER
4256	016200	012703	007412		MOV	#NSSCRO,R3	:SET SELF-TEST SCORE POINTER
4257	016204	012700	005512	14\$:	MOV	#DRVLST,R0	:SET DRIVE LIST POINTER
4258	016210	012737	000014	001304	MOV	#12.,\$TMP11	:COMPUTE DIFFERENCE OF 12 (DEC) AND
4259							SELF-TEST SCORE FOR THIS TRACK
4260	016216	162337	001304		SUB	(R3)+,\$TMP11	
4261	016222	010104			MOV	R1,R4	:GET READ SCORE POINTER IN R4
4262	016224	005720		15\$:	TST	(R0)+	:SEE IF A DRIVE LISTED HERE
4263	016226	001417			BEQ	18\$:BR IF NOT
4264	016230	013737	001304	001306	MOV	\$TMP11,\$TMP12	:PUT DIFFERENCE INTO \$TMP12
4265	016236	061437	001306		ADD	(R4),\$TMP12	:ADD DIFF TO READ SCORE
4266	016242	023727	001306	000014	CMP	\$TMP12,#12.	:SEE IF 12 EXCEEDED
4267	016250	101403			BLOS	16\$:BR IF NOT
4268	016252	012737	000014	001306	MOV	#12.,\$TMP12	:SET SCORE = 12.
4269	016260	013724	001306	16\$:	MOV	\$TMP12,(R4)+	:PUT BACK ADJUSTED SCORE
4270	016264	000757			BR	15\$:BR TO CONTINUE
4271	016266	062701	000040	18\$:	ADD	#32.,R1	:POINT TO SCORES FOR NEXT TRACK
4272	016272	020127	007212		CMP	R1,#NCSCRO+64.	:SEE IF SCORE TABLE EXCEEDED
4273	016276	101742			BLOS	14\$:BR IF NOT YET
4274					;ADJUST	COMPAT READ SCORES (+ OFSTS)	BY SELF-TEST SCORES
4275	016300	012701	007252		MOV	#PCSCRO,R1	:SET COMPAT SCORE POINTER
4276	016304	012703	007420		MOV	#PSSCRO,R3	:SET SELF-TEST SCORE POINTER
4277	016310	012700	005512	24\$:	MOV	#DRVLST,R0	:SET DRIVE LIST POINTER
4278	016314	012737	000014	001304	MOV	#12.,\$TMP11	:COMPUTE DIFFERENCE OF 12 (DEC) AND
4279							SELF-TEST SCORE FOR THIS TRACK

```

4280 016322 162337 001304 SUB (R3)+,$TMP11
4281 016326 010104 MOV R1,R4 ;GET READ SCORE POINTER IN R4
4282 016330 005720 25$: TST (R0)+ ;SEE IF A DRIVE LISTED HERE
4283 016332 001417 BEQ 28$ ;BR IF NOT
4284 016334 013737 001304 001306 MOV $TMP11,$TMP12 ;PUT DIFFERENCE INTO $TMP12
4285 016342 061437 001306 ADD (R4),$TMP12 ;ADD DIFF TO READ SCORE
4286 016346 023727 001306 000014 CMP $TMP12,#12. ;SEE IF 12 EXCEEDED
4287 016354 101403 BLOS 26$ ;BR IF NOT
4288 016356 012737 000014 001306 MOV #12,$TMP12 ;SET SCORE = 12.
4289 016364 013724 001306 26$: MOV $TMP12,(R4)+ ;PUT BACK ADJUSTED SCORE
4290 016370 000757 BR 25$ ;BR TO CONTINUE
4291 016372 062701 000040 28$: ADD #32.,R1 ;POINT TO SCORES FOR NEXT TRACK
4292 016376 020127 007352 CMP R1,#PCSCRO+64. ;SEE IF SCORE TABLE EXCEEDED
4293 016402 101742 BLOS 24$ ;BR IF NOT YET
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
  
```

```

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

```

4317 016404 000004 TST11: SCOPE
4318 016406 012737 000011 001334 MOV #11,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4319 016414 004737 022020 JSR PC,SETUP ;SET UP FOR LOOP ON ERROR
4320 016420 004737 021140 JSR PC,INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
4321 016424 004737 020744 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
4322 016430 112737 000001 003127 MOV #1,TSTING ;ALLOW TTY ESCAPE
4323 :CONVERT AND TYPE TEST SCORES
4324 016436 013700 005554 MOV DRVPTR,R0 ;GET DRIVE LIST POINTER
4325 016442 112037 010713 MOV (R0)+,RESLTS+20. ;GET NO. OF THIS DRIVE
4326 016446 111037 010712 MOV (R0),RESLTS+19.
4327 016452 104401 010667 TYPE ,RESLTS ;TYPE TEST RESULT HEADINGS
4328 016456 012737 006612 001306 MOV #NOSCR0,$TMP12 ;GET POINTER TO OVRWRT SCORES
4329 016464 012700 000060 MOV #0,R0 ;INIT TRACK NO. TO 0
4330 016470 012701 005512 5$: MOV #DRVLST,R1 ;GET DRIVE LIST POINTER
4331 016474 110037 011175 MOV #R0,TRKBUF+1 ;SET TRACK NO. FOR PRINTOUT
4332 016500 013704 001306 MOV $TMP12,R4 ;GET SCORE POINTER
4333 016504 004737 025542 6$: JSR PC,CTLOUT ;CHECK FOR TTY ESCAPE RQST
4334 016510 104401 011174 TYPE ,TRKBUF ;TYPE TRACK (HEAD) NUMBER
4335 016514 012103 MOV (R1)+,R3 ;GET A DRIVE NAME
  
```

```

4336 016516 020377 167032      CMP      R3,@DRVPTR      ;SEE IF IT IS CURRENT DRIVE
4337 016522 001003      BNE      8$              ;BR IF NOT
4338 016524 104401 011205      TYPE     ,SELF          ;TYPE "SELF"
4339 016530 000407      BR       10$            ;CONTINUE
4340 016532 110337 011202      8$:     MOVB     R3,DRVBUF+2 ;GET DRIVE NO.
4341 016536 000303      SWAB     R3
4342 016540 110337 011201      MOVB     R3,DRVBUF+1    ;GET SUBSYS NAME
4343 016544 104401 011200      TYPE     ,DRVBUF        ;TYPE NO. OF DRIVE READ
4344 016550 005037 005212      10$:    CLR      BUFFO      ;CLEAR TTY CHAR BUF
4345 016554 005037 005214      CLR      BUFFO+2
4346 016560 010437 001310      MOV      R4,$TMP13      ;GET SCORE POINTER
4347 016564 012437 005212      MOV      (R4)+,BUFFO    ;PUT AN OVRWRT SCORE INTO BUF
4348 016570 012737 000004 001304      MOV      #4,$TMP11      ;SET COUNTER = 4
4349 016576 023727 005212 000012 11$:    CMP      BUFFO,#10.     ;SEE IF SCORE < 10.
4350 016604 103002      BHIS     12$            ;BR IF NOT
4351 016606 104401 012120      TYPE     ,SPACE1        ;TYPE 1 SPACE
4352 016612 023727 005212 000007 12$:    CMP      BUFFO,#7       ;SEE IF SCORE > 7
4353 016620 101003      BHI      14$            ;BR IF YES
4354 016622 104401 012140      TYPE     ,PROMPT        ;TYPE "*"
4355 016626 000402      BR       16$            ;CONTINUE
4356 016630 104401 012117      14$:    TYPE     ,SPACE2        ;TYPE 2 SPACES
4357 016634 012746 005212 16$:    MOV      #BUFFO,-(SP)   ;GET BUF ADRS ON STACK
4358 016640 004737 043162      JSR      PC,@#$DB2D     ;CONVERT SCORE TO DEC.
4359 016644 004737 043356      JSR      PC,@#$SUPRS    ;TYPE IT
4360 016650 104401 011213      TYPE     ,TAB           ;TYPE A TAB
4361 016654 062737 000140 001310      ADD      #96,$TMP13
4362 016662 017737 162422 005212      MOV      @TMP13,BUFFO   ;PUT NEXT SCORE INTO BUFFER
4363 016670 005337 001304      DEC      $TMP11         ;DECREMENT SCORE COUNTER
4364 016674 001340      BNE      11$            ;BR IF READ SCORE SHOULD BE TYPED
4365 016676 104401 001325      TYPE     ,$CRLF         ;TYPE <CR>, <LR>
4366 016702 005711      TST      (R1)           ;SEE IF ALL SCORES TYPED FOR THIS TRACK
4367 016704 001277      BNE      6$             ;BR IF NOT
4368 016706 104401 001325      TYPE     ,$CRLF         ;TYPE <CR>, <LF>
4369 016712 005200      INC      R0             ;INCR TRACK NO.
4370 016714 062737 000040 001306      ADD      #32,$TMP12     ;INCREMENT SCORE POINTER FOR NEXT TRACK
4371 016722 020027 000062      CMP      R0,#12         ;SEE IF ALL TRACKS TYPED YET
4372 016726 101660      BLOS     5$             ;BR IF NOT YET
4373      ;COMPUTE, CONVERT, AND TYPE TEST AVERAGES (OVER ALL SURFACES)
4374 016730 004737 020232      JSR      PC,TRKAVG      ;GET AVGS OVER ALL TRKS
4375 016734 013700 005554      MOV      DRVPTR,R0      ;PUT DRIVE NAME INTO BUF
4376 016740 112037 011262      MOVB     (R0)+,BADDRV+7
4377 016744 111037 011261      MOVB     (R0),BADDRV+6
4378 016750 005037 001304      CLR      $TMP11         ;CLEAR SCORE INDICATOR
4379 016754 013737 007426 005212      MOV      OVRAVG,BUFFO   ;GET OVRWRT AVG SCORE
4380 016762 012737 011102 016776      MOV      #OAVERG,22$
4381 016770 104401 011253 20$:    TYPE     ,BADDRV        ;TYPE "DRIVE XX"
4382 016774 104401      TYPE
4383 016776 000000 22$:    .WORD     0            ;TYPE WHICH KIND OF AVERAGE
4384 017000 012746 005212      MOV      #BUFFO,-(SP)   ;PUT POINTER TO AVG ON STACK
4385 017004 004737 043162      JSR      PC,@#$DB2D     ;CONVERT TO DECIMAL
4386 017010 004737 043356      JSR      PC,@#$SUPRS    ;TYPE IT
4387 017014 104401 001325      TYPE     ,$CRLF         ;TYPE <CR>, <LF>
4388 017020 013737 007430 005212      MOV      COMAVG,BUFFO   ;GET READ AVG SCORE
4389 017026 012737 011121 016776      MOV      #RAVERG,22$
4390 017034 005137 001304      COM      $TMP11         ;COMPLEMENT SCORE INDIC
4391 017040 001353      BNE      20$            ;BR TO TYPE READ SCORE
    
```

```

4392 017042 104401 001325          TYPE      ,SCLF          ;TYPE <CR>,<LF>
4393          ;DISMOUNT PACK ON THIS DRIVE
4394 017046 105037 003127          CLR      TSTING          ;DON'T ALLOW TTY ESCAPE
4395 017052 004737 022450          JSR      PC,DMCART        ;DIRECT OPERATOR TO DISMOUNT PACK
4396 017056 062737 000002 005554      ADD      #2,DRVPT        ;INCREMENT DRIVE LIST POINTER
4397 017064 013700 005554      MOV      DRVPT,RO
4398 017070 005710          TST      (RO)            ;SEE IF ALL DONE WITH PASS 2 YET
4399 017072 001421          BEQ      28$             ;BR IF DONE WITH PASS 2
4400 017074 126037 000001 005510      CMPB    1(RO),SUBSYS     ;SEE IF DONE WITH THIS SUBSYS YET
4401 017102 001005          BNE      24$             ;BR IF DONE WITH THIS SUBSYS
4402 017104 112737 000004 001102      MOVB    #4,$STSTNM       ;SET TEST NO. = 4
4403 017112 000137 015116          JMP      TST5            ;DO PASS 2 ON NEXT DRIVE OF SUBSYS
4404 017116 116037 000001 010544 24$:  MOVB    1(RO),STR220+39. ;PUT SUBSYS NAME INTO MSG
4405 017124 104401 010475          TYPE    ,STR220          ;TYPE RESTRT MSG FOR PASS 2
4406 017130 000000          26$:  HALT
4407 017132 000137 000220          JMP      220             ; CAN RESTART AT PROPER ADRS
4408 017136 104401 011140          28$:  TYPE    ,ENDTST      ;TYPE '** END OF TESTING **'
4409 017142 012737 005512 005554      MOV      #DRVLST,DRVPT   ;RE-INIT DRIVE LIST POINTER
4410 017150 105737 003126          TSTB    MDFLAG          ;SEE IF 200 START
4411 017154 001002          BNE      30$             ;BR IF NOT
4412 017156 000137 000200          JMP      200             ;RESTART AT ADRS 200
4413 017162 000000          30$:  HALT                ;HALT THE PROCESSOR
4414 017164 000137 000204          JMP      204             ;RESTART AT ADRS 204
4415
4416
4417
4418          .SBTTL  END OF PASS ROUTINE
4419
4420          ;*****
4421          ;*INCREMENT THE PASS NUMBER ($PASS)
4422          ;*IF THERES A MONITOR GO TO IT
4423          ;*IF THERE ISN'T JUMP TO 200
4424
4425          $EOP:
4426          SCOPE
4427          CLR      $STSTNM          ;;ZERO THE TEST NUMBER
4428          INC      $PASS            ;;INCREMENT THE PASS NUMBER
4429          BIC      #100000,$PASS    ;;DON'T ALLOW A NEG. NUMBER
4430          DEC      (PC)+            ;;LOOP?
4431          $EOPCT: .WORD 1
4432          BGT      $DOAGN          ;;YES
4433          MOV      (PC)+,@(PC)+     ;;RESTORE COUNTER
4434          $ENDCT: .WORD 1
4435          $EOPCT
4436          $GET42: MOV      @#42,RO    ;;GET MONITOR ADDRESS
4437          BEQ      $DOAGN          ;;BRANCH IF NO MONITOR
4438          RESET
4439          $ENDAD: JSR      PC,(RO)    ;;GO TO MONITOR
4440          NOP
4441          NOP
4442          NOP
4443          $DOAGN:
4444          JMP      @#200            ;;RETURN
4445
4446
4447
    
```



```

4448
4449
4450
4451
4452
4453
4454
4455
4456
4457 017250 104407
4458
4459 017252 122737 000003 003143
4460 017260 001406
4461 017262 012700 007432
4462 017266 105020
4463 017270 020027 007572
4464 017274 103774
4465
4466 017276 112765 000117 000001
4467 017304 105737 003147
4468 017310 001412
4469 017312 163765 024374 000002
4470 017320 004737 030124
4471 017324 063765 024374 000002
4472 017332 105037 003147
4473 017336 004737 030124
4474 017342 105037 003156
4475 017346 012703 000014
4476 017352 012700 000204
4477 017356 105037 003157
4478 017362 122737 000003 003143
4479 017370 001020
4480 017372 116501 000005
4481 017376 006301
4482 017400 006301
4483 017402 006301
4484 017404 006301
4485 017406 062701 007432
4486 017412 105737 003156
4487 017416 001402
4488 017420 062701 000060
4489 017424 012704 000001
4490 017430 000411
4491 017432 012701 007432
4492 017436 012704 000060
4493 017442 105737 003156
4494 017446 001402
4495 017450 012701 007512
4496 017454 122137 003157
4497 017460 001002
4498 017462 105261 177777
4499 017466 005304
4500 017470 001371
4501 017472 110065 000006
4502 017476 112765 000115 000001
4503 017504 004737 030124

```

```

*****
*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,
*UNTIL THERE IS NO FAILURE. EACH TIME A READ ERROR OCCURS,
*THE APPROPRIATE SCORE FOR THE DRIVE INVOLVED IS INCREMENTED NO FURTHER
*(IN THE SCRATCH SCORE TABLE). EACH READ IS DONE TO A PACK
*AREA, WHOSE ADDRESS IS IN THE PARAMETER BLOCK ON ENTRY.
*****
REDOFS: SAVREG ;SAVE R0-R5
;INITIALIZE SCRATCH SCORES
CMPB #3,TSTTYP ;SEE IF DRIVE SELF TEST
BEQ 5$ ;BR IF YES
MOV #NSCORO,R0 ;GET ADRS OF SCRATCH SCORE TABLE
4$: CLRB (R0)+ ;INIT A SCORE TO 0
CMP R0,#NSCORO+96. ;SEE IF DONE YET
BLO 4$ ;BR IF NOT YET
;PERFORM READS WITH OFFSET
5$: MOVB #SEEK,P.CMND(R5) ;SET SEEK COMMAND
TSTB DIF100 ;SEE IF CYL DIF = 100/200 (OCT)
BEQ 9$ ;BR IF NOT
SUB HOLDS,P.CYLN(R5) ;SEEK HALF THE CYLS
JSR PC,DRVCAL
ADD HOLDS,P.CYLN(R5) ;SEEK THE REST OF THE WAY
CLRB DIF100 ;CLEAR "CYL DIF = 100/200 FLAG"
9$: JSR PC,DRVCAL ;PERFORM SEEK
CLRB OFSDIR ;CLEAR OFFSET DIRECTION FLAG
MOV #12.,R3 ;SET OFFSET COUNTER FOR NEG OFSTS
MOV #204,R0 ;INIT NEG OFST VALUE TO 100 UIN
12$: CLRB OFSCNT ;INIT OFFSET NUMBER
6$: CMPB #3,TSTTYP ;SEE IF SELF-TEST
BNE 15$ ;BR IF NOT
MOVB P.TRCK(R5),R1 ;GET TRACK NUMBER
ASL R1 ;MULT BY 16.
ASL R1
ASL R1
ASL R1
ADD #NSCORO,R1 ;GET POINTER TO SCORES
TSTB OFSDIR ;SEE WHICH OFST DIRECTION
BEQ 17$ ;BR IF NEGATIVE
ADD #48.,R1 ;POINT TO POSITIVE SCORES
17$: MOV #1,R4 ;SET COUNTER TO 1
BR 14$ ;CONTINUE
15$: MOV #NSCORO,R1 ;GET POINTER TO SCORES
MOV #48.,R4 ;SET COUNTER TO 48.
TSTB OFSDIR ;SEE WHICH OFST DIRECTION
BEQ 14$ ;BR IF NEGATIVE
MOV #PSCORO,R1 ;GET POINTER FOR POS SCORES
14$: CMPB (R1)+,OFSCNT ;SEE IF THIS IS A PERFECT SCORE YET
BNE 18$ ;BR IF NOT
INCB -1(R1) ;INCREMENT THIS SCORE
18$: DEC R4 ;DECREMENT COUNTER
BNE 14$ ;BR IF NOT DONE YET
16$: MOVB R0,P.OFST(R5) ;GET AN OFFSET VALUE
MOVB #OFFSET,P.CMND(R5) ;SET OFFSET CMND
JSR PC,DRVCAL ;PERFORM OFFSET

```

```

4504 017510 105237 003157      INCB      OFSCNT      : INCR OFFSET NUMBER
4505 017514 112765 000121 000001  MOVB      #RDATA,P.CMND(R5) ; SET READ COMMAND
4506 017522 004737 027456      JSR      PC,SVPRMS      ; SAVE PARAMETERS OF TRANSFER
4507 017526 004737 027546      JSR      PC,TRNSFR      ; DO XFER, HANDLE DCK ERRORS
4508 017532 112765 000177 000001  MOVB      #SUBCLR,P.CMND(R5) ; SET SUBSYS CLEAR COMMAND
4509 017540 004737 030124      JSR      PC,DRVCAL      ; DO A SUBSYSTEM CLEAR
4510 017544 105065 000006      CLRB      P.OFST(R5)      ; CLEAR OFFSET VALUE
4511 017550 112765 000115 000001  MOVB      #OFFSET,P.CMND(R5) ; SET OFFSET COMMAND
4512 017556 004737 030124      JSR      PC,DRVCAL      ; PERFORM OFFSET TO 0
4513 017562 062700 000004      ADD      #4,R0          ; INCR OFST VALUE BY 100 UIN
4514 017566 005303      DEC      R3            ; DECR COUNTER
4515 017570 001274      BNE      6$           ; BR IF NOT DONE IN THIS DIRECTION YET
4516 017572 012703 000014      MOV      #12,R3       ; SET COUNTER FOR POS SFSTS
4517 017576 012700 000004      MOV      #4,R0        ; INIT POS OFST TO 100 UIN
4518 017602 105137 003156      COMB      OFSDIR      ; COMPLEMENT OFFSET DIRECTION FLAG
4519 017606 001263      BNE      12$         ; BR IF POS OFSTS NOT DONE YET
4520 017610 104410      RESREG      ; RESTORE R0-R5
4521 017612 000207      RTS      PC          ; RETURN
4522
4523
4524
4525
4526
4527
4528
4529

```

```

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

```

```

4530 017614 104407      ADSCOR: SAVREG      : SAVE R0-R5
4531 017616 122737 000003 003143  CMPB      #3,TSTTYP      ; SEE IF SELF-TEST
4532 017624 001031      BNE      2$           ; BR IF NOT
4533 017626 113700 007432      MOVB      NSCOR0,R0      ; ADD SELF-TEST SCORES
4534 017632 060037 007412      ADD      R0,NSSCRO
4535 017636 113700 007452      MOVB      NSCOR1,R0
4536 017642 060037 007414      ADD      R0,NSSCR1
4537 017646 113700 007472      MOVB      NSCOR2,R0
4538 017652 060037 007416      ADD      R0,NSSCR2
4539 017656 113700 007512      MOVB      PSCOR0,R0
4540 017662 060037 007420      ADD      R0,PSSCRO
4541 017666 113700 007532      MOVB      PSCOR1,R0
4542 017672 060037 007422      ADD      R0,PSSCR1
4543 017676 113700 007552      MOVB      PSCOR2,R0
4544 017702 060037 007424      ADD      R0,PSSCR2
4545 017706 000430      BR      8$           ; EXIT
4546 017710 012700 006612      2$: MOV      #NOSCRO,R0      ; GET POINTER TO (-) OVRWRT SCORES
4547 017714 012705 006752      MOV      #POSCRO,R5     ; GET POINTER TO (+) OVRWRT SCORES
4548 017720 122737 000001 003143  CMPB      #1,TSTTYP      ; SEE IF READING OVRWRT DATA
4549 017726 001404      BEQ      4$           ; BR IF YES
4550 017730 012700 007112      MOV      #NCSCRO,R0     ; GET POINTER TO (-) COMPAT DATA SCORES
4551 017734 012705 007252      MOV      #PCSCRO,R5     ; GET POINTER TO (+) COMPAT DATA SCORES
4552 017740 012704 000060      4$: MOV      #48,R4        ; INIT COUNTER TO 48.
4553 017744 012701 007432      MOV      #NSCOR0,R1     ; GET POINTER TO NEG SCRATCH SCORES
4554 017750 012703 007512      MOV      #PSCOR0,R3     ; GET POINTER TO POS SCRATCH SCORES
4555 017754 112102      6$: MOVB      (R1)+,R2      ; GET A SCRATCH NEG SCORE
4556 017756 060220      ADD      R2,(R0)+      ; ADD IT TO (-) SCORE SUM
4557 017760 112302      MOVB      (R3)+,R2      ; GET PNTR TO POS SCRATCH SCORES
4558 017762 060225      ADD      R2,(R5)+      ; ADD IT TO (+) SCORE SUM
4559 017764 005304      DEC      R4            ; DECREMENT COUNTER

```

```

4560 017766 001372
4561 017770 104410
4562 017772 000207
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572 017774 104407
4573 017776 122737 000003 003143
4574 020004 001014
4575 020006 012737 007412 001304
4576 020014 012737 007420 001310
4577 020022 012705 000030
4578 020026 012737 000003 001306
4579 020034 000427
4580 020036 012737 006612 001304
4581 020044 012737 006752 001310
4582 020052 012705 000007
4583 020056 122737 000001 003143
4584 020064 001410
4585 020066 012737 007112 001304
4586 020074 012737 007252 001310
4587 020102 012705 000140
4588 020106 012737 000060 001306
4589 020114 105037 003156
4590 020120 005000
4591 020122 005001
4592 020124 005002
4593 020126 005004
4594 020130 017703 161150
4595 020134 105737 003156
4596 020140 001402
4597 020142 017703 161142
4598 020146 004737 042664
4599 020152 010500
4600 020154 006200
4601 020156 020100
4602 020160 103401
4603 020162 005203
4604 020164 105137 003156
4605 020170 001406
4606 020172 010377 161106
4607 020176 062737 000002 001304
4608 020204 000745
4609 020206 010377 161076
4610 020212 062737 000002 001310
4611 020220 005337 001306
4612 020224 001333
4613 020226 104410
4614 020230 000207
4615

      BNE      6$      ;BR IF NOT DONE YET
      RESREG   ;RESTORE R0-R5
      RTS      PC      ;RETURN

      8$:
      BNE      6$      ;BR IF NOT DONE YET
      RESREG   ;RESTORE R0-R5
      RTS      PC      ;RETURN

      ;*****
      ;*AVSCOR - COMPUTE THE + AND - AVERAGE SCORES FOR THE OVRWRT, COMPATIBILITY
      ;*DATA TEST, OR SELF-TEST, AS DETERMINED BY 'TSTTYP'. 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,TSTTYP ;SEE IF SELF-TEST
        BNE     2$      ;BR IF NOT
        MOV     #NSSCRO,$TMP11 ;GET ADRS OF (-) SELF-TEST SCORES
        MOV     #PSSCRO,$TMP13 ;GET ADRS OF (+) SELF-TEST SCORES
        MOV     #24.,R5 ;SET DIVISOR = 24.
        MOV     #3,$TMP12 ;SET COUNTER = 3
        BR      12$     ;CONTINUE
      2$: MOV     #NOSCRO,$TMP11 ;GET POINTER TO (-) OVRWRT SCORES
        MOV     #POSCRO,$TMP13 ;GET POINTER TO (+) OVRWRT SCORES
        MOV     #7.,R5 ;SET DIVISOR = 7.
        CMPB    #1,TSTTYP ;SEE IF READING OVRWRT DATA
        BEQ     4$      ;BR IF YES
        MOV     #NCSCRO,$TMP11 ;GET POINTER TO (-) COMPAT DATA SCORES
        MOV     #PCSCRO,$TMP13 ;GET POINTER TO (+) COMPAT DATA SCORES
        MOV     #96.,R5 ;SET DIVISOR = 96.
        MOV     #48.,$TMP12 ;SET COUNTER = 48.
      4$: MOV     OFSDIR ;INIT FLAG FOR (-) DIRECTION
      12$: CLR    R0 ;CLEAR HI BITS OF DIVIDEND AND DIVISOR
          CLR    R1
          CLR    R2
          CLR    R4
          MOV     @ $TMP11,R3 ;SET DIVIDEND = (-) SUM
          TSTB   OFSDIR ;SEE WHICH OFST DIRECTION
          BEQ    10$
          MOV     @ $TMP13,R3 ;SET DIVIDEND = (+) SUM
          JSR    PC,M.DPID ;DIVIDE TO GET AVERAGES
          MOV     R5,R0 ;GET DIVISOR
          ASR    R0 ;DIVIDE IT BY 2
          CMP    R1,R0 ;SEE IF REMNDR > HALF DIVISOR
          BLO   8$      ;BR IF NOT
          INC    R3 ;ADD 1 TO QUOTIENT
          COMB   OFSDIR ;COMPL DIR FLAG
          BEQ   14$     ;BR FOR (+)
          MOV     R3,@ $TMP11 ;PUT (-) AVG INTO TABLE
          ADD    #2,$TMP11
          BR     6$
      8$: MOV     R3,@ $TMP13 ;PUT (+) AVG INTO TABLE
          ADD    #2,$TMP13
          DEC    $TMP12 ;DECREMENT COUNTER
          BNE   12$     ;BR IF NOT DONE YET
          RESREG ;RESTORE R0-R5
          RTS    PC      ;RETURN
  
```

```

4616
4617
4618
4619
4620
4621
4622
4623
4624
4625 020232 104407
4626
4627 020234 012700 005512
4628 020240 005001
4629 020242 005037 007426
4630 020246 005037 007430
4631 020252 012702 006612
4632 020256 012703 007112
4633 020262 005720
4634 020264 001402
4635 020266 005201
4636 020270 000774
4637 020272 010100
4638 020274 010204
4639 020276 010305
4640 020300 062437 007426
4641 020304 066437 000136 007426
4642 020312 062537 007430
4643 020316 066537 000136 007430
4644 020324 005300
4645 020326 001364
4646 020330 062702 000040
4647 020334 062703 000040
4648 020340 020227 006712
4649 020344 101752
4650
4651 020346 005004
4652 020350 005005
4653 020352 060105
4654 020354 060105
4655 020356 060105
4656 020360 006305
4657 020362 012737 007426 001304
4658 020370 005037 001306
4659 020374 005000
4660 020376 005001
4661 020400 005002
4662 020402 017703 160676
4663 020406 004737 042664
4664 020412 010500
4665 020414 006200
4666 020416 020100
4667 020420 103401
4668 020422 005203
4669 020424 010377 160654
4670 020430 012737 007430 001304
4671 020436 005137 001306

;*****
;*TRKAVG - COMPUTE AVERAGES OF TEST SCORES OVER ALL SURFACES
;*THIS SUBROUTINE COMPUTES THE AVERAGE OF THE OVERWRITE TEST SCORES
;*OVER ALL TRACKS (SURFACES) AND PLACES THE AVERAGE IN OVRAVG.
;*THEN, IT COMPUTES THE AVERAGE OF THE COMPATIBILITY DATA READ SCORES
;*OVER ALL TRACKS AND PLACES THE AVERAGE IN COMAVG.
;*****
TRKAVG: SAVREG ;SAVE R0-R5
;TAKE SUMS OF OVERWRITE AND COMPAT READ TEST SCORES OVER ALL TRACKS
MOV #DRVLST,R0 ;INIT DRIVE LIST POINTER
CLR R1 ;INIT DRIVE COUNTER
CLR OVRAVG ;INIT SUMS TO 0
CLR COMAVG
MOV #NOSCRO,R2 ;SET POINTER TO OVRWRT SCORES
MOV #NCSCRO,R3 ;SET POINTER TO READ SCORES
6$: TST (R0)+ ;COMPUTE TOTAL NUMBER OF DRIVES
BEQ 8$ ; IN R1
INC R1
BR 6$
8$: MOV R1,R0 ;PUT NO. OF DRIVES IN R0
MOV R2,R4 ;SET POINTERS TO SCORES FOR THIS TRACK
MOV R3,R5
9$: ADD (R4)+,OVRAVG ;ADD SCORES TO SUMS
ADD 94.(R4),OVRAVG
ADD (R5)+,COMAVG
ADD 94.(R5),COMAVG
DEC R0 ;SEE IF ALL SCORES ADDED FOR THIS TRACK
BNE 9$ ;BR IF NOT YET
ADD #32.,R2 ;POINT TO SCORES FOR NEXT TRACK
ADD #32.,R3
CMP R2,#NOSCRO+64. ;SEE IF ALL TRACKS DONE YET
BLOS 8$ ;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 #OVRAVG,$TMP11 ;SET POINTER TO OVRWRT SCORE SUM
CLR $TMP12 ;CLEAR SCORE INDICATOR
10$: CLR R0 ;GET DIVIDEND IN R0-R1-R2-R3
CLR R1
CLR R2
MOV @ $TMP11,R3
JSR PC,M.DPID ;DIVIDE TO GET AVERAGE
MOV R5,R0 ;GET DIVISOR
ASR R0 ;DIVIDE IT BY 2
CMP R1,R0 ;SEE IF REMNDR > HALF DIVISOR
BLO 12$ ;BR IF NOT
INC R3 ;ADD 1 TO QUOTIENT
MOV R3,@ $TMP11 ;STORE AVERAGE SCORE
MOV #COMAVG,$TMP11 ;SET POINTER FOR READ SCORE SUM
COM $TMP12 ;COMPLEMENT SCORE INDICATOR
12$:

```

```

4672 020442 001354          BNE      10$          ;BR TO COMPUTE READ SCORE AVERAGE
4673 020444 104410          RESREG
4674 020446 000207          RTS       PC          ;RESTORE R0-R5
4675                                     ;RETURN
4676
4677
4678
4679

```

```

4680 *****
4681 ;*NEDHDL - NON-EXISTENT DRIVE HANDLER
4682 ;*THIS IS THE ABNORMAL RETURN FROM THE DRIVER,
4683 ;* WHICH IS USED WHEN NED INDICATION IS EXPECTED (AND VALID).
4684 ;* (NED = NON-EXISTENT DRIVE)
4685 *****

```

```

4684 020450 032765 010000 000020 NEDHDL: BIT      #NED,P.CS2(R5) ;SEE IF NED ON DRIVE SELECT
4685 020456 001002          BNE      1$          ;BR IF NED SET
4686 020460 000137 031472          JMP      ERRHDL      ;GO HANDLE OTHER ERROR
4687 020464 010046          1$: MOV     RO,-(SP)   ;SAVE R0,R1
4688 020466 010146          MOV     R1,-(SP)
4689 020470 012700 000001          MOV     #1,R0       ;SET BIT POINTER
4690 020474 005001          CLR     R1          ;CLEAR COUNTER
4691 020476 120137 005430          2$: CMPB   R1,DRIVE  ;SEE IF R1 = CURRENT DRIVE NUMBER
4692 020502 001403          BEQ     3$          ;BR IF =
4693 020504 005201          INC     R1          ;INCREMENT COUNTER
4694 020506 006300          ASL     RO          ;SHIFT BIT POINTER
4695 020510 000772          BR      2$          ;TRY AGAIN
4696 020512 040037 005476          3$: BIC     RO,NEWON  ;CLEAR ONLINE BIT FOR THIS DRIVE
4697 020516 012601          MOV     (SP)+,R1   ;RESTORE R0,R1
4698 020520 012600          MOV     (SP)+,RO
4699 020522 000137 033742          JMP      RETNML     ;TAKE NORMAL RETURN
4700
4701
4702
4703
4704
4705

```

```

4706 *****
4707 ;*DCKHDL - DATA CHECK ERROR HANDLER
4708 ;*THIS IS THE ABNORMAL RETURN FROM THE DRIVER, WHICH IS USED
4709 ;*WHEN DCK ERROR IS EXPECTED (AND VALID).
4710 ;*IF A HEADER VRC ERROR OCCURS, IT IS HANDLED, ALSO.
4711 *****

```

```

4711 020526 005037 005424          DCKHDL: CLR     RECODE ;CLEAR ERROR FLAGS
4712 020532 032765 000400 000034          BIT      #HVRC,P.ER(R5) ;SEE IF HEADER VRC ERROR
4713 020540 001404          BEQ     4$          ;BR IF NOT
4714 020542 052737 000004 005424          BIS     #HVRCER,RECODE ;SET HVRC ERROR FLAG
4715 020550 000411          BR      8$          ;EXIT
4716 020552 032765 100000 000034          4$: BIT      #DCK,P.ER(R5) ;SEE IF DCK ERROR OCCURRED
4717 020560 001002          BNE     6$          ;BR IF YES
4718 020562 000137 031472          JMP      ERRHDL      ;GO HANDLE OTHER ERROR
4719 020566 052737 000020 005424          6$: BIS     #DCKERR,RECODE ;SET DCK ERROR FLAG
4720 020574 105237 003144          8$: INCB   DCEFLG    ;INCREMENT DCK ERROR FLAG
4721 020600 000137 033742          JMP      RETNML     ;TAKE NORMAL RETURN FROM DRIVER
4722
4723
4724
4725

```

```

4726 *****
4727 ;*SBTTL KBDHDL - TTY KEYBOARD INTERRUPT HANDLER
4728 *****

```

```

4728
4729 020604 010146          KBDHDL: MOV      R1,-(SP)          ;SAVE R1 ON STACK
4730 020606 017701 160334    MOV      @STKB,R1          ;READ A CHAR FROM KBD BUFFER
4731 020612 042701 177600    BIC      #177600,R1       ;MASK OUT UNUSED BITS
4732 020616 120127 000172    CMPB    R1,#172          ;SEE IF LOWER CASE TYPED
4733 020622 003005          BGT      20$              ;BR IF NOT
4734 020624 120127 000141    CMPB    R1,#141          ;BR IF NOT
4735 020630 002402          BLT      20$              ;BR IF NOT
4736 020632 042701 000040    BIC      #BIT5,R1         ;MAKE IT UPPER CASE
4737 020636 010137 005442    20$:    MOV      R1,INTCHR     ;SAVE INPUT CHARACTER
4738 020642 122701 000003    CMPB    #003,R1         ;SEE IF (^C) TYPED
4739 020646 001007          BNE      2$              ;BR IF NOT (^C)
4740 020650 104401 012052    TYPE    ,CNTRLC         ;ECHO (^C)
4741 020654 042777 000100 160262 1$:    BIC      #BIT6,@STKS     ;CLEAR KBD INTERRUPT ENABLE BIT
4742 020662 012601          MOV      (SP)+,R1        ;RESTORE R1
4743 020664 000002          RTI                     ;RETURN
4744 020666 122701 000032    2$:    CMPB    #032,R1        ;SEE IF (^Z) TYPED
4745 020672 001003          BNE      3$              ;BR IF NOT (^Z)
4746 020674 104401 012057    TYPE    ,CNTRLZ         ;ECHO (^Z)
4747 020700 000765          BR       1$              ;RETURN
4748 020702 122701 000022    3$:    CMPB    #022,R1        ;SEE IF (^R) TYPED
4749 020706 001003          BNE      4$              ;BR IF NOT (^R)
4750 020710 104401 012064    TYPE    ,CNTRLR         ;ECHO (^R)
4751 020714 000757          BR       1$              ;RETURN
4752 020716 122701 000007    4$:    CMPB    #007,R1        ;SEE IF (^G) TYPED
4753 020722 001003          BNE      5$              ;BR IF NOT (^G)
4754 020724 104401 012076    TYPE    ,CNTRLG         ;ECHO (^G)
4755 020730 000751          BR       1$              ;RETURN
4756 020732 104401 005442    5$:    TYPE    ,INTCHR         ;ECHO INPUT
4757 020736 104401 001325    TYPE    ,SCRLF          ;DO <CR> AND <LF>
4758 020742 000744          BR       1$              ;RETURN
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772 020744 005077 160176    PREPKB: CLR      @STKB          ;CLEAR KBD BUFFER AND DONE BIT
4773 020750 005037 005442    CLR      INTCHR          ;CLEAR TTY INPUT WORD
4774 020754 052777 000100 160162  BIS      #BIT6,@STKS     ;ENABLE KBD INTERRUPT
4775 020762 000207          RTS      PC              ;RETURN
4776
4777
4778
4779
4780
4781
4782
4783

```

```

:*****
: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
:*****

```

```

:*****
:SBTTL ECHOBAD - 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>

```

4784
 4785
 4786
 4787
 4788 020764 104401 005442
 4789 020770 104401 001324
 4790 020774 000207
 4791
 4792
 4793
 4794
 4795
 4796
 4797
 4798
 4799
 4800
 4801
 4802
 4803 020776 016646 000002
 4804 021002 104403
 4805 021004 006
 4806 021005 000
 4807 021006 104401 011224
 4808 021012 004737 023402
 4809 021016 021050
 4810 021020 021050
 4811 021022 021050
 4812 021024 005700
 4813 021026 001001
 4814 021030 000207
 4815 021032 020027 000006
 4816 021036 003407
 4817 021040 104401 005212
 4818 021044 104401 001324
 4819 021050 162716 000010
 4820 021054 000207
 4821 021056 012746 005212
 4822 021062 004737 042104
 4823 021066 021040
 4824 021070 012600
 4825 021072 005737 042236
 4826 021076 001360
 4827 021100 010066 000002
 4828 021104 000207
 4829
 4830
 4831
 4832
 4833
 4834
 4835
 4836
 4837
 4838
 4839 021106 022737 000176 001140

```

;*AND <LF> ARE DONE.
;* CALL - JSR PC,ECOBAD
;*****
    
```

```

ECOBAD: TYPE ,INTCHR ;ECHO BAD CHARACTER
        TYPE ,SQUES ;TYPE <?> AND <CR>, <LF>
        RTS PC ;RETURN
    
```

```

;*****
;* 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
        TYPOS ;TYPE IT
        .BYTE 6 ;SIX DIGITS
        .BYTE 0 ;SUPPRESS LEADING ZEROS
        TYPE ,NEWMMSG ;TYPE " NEW = "
        JSR PC,RDCHRS ;READ NEW VALUE FROM KBD
        7$ ;(^C) RETURN ADDRESS
        7$ ;(^Z) RETURN ADDRESS
        7$ ;(^U) 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$: TYPE ,BUFF0 ;ECHO BAD INPUT
    TYPE ,SQUES
7$: 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.
;*****
    
```

```

GTSWRG: CMP #SWREG,SWR ;SEE IF SOFTWARE SWR SELECTED
    
```

4840	021114	001010		BNE	6\$;BR IF NOT
4841	021116	013746	000176	MOV	SWREG,-(SP)	;PUT OLD VALUE ON STACK
4842	021122	104401	011215	TYPE	,SWRMSG	;TYPE 'SWR = '
4843	021126	004737	020776	JSR	PC,GETPRM	;TYPE OLD, GET NEW SWREG VALUE
4844	021132	012637	000176	MOV	(SP)+,SWREG	;STORE NEW VALUE
4845	021136	000207		6\$: RTS	PC	;RETURN

4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895

```

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

4864	021140	012737	031472	003056	INITSS: MOV	#ERRHDL,A.ABNL	;SET UP ABNORMAL ERROR RETURN ADDRESS
4865	021146	012737	030250	003054	MOV	#ERRFRE,A.NORM	;SET UP NORMAL RETURN ADDRESS
4866	021154	013702	003046		MOV	RKBAS,R2	;GET ADDRESS OF RK611 REGISTERS
4867	021160	012705	002640		MOV	#PARM0,R5	;GET ADDRESS OF PARAMETER BLOCK
4868	021164	105037	003150		CLRB	NORTRY	;CLEAR 'NO-RETRY' FLAG
4869	021170	004737	021246		JSR	PC,CLRPRM	;CLEAR DRIVER INPUT PARAMETERS
4870	021174	112765	000177	000001	MOVB	#SUBCLR,P.CMND(R5)	;SET SUBSYSTEM CLEAR COMMAND
4871	021202	004737	030124		JSR	PC,DRVCAL	;DO SUBSYSTEM CLEAR
4872	021206	113765	005430	000000	MOVB	DRIVE,P.DRVN(R5)	;SET CURRENT DRIVE NO.
4873	021214	113737	005430	002724	MOVB	DRIVE,PARM1	;SET DRIVE NO. IN ALTERNATE P.B.
4874	021222	113765	003134	000007	MOVB	FORMAT,P.CS1H(R5)	;SET CURRENT DRIVE FORMAT
4875	021230	153765	003133	000007	BISB	TYPFMT,P.CS1H(R5)	;SET DRV TYPE. 0=RK06, 4=RK07
4876	021236	116537	000007	002733	MOVB	P.CS1H(R5),P.CS1H+PARM1	;COPY FORMAT & DR TYPE INTO PARM1
4877	021244	000207			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
:*****
    
```

4888	021246	010046			CLRPRM: MOV	R0,-(SP)	;SAVE R0
4889	021250	010546			MOV	R5,-(SP)	;SAVE R5
4890	021252	010500			MOV	R5,R0	;GET PARAMETER BLOCK ADDRESS
4891	021254	062705	000016		ADD	#P.CS1,R5	;COMPUTE LIMIT ADDRESS
4892	021260	005020			1\$: CLR	(R0)+	;CLEAR A WORD IN PARAMETER BLOCK
4893	021262	020005			CMP	R0,R5	;SEE IF DONE YET
4894	021264	001375			BNE	1\$;BR IF NOT DONE YET
4895	021266	012605			MOV	(SP)+,R5	;RESTORE R5


```

4896 021270 012600          MOV      (SP)+,R0      ;RESTORE R0
4897 021272 000207          RTS        PC          ;RETURN
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916 021274 004737 021140    SCNDRV: JSR      PC,INITSS      ;INITIALIZE DRIVER AND CLEAR SUBSYSTEM
4917 021300 042712 000100    3$:      BIC        #IE,(R2)      ;DISABLE RK06 INTERRUPT
4918 021304 113762 005430 000010    MOV      DRIVE,RKCS2(R2) ;SET DRIVE NO. IN RKCS2
4919 021312 105737 003133    TSTB     TYPFMT          ;SEE IF RK07
4920 021316 001003          BNE      5$              ;BR IF YES
4921 021320 012712 000001    MOV      #GO,(R2)        ;ELSE SET GO BIT FOR RK06 IN RKCS1
4922 021324 000402          BR       7$
4923
4924 021326 012712 002001    5$:      MOV      #<CDT!GO>,(R2) ;SET GO FOR RK07
4925 021332 005037 005444    7$:      CLR        SCRACH        ;CLEAR STALL COUNTER
4926 021336 005237 005444    15$:     INC        SCRACH        ;INCREMENT STALL COUNTER
4927 021342 001375          BNE      15$             ;STALL FOR SEVERAL MILLI-SEC
4928 021344 032762 001000 000010    BIT      #MDS,RKCS2(R2) ;SEE IF MDS ERROR
4929 021352 001417          BEQ      18$             ;BR IF NOT
4930 021354 112765 000101 000001    MOV      #SELDRV,P.CMND(R5) ;SET CMND FOR ERROR REPORT
4931 021362 011265 000016    MOV      (R2),P.CS1(R5) ;GET RKCS1 FOR REPORT
4932 021366 004737 037632    JSR      PC,I.CST1      ;GET OTHER RK611 REGS FOR REPORT
4933 021372 004737 031212    JSR      PC,REPSUP      ;SET UP ERROR REPORT
4934 021376 104052          ERROR    52              ;REPORT MDS ERROR
4935 021400 052737 000200 005424    BIS      #ABORT,RECODE   ;SET ABORT FLAG
4936 021406 000137 033324    JMP      ALLTRM          ;GO ABORT TESTING
4937 021412 004737 021140    18$:     JSR      PC,INITSS      ;INIT THE S.S.
4938 021416 112765 000141 000001    MOV      #RDSTAT,P.CMND(R5) ;SET READ DRIVE STATUS COMMAND
4939 021424 012737 020450 003056    MOV      #NEDHDL,A.ABNL ;SET NED ABNORMAL RETURN ADDRESS
4940 021432 012737 000377 005476    MOV      #377,NEWON     ;INIT. ON-LINE INDICATOR
4941 021440 004737 030124    JSR      PC,DRVCAL      ;READ ALL DRIVE STATUS
4942 021444 112737 000040 011262    MOV      #40,BADDRV+7   ;PUT A SPACE IN MSG BUF
4943
4944 021452 012737 031472 003056    ;SEE IF THIS DRIVE EXISTS AND IS ON-LINE
4945 021460 022737 000377 005476    MOV      #ERRHDL,A.ABNL ;RESTORE ABNL RETURN ADDRESS
4946 021466 001425          CMP      #377,NEWON     ;SEE IF NED INDICATION ON THIS DRIVE
4947 021470 113737 005430 011261    BEQ      4$              ;BR IF NED NOT SET
4948 021476 152737 000060 011261    MOV      DRIVE,BADDRV+6 ;GET DRIVE NO. INTO BUF
4949 021504 104401 011253    BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
4950 021510 104401 011264    TYPE     ,BADDRV        ;TYPE 'DRIVE X'
4951 021514 132737 000200 001351    TYPE     ,NXDRIV        ;TYPE 'NON-EXISTENT'
                                BITB     #BIT7,$ENVM      ;SEE IF UNDER APT

```

```

4952 021522 001401          BEQ      2$          ;BR IF NO
4953 021524 104123          ERROR   123         ;NED UNDER APT SIZING
4954                                ;SERVICE ERRORS HERE
4955 021526 042762 000100 000000 2$: BIC      #IE,RKCS1(R2) ;DISABLE RK06 INTERRUPT
4956 021534 017616 000000      MOV      @ (SP), (SP) ;SET UP ERROR RETURN ADDRESS
4957 021540 000207          RTS      PC          ;ERROR RETURN
4958                                ;SEE IF DRIVE IS READY
4959 021542 032765 000200 000040 4$: BIT      #S.DRY,P.A00(R5) ;TEST FOR DRIVE READY
4960 021550 001013          BNE      6$          ;BR IF DRIVE IS READY
4961 021552 113737 005430 011261      MOVB     DRIVE,BADDRV+6 ;GET DRIVE NO. INTO BUF
4962 021560 152737 000060 011261      BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
4963 021566 104401 011253          TYPE     ,BADDRV      ;TYPE 'DRIVE X'
4964 021572 104401 011303          TYPE     ,NTREDY     ;TYPE 'NOT READY'
4965 021576 000753          BR       2$          ;TAKE ERROR EXIT
4966                                ;SEE IF DRIVE IS WRITE ENABLED
4967 021600 032765 004000 000040 6$: BIT      #S.WRL,P.A00(R5) ;SEE IF WRITE LOCK SET
4968 021606 001413          BEQ      8$          ;BR IF WRITE LOCK NOT SET
4969 021610 113737 005430 011261      MOVB     DRIVE,BADDRV+6 ;GET DRIVE NO.
4970 021616 152737 000060 011261      BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
4971 021624 104401 011253          TYPE     ,BADDRV      ;TYPE 'DRIVE X'
4972 021630 104401 011317          TYPE     ,WRTLOK     ;TYPE 'WRITE-LOCKED'
4973 021634 000734          BR       2$          ;TAKE ERROR EXIT
4974                                ;SEE IF DRIVE NOT LOADED WITH ALIGNMENT CARTRIDGE
4975 021636 112765 000103 000001 8$: MOVB     #PACK,P.CMND(R5) ;SET PACK ACKNOWLEDGE COMMAND
4976 021644 004737 030124          JSR      PC,DRVCAL    ;SET VOLUME VALID
4977 021650 112765 000113 000001      MOVB     #RECAL,P.CMND(R5) ;SET RECAL CMND
4978 021656 004737 030124          JSR      PC,DRVCAL    ;RECAL THE DRIVE
4979 021662 112765 000121 000001      MOVB     #RDATA,P.CMND(R5) ;SET READ COMMAND
4980 021670 013765 024362 000002      MOV      LSTCYL,P.CYLN(R5) ;SET CYLINDER = 632(8)/1456(8)
4981 021676 112765 000002 000005      MOVB     #LSTTRK,P.TRCK(R5) ;SET TRACK = 2
4982 021704 012765 053330 000010      MOV      #RWBUF,P.BALO(R5) ;BUS ADDRESS
4983 021712 012765 177774 000012      MOV      #-4,P.WC(R5) ;READ 4 WORDS
4984 021720 142765 000020 000007      BICB     #B.CFMT,P.CS1H(R5) ;SET 22 SECTOR FORMAT
4985 021726 153765 003133 000007      BISB     TYPFMT,P.CS1H(R5)
4986 021734 004737 030124          JSR      PC,DRVCAL    ;READ 4 WORDS OF BAD SECTOR FILE
4987 021740 032737 100000 005424      BIT      #ANYDER,RECODE ;SEE IF DATA ERROR
4988 021746 001402          BEQ      10$         ;BR IF OK
4989 021750 104401 012006          TYPE     ,BAD632     ;TYPE READ ERROR MESSAGE
4990 021754 022737 177777 053336 10$: CMP      #177777,RWBUF+6 ;SEE IF ALL 1'S IN I.D. WORD 3
4991 021762 001013          BNE      12$         ;BR IF NOT ALL 1'S
4992 021764 113737 005430 011261      MOVB     DRIVE,BADDRV+6 ;GET DRIVE NO.
4993 021772 152737 000060 011261      BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
4994 022000 104401 011253          TYPE     ,BADDRV      ;TYPE 'DRIVE X'
4995 022004 104401 011336          TYPE     ,ALNPAK     ;TYPE 'LOADED WITH ALIGN PACK'
4996 022010 000646          BR       2$          ;TAKE ERROR EXIT
4997                                ;ERROR FREE RETURN
4998 022012 062716 000002 12$: ADD      #2,(SP) ;FIX UP RETURN PC
4999 022016 000207          RTS      PC          ;RETURN
5000
5001
5002
5003
5004
5005
5006
5007 022020 011637 001076      ;*****
;SETUP - SET UP FOR LOOP ON ERROR
;THIS SUBROUTINE CANNOT BE CALLED BY ANY OTHER
;SUBROUTINE --- ONLY MAIN-LINE CODE !!!!
;*****
SETUP: MOV      (SP),@#STACK-2 ;MOVE RETURN PC ON STACK
  
```

```

5008 022024 012706 001076      MOV      #STACK-2,SP      ;RE-INIT THE STACK POINTER
5009 022030 005037 005424      CLR      RECODE           ;CLEAR ERROR RECOVERY FLAGS
5010 022034 105037 003135      CLRB    ERRCNT           ;CLEAR RETRY ERROR COUNT
5011 022040 012705 002640      MOV      #PARMO,R5       ;SET PARAM BLK ADRS
5012 022044 112765 000177 000001  MOVB    #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR CMND
5013 022052 004737 030124      JSR     PC,DRVCAL        ;CLEAR THE SUBSYSTEM
5014 022056 012737 000000 177776  MOV     #PRO,@#PS        ;RE-ESTABLISH PRIORITY 0
5015 022064 000207      RTS      PC              ;RETURN
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029 022066
5030 022066 013700 005554      MOV     DRVPTR,R0        ;GET POINTER TO FIRST DRIVE FOR THIS SUBSYS
5031 022072 112037 010336      MOVB   (R0)+,MNTPAK+22. ;PUT DRIVE NAME INTO MSG
5032 022076 111037 010335      MOVB   (R0),MNTPAK+21.
5033 022102 162700 005512      SUB     #DRVLST,R0      ;SUBTRACT LIST ADRS FROM POINTER
5034 022106 006200      ASR     R0              ;DIVIDE BY TWO TO GET DRIVE NO.
5035 022110 110037 003154      MOVB   R0,LOGDRV        ;STORE LOGICAL DRIVE NUMBER
5036 022114 004737 021140      JSR     PC,INITSS       ;INITIALIZE SUBSYSTEM
5037 022120 042762 000100 000000  BIC    #IE,RKCS1(R2)    ;DISABLE RK06 INTERRUPT FOR MAN. INTERVENTION
5038 022126 104401 010310      10$:   TYPE    ,MNTPAK   ;TYPE MOUNT PACK MSG
5039 022132 104401 010552      5$:   TYPE    ,TYPRWN    ;TYPE 'TYPE R<CR> WHEN ''
5040 022136 104401 010573      TYPE    ,DRIRDY        ;TYPE 'DRIVE READY :''
5041 022142 004737 023402      JSR     PC,RDCHRS       ;WAIT FOR R<CR> RESPONSE
5042 022146 022172      12$:   ;(^C) RETURN ADDRESS
5043 022150 022214      16$:   ;(^Z) RETURN ADDRESS
5044 022152 022132      5$:   ;(^U) RETURN ADDRESS
5045 022154 005700      TST     R0              ;SEE IF NULL INPUT
5046 022156 001025      BNE     8$              ;BR IF NOT NULL
5047 022160 104401 005212      6$:   TYPE    ,BUFFO     ;ECHO BAD INPUT
5048 022164 104401 001324      TYPE    ,SQUES         ;TYPE '<?> AND <CR>,<LF>'
5049 022170 000760      BR      5$              ;GO ASK AGAIN
5050 022172 012706 001100      12$:   MOV     #STACK,SP    ;RE-INIT THE STACK
5051 022176 105737 003126      TSTB   MDFLAG          ;SEE IF 200 START
5052 022202 001002      BNE     14$            ;BR IF NOT
5053 022204 000137 012146      JMP     DFSTRT          ;RESTART- PROGRAM - 200 START
5054 022210 000137 012752      14$:   JMP     ALLSYS          ;GO ASK FOR SUBSYS'S AND DRIVES AGAIN
5055 022214 105737 003126      16$:   TSTB   MDFLAG          ;SEE IF 200 START
5056 022220 001744      BEQ     5$              ;BR IF YES, TO ASK AGAIN
5057 022222 012706 001100      MOV     #STACK,SP      ;RE-INIT THE STACK
5058 022226 000137 013416      JMP     ASKSYS          ;GO ASK FOR SUBSYS TO TEST AGAIN
5059 022232 023727 005212 000122  8$:   CMP     BUFFO,#'R      ;SEE IF R<CR> TYPED
5060 022240 001347      BNE     6$              ;BR IF NOT
5061 022242 004737 021140      JSR     PC,INITSS       ;INITIALIZE S.S.
5062
5063 022246 117737 163302 005430  ;CHECK STATUS OF DRIVE TO BE TESTED NEXT
MOV     @DRVPTR,DRIVE    ;SET DRIVE NUMBER
    
```

```

5064 022254 142737 000060 005430      BICB  #'0,DRIVE      ;STRIP THE ASCII
5065 022262 004737 021274      JSR   PC,SCNDRV     ;CHECK STATUS OF THIS DRIVE -
5066 022266 022126      10$      ; IF NOT VALID, TYPE MSG AND RETURN
5067      ; TO WAIT FOR R<CR> AGAIN
5068      ;READ THE HEADER ON SECTOR 0,TRACK 0,CYL 0, AND GET THE DRIVE FORMAT
5069 022270 004737 021140      JSR   PC,INITSS     ;INIT. DRIVER PARAMS AND S.S.
5070 022274 112765 000125 000001  MOVB  #RDHEAD,P.CMND(R5) ;SET READ HEADER COMMAND
5071 022302 004737 030124      JSR   PC,DRVCAL     ;DO A READ HEADER
5072 022306 013762 024364 000000  MOV   HOLD1,RKCS1(R2) ;DO A READ HDR, WITH FMT BIT = 0
5073 022314 032762 000200 000000 24$:  BIT   #RDY,RKCS1(R2) ;WAIT FOR READY
5074 022322 001774      BEQ   24$
5075 022324 013762 024366 000000  MOV   HOLD2,RKCS1(R2) ;DO A READ HDR, WITH FMT BIT = 1
5076 022332 032762 000200 000000 26$:  BIT   #RDY,RKCS1(R2) ;WAIT FOR READY
5077 022340 001774      BEQ   26$
5078 022342 105037 003134      CLRB  FORMAT        ;INITIALIZE FORMAT BYTE TO 0
5079 022346 005762 000024      TST   RKDB(R2)      ;POP SILO ONE TIME
5080 022352 032762 001000 000024  BIT   #BIT9,RKDB(R2) ;TEST FORMAT BIT IN HEADER WORD 2
5081 022360 001403      BEQ   28$           ;BR IF 22 SECTOR FORMAT
5082 022362 152737 000020 003134  BISB  #B.CFMT,FORMAT ;SET 20 SECTOR FORMAT FOR THIS DRIVE
5083 022370 004737 021140 28$:  JSR   PC,INITSS     ;INIT THE S.S.
5084 022374 004737 026516      JSR   PC,REDBSF     ;READ BAD SECTOR FILE FOR THIS DRIVE
5085 022400 105737 003126      TSTB  MDFLAG        ;SEE IF 200 START
5086 022404 001004      BNE   32$           ;BR IF NOT 200 START
5087 022406 122737 000061 003155  CMPB  #'1,PASSNO    ;SEE IF FIRST PASS
5088 022414 001012      BNE   30$           ;BR IF NOT FIRST PASS
5089 022416 004737 023714 32$:  JSR   PC,DRVSR      ;TYPE 'DRIVE SER. NO.  XXX''
5090 022422 004737 024034      JSR   PC,CRTSER     ;TYPE 'CART. SER. NO.  XXXXXXXXXXXX''
5091 022426 032777 000200 156504  BIT   #BIT7,@SWR    ;SEE IF BAD SECTORS SHOULD BE TYPED
5092 022434 001402      BEQ   30$           ;BR IF NOT
5093 022436 004737 027006      JSR   PC,TYPBSF     ;TYPE BAD SECTOR FILES
5094 022442 005037 005442 30$:  CLR   INTCHR        ;INIT. TTY INPUT CHAR BUFFER
5095 022446 000207      RTS   PC            ;RETURN
5096
5097
5098
5099

```

```

5100      ;*****
5101      ;SBTTL  DMCART - DISMOUNT TEST CARTRIDGE
5102      ;*THIS SUBROUTINE DIRECTS THE OPERATOR IN THE DISMOUNTING
5103      ;*OF THE TEST CARTRIDGE ON THE DRIVE CURRENTLY UNDER TEST.
5104      ;*****
5104 022450      DMCART:
5105 022450 013700 005554      MOV   DRVPTR,RO     ;GET DRIVE LIST POINTER
5106 022454 112037 010373      MOVB  (RO)+,DISPAK+15. ;PUT DRIVE NAME INTO MSG
5107 022460 111037 010372      MOVB  (RO),DISPAK+14.
5108 022464 004737 021140      JSR   PC,INITSS     ;INITIALIZE SUBSYS
5109 022470 042762 000100 000000  BIC   #IE,RKCS1(R2) ;DISABLE RK06 INTRPT FOR MANUAL INTERVENTION
5110 022476 104401 010354      TYPE  ,DISPAK       ;TYPE 'UNLOAD DRIVE XX AND REMOVE PACK''
5111 022502 104401 010552 5$:  TYPE  ,TYPRWN       ;TYPE 'TYPE R<CR> WHEN ''
5112 022506 104401 010612      TYPE  ,DRIDON       ;TYPE 'DONE :''
5113 022512 004737 023402      JSR   PC,RDCHRS     ;WAIT FOR R<CR> RESPONSE
5114 022516 022542      12$      ;(^C) RETURN ADDRESS
5115 022520 022564      16$      ;(^Z) RETURN ADDRESS
5116 022522 022502      5$       ;(^U) RETURN ADDRESS
5117 022524 005700      TST   RO            ;SEE IF NULL INPUT
5118 022526 001025      BNE   8$            ;BR IF NOT NULL
5119 022530 104401 005212 6$:  TYPE  ,BUFFO       ;ECHO BAD INPUT

```

```

5120 022534 104401 001324          TYPE      $QUES      ;TYPE <?> AND <CR>,<LF>
5121 022540 000760          BR          5$          ;GO ASK AGAIN
5122 022542 012706 001100      12$: MOV      #STACK,SP  ;RE-INIT THE STACK
5123 022546 105737 003126          TSTB       MDFLAG      ;SEE IF 200 START
5124 022552 001002          BNE        14$          ;BR IF NOT
5125 022554 000137 012146          JMP        DFSTRT       ;RESTART PROGRAM - 200 START
5126 022560 000137 012752      14$: JMP        ALLSYS     ;GO ASK FOR SUBSYS'S AND DRIVES AGAIN
5127 022564 105737 003126      16$: TSTB       MDFLAG      ;SEE IF 200 START
5128 022570 001744          BEQ        5$          ;BR IF YES, TO ASK AGAIN
5129 022572 012706 001100          MOV      #STACK,SP  ;RE-INIT THE STACK
5130 022576 000137 013416          JMP        ASKSYS       ;GO ASK FOR SUBSYS TO TEST AGAIN
5131 022602 023727 005212 000122 8$: CMP        BUFFO,#'R  ;SEE IF R<CR> TYPED
5132 022610 001347          BNE        6$          ;BR IF NOT
5133 022612 004737 021140          JSR        PC,INITSS   ;INITIALIZE S.S.
5134 022616 000207          RTS          PC          ;RETURN
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146 022620 104407
5147
5148 022622 012700 006552          MOV      #STDLST,R0   ;STARTING ADRS OF STDLST
5149 022626 012701 006112          MOV      #TABLED,R1   ;STARTING ADRS OF TABLED
5150 022632 012703 000020          MOV      #16.,R3      ;SET COUNTER = 16
5151 022636 112011      4$: MOVB     (R0)+,(R1) ;LOAD A BYTE INTO TABLED
5152 022640 062701 000020          ADD      #16.,R1      ;INCREMENT TABLED POINTER
5153 022644 005303          DEC      R3           ;DECR COUNTER
5154 022646 001373          BNE      4$           ;BR IF NOT DONE YET
5155          ;GET INDEX OF FIRST ZERO ENTRY IN DRVLST INTO R1
5156 022650 012700 005512          MOV      #DRVLST,R0   ;STARTING ADRS OF DRVLST
5157 022654 005001          CLR      R1           ;INIT INDEX TO 0
5158 022656 005720      6$: TST      (R0)+      ;SEE IF ENTRY = 0
5159 022660 001404          BEQ      8$           ;BR IF YES
5160 022662 005201          INC      R1           ;INCREMENT POINTER
5161 022664 020127 000020          CMP      R1,#16.      ;SEE IF DONE LOOKING YET
5162 022670 002772          BLT      6$           ;BR IF NOT DONE YET
5163          ;INCREMENT THE DRIVE NOS. TO FORM THE ROWS
5164 022672 012700 006112      8$: MOV      #TABLED,R0   ;INIT TABLED POINTER
5165 022676 012704 000017      16$: MOV      #15.,R4    ;INIT ROW ELEMENT COUNTER
5166 022702 112003          MOVB     (R0)+,R3     ;INIT ROW ELEMENT VALUE
5167 022704 005203      18$: INC      R3         ;INCR ROW ELEMENT VALUE
5168 022706 020327 000020          CMP      R3,#20       ;SEE IF ROW ELEMENT IS VALID
5169 022712 002401          BLT      20$          ;BR IF VALUE IS OK
5170 022714 005003          CLR      R3           ;RESET VALUE TO 0
5171 022716 020301      20$: CMP      R3,R1       ;SEE IF VALUE IS > ACTUAL DRIVES
5172 022720 002403          BLT      22$          ;BR IF VALUE OK
5173 022722 112720 000377          MOVB     #377,(R0)+   ;SET INVALID ENTRY = 377
5174 022726 000401          BR        24$          ;PROCEED
5175 022730 110320      22$: MOVB     R3,(R0)+   ;SET VALUE IN TABLE
  
```

```

;*****
;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 ADRS OF STDLST
          MOV      #TABLED,R1   ;STARTING ADRS OF TABLED
          MOV      #16.,R3      ;SET COUNTER = 16
4$: MOVB     (R0)+,(R1) ;LOAD A BYTE INTO TABLED
          ADD      #16.,R1      ;INCREMENT TABLED POINTER
          DEC      R3           ;DECR COUNTER
          BNE      4$           ;BR IF NOT DONE YET
;GET INDEX OF FIRST ZERO ENTRY IN DRVLST INTO R1
          MOV      #DRVLST,R0   ;STARTING ADRS 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
          MOVB     (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
          MOVB     #377,(R0)+   ;SET INVALID ENTRY = 377
          BR        24$          ;PROCEED
22$: MOVB     R3,(R0)+   ;SET VALUE IN TABLE
  
```

```

5176 022732 005304      24$: DEC R4 ;DECR ROW ELEMENT COUNTER
5177 022734 001363      BNE 18$ ;BR IF NOT DONE WITH ROW YET
5178 022736 020027 006512  CMP RO,#TABLED+256. ;SEE IF DONE WITH TABLE YET
5179 022742 103755      BLO 16$ ;BR IF NOT DONE YET
5180 ;SET INVALID ELEMENTS OF FIRST COLUMN = 377
5181 022744 012700 006112  MOV #TABLED,RO ;INIT COLUMN POINTER
5182 022750 012703 000020  MOV #16.,R3 ;INIT COL. ELEMENT COUNTER
5183 022754 121001      26$: CMPB (RO),R1 ;SEE IF ELEMENT IS VALID
5184 022756 002402      BLT 30$ ;BR IF YES
5185 022760 112710 000377  MOVB #377,(RO) ;SET INVALID ELEMENT = 377
5186 022764 062700 000020  30$: ADD #16.,RO ;INCREMENT COLUMN POINTER
5187 022770 005303      DEC R3 ;SEE IF ALL DONE YET
5188 022772 001370      BNE 26$ ;BR IF NOT ALL DONE YET
5189 022774 104410      RESREG ;RESTORE R0-R5
5190 022776 000207      RTS PC ;RETURN
5191
5192
5193
5194
5195
5196
5197
5198
5199

```

```

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

```

```

5200 023000 104407      ROTBLD: SAVREG ;SAVE R0-R5
5201 023002 012700 006512  MOV #TABLED+256.,RO ;POINT TO END OF TABLED
5202 023006 010001      MOV RO,R1 ;GET A COPY IN R1
5203 023010 012704 000015  4$: MOV #13.,R4 ;INIT ROW ELEMENT COUNTER
5204 023014 114037 001304  MOVB -(RO),$TMP11 ;SAVE LAST 3 ELEMENTS IN THIS ROW
5205 023020 114037 001306  MOVB -(RO),$TMP12
5206 023024 114037 001310  MOVB -(RO),$TMP13
5207 023030 114041      6$: MOVB -(RO),-(R1) ;MOVE AN ELEMENT RIGHT 3 PLACES
5208 023032 005304      DEC R4 ;DECR ROW ELEMENT COUNTER
5209 023034 001375      BNE 6$ ;BR IF NOT DONE YET
5210 023036 113741 001304  MOVB $TMP11,-(R1) ;MOVE IN 3 SAVED ELEMENTS
5211 023042 113741 001306  MOVB $TMP12,-(R1)
5212 023046 113741 001310  MOVB $TMP13,-(R1)
5213 023052 020027 006112  CMP RO,#TABLED ;SEE IF DONE WITH TABLE YET
5214 023056 101354      BHI 4$ ;BR IF NOT DONE YET
5215 023060 104410      RESREG ;RESTORE R0-R5
5216 023062 000207      RTS PC ;RETURN
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228

```

```

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

```

```

5229 023064 104407      WRTBLK: SAVREG ;SAVE R0-R5
5230 023066 113700 003154  MOVB LOGDRV,RO ;GET LOGICAL DRIVE NO.
5231 023072 006300      ASL RO ;CONVERT IT TO DATA PATTERN INDEX

```

```

5232 023074 016003 006512      MOV    TABLEG(R0),R3      ;GET DATA PATTERN WORD INTO R3
5233 023100 012701 006112      MOV    #TABLED,R1         ;GET TABLED POINTER
5234 023104 012704 000020      MOV    #16,R4             ;INIT CYLINDER COUNTER
5235 023110 010100              4$:  MOV    R1,R0
5236 023112 123720 003154      6$:  CMPB   LOGDRV,(R0)+      ;SEE IF DRIVE LISTED HERE
5237 023116 001375              BNE    6$                 ;BR IF NOT FOUND YET IN THIS ROW
5238 023120 005300              DEC    R0                 ;BACK UP BY 1 BYTE
5239 023122 160100              SUB    R1,R0              ;COMPUTE ROW POSITION
5240 023124 116065 006572 000004  MOVB   SECLST(R0),P.SECT(R5) ;GET SECTOR NO. FROM ROW POSITION
5241 023132 105065 000005      CLRB   P.TRCK(R5)         ;SET TRACK = 0
5242 023136 004737 025746      8$:  JSR    PC,WRTSEC         ;WRITE 256 WORDS AT THIS SECTOR
5243 023142 105265 000005      INCB   P.TRCK(R5)         ;INCR TRACK NO.
5244 023146 126527 000005 000002  CMPB   P.TRCK(R5),#2      ;SEE IF ALL TRACKS DONE YET
5245 023154 003770              BLE    8$                 ;BR IF NOT DONE YET
5246 023156 062701 000020      ADD    #16.,R1           ;POINT TO START OF NEXT ROW
5247 023162 005265 000002      INC    P.CYLN(R5)        ;INCR CYLINDER NO.
5248 023166 005304              DEC    R4                 ;DECREMENT CYLINDER COUNTER
5249 023170 001347              BNE    4$                 ;BR IF 16 CYLS NOT DONE YET
5250 023172 104410              RESREG                    ;RESTORE R0-R5
5251 023174 000207              RTS    PC                  ;RETURN

```

5252
5253
5254
5255

```

*****
*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
*****

```

```

5268 023176 104407              CKSCOR: SAVREG            ;SAVE R0-R5
5269 023200 105737 003143      TSTB   TSTTYP             ;SEE IF CURRENTLY TESTING WITH OFFSET
5270 023204 001474              BEQ    20$                ;BR TO EXIT, IF NOT
5271 023206 032737 000024 005424  BIT    #DCKERR!HVR CER,RECODE ;SEE IF A DCK OR HVRC ERR OCCURRED
5272 023214 001470              BEQ    20$                ;BR IF NOT
5273              ;GET THE NO. OF THE DRIVE WHICH WROTE DATA WHICH CAUSED DCK OR HVRC ERR
5274 023216 004737 034276              JSR    PC,GTPKAD          ;COMPUTE PACK ADDRESS OF ERROR
5275 023222 122737 000003 003143  CMPB   #3,TSTTYP          ;SEE IF DOING SELF-TEST
5276 023230 001002              BNE    4$                 ;BR IF NOT
5277 023232 005004              CLR    R4                 ;SET DRIVE NO. = 0
5278 023234 000431              BR     10$                ;CONTINUE
5279 023236 013701 001174      4$:  MOV    $REG5,R1           ;GET CYL NO. OF ERROR
5280 023242 042701 177760      BIC    #177760,R1         ;CLEAR HI CYL BITS
5281 023246 006301              ASL    R1                 ;CONVERT TO TABLED INDEX FOR ROW
5282 023250 006301              ASL    R1
5283 023252 006301              ASL    R1
5284 023254 006301              ASL    R1
5285 023256 062701 006112      ADD    #TABLED,R1         ;COMPUTE POINTER TO TABLED ROW
5286 023262 005000              CLR    R0                 ;INIT SECLST INDEX TO 0
5287 023264 123760 001200 006572 6$:  CMPB   $REG7,SECLST(R0)   ;SEE IF SECTOR IS LISTED IN SECLST

```

```

5288 023272 001405          BEQ      8$          ;BR IF YES
5289 023274 005200          INC      RO          ;INCR SECLST INDEX
5290 023276 020027 000020  CMP      RO,#16.    ;SEE IF WHOLE LIST CHECKED YET
5291 023302 002770          BLT      6$          ;BR IF NOT YET
5292 023304 000434          BR       20$        ;SECTOR NOT LISTED - EXIT, BECAUSE
5293                                ; FAILING DRIVE IS NOT BEING TESTED
5294 023306 060001          8$: ADD     RO,R1     ;COMPUTE TABLED POINTER TO FAILING DRIVE
5295 023310 111104          MOV     (R1),R4     ;GET ACTUAL DRIVE NO.
5296 023312 122704 000377  CMP     #377,R4    ;SEE IF DRIVE IS NOT BEING TESTED
5297 023316 001427          BEQ     20$        ;BR IF DRIVE NOT TESTED
5298 023320 012700 007432  10$: MOV    #NSCORO,RO ;GET ADRS OF NEG SCRATCH SCORES
5299 023324 105737 003156  TST    OFSDIR     ;SEE WHICH OFST DIRECTION
5300 023330 001402          BEQ     12$        ;BR IF NEG
5301 023332 012700 007512  MOV    #PSCORO,RO ;GET ADRS OF POS SCRATCH SCORES
5302 023336 105737 001176  12$: TST    $REG6   ;SEE IF TRACK = 0
5303 023342 001410          BEQ     14$        ;BR IF YES
5304 023344 062700 000020  ADD    #16.,RO     ;POINT TO SCORES FOR TRACK 1
5305 023350 123727 001176 000001  CMP    $REG6,#1    ;SEE IF TRACK = 1
5306 023356 001402          BEQ     14$        ;BR IF YES
5307 023360 062700 000020  ADD    #16.,RO     ;POINT TO SCORES FOR TRACK 2
5308 023364 060400          14$: ADD    R4,RO    ;COMPUTE POINTER TO SCORE FOR DRIVE
5309 023366 121037 003157  CMP    (RO),OFSCNT ;SEE IF THIS SCORE IS PERFECT SO FAR
5310 023372 001001          BNE     20$        ;EXIT IF NOT
5311 023374 105310          DECB   (RO)        ;DECREMENT SCORE FOR THIS DRIVE
5312 023376 104410          20$: RESREG ;RESTORE RO-R5
5313 023400 000207          RTS     PC         ;RETURN
  
```

5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339

```

:*****
: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 BUFFO, TERMINATED BY A NULL (0) BYTE.
:*RUB-OUT AND (^U) 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 RO.
:*
:* CALL - JSR PC,RDCHRS
:*         <CONTROL-C RETURN ADDRESS>
:*         <CONTROL-Z RETURN ADDRESS>
:*         <CONTROL-U OR ERROR RETURN ADDRESS>
:*         RETURN
:*****
  
```

```

5340 023402 010146          RDCHRS: MOV    R1,-(SP)    ;SAVE R1
5341 023402 010246          MOV    R2,-(SP)    ;SAVE R2
5342 023404 005000          CLR    RO          ;INITIALIZE CHARACTER COUNT
5343 023406
  
```



```

5344 023410 005001          CLR      R1          ;INITIALIZE RUB-OUT INDICATOR
5345          ;READ A CHARACTER
5346 023412 104406          RDCHR          ;READ A CHARACTER
5347 023414 112602          MOV      (SP)+,R2    ;GET CHARACTER INTO R2
5348          ;CHECK FOR (^C)
5349 023416 122702 000003    CMPB     #003,R2     ;SEE IF (^C) TYPED
5350 023422 001006          BNE      4$         ;BR IF NOT (^C)
5351 023424 104401 012052    TYPE     ,CNTRLC    ;ECHO (^C)
5352 023430 017666 000004 000004 3$:  MOV      @4(SP),4(SP) ;PUT RETURN ADDRESS ON STACK
5353 023436 000523          BR       24$        ;BR TO TAKE EXIT
5354          ;CHECK FOR (^Z)
5355 023440 122702 000032    4$:  CMPB     #032,R2     ;SEE IF (^Z) TYPED
5356 023444 001006          BNE      6$         ;BR IF NOT (^Z)
5357 023446 104401 012057    TYPE     ,CNTRLZ    ;ECHO (^Z)
5358 023452 062766 000002 000004  ADD     #2,4(SP)    ;MAKE OLD PC POINT TO NEXT RETURN ADR.
5359 023460 000763          BR       3$         ;BR TO TAKE (^Z) EXIT
5360          ;CHECK FOR (^U)
5361 023462 122702 000025    6$:  CMPB     #025,R2     ;SEE IF (^U) TYPED
5362 023466 001006          BNE      8$         ;BR IF NOT (^U)
5363 023470 104401 012071    TYPE     ,CNTRLU    ;ECHO (^U)
5364 023474 062766 000004 000004 7$:  ADD     #4,4(SP)    ;MAKE OLD PC POINT TO NEXT RETURN ADDR.
5365 023502 000752          BR       3$         ;BR TO TAKE (^U) EXIT
5366          ;CHECK FOR (^G)
5367 023504 122702 000007    8$:  CMPB     #007,R2     ;SEE IF (^G) TYPED
5368 023510 001005          BNE      9$         ;BR IF NOT (^G)
5369 023512 104401 012076    TYPE     ,CNTRLG    ;ECHO (^G)
5370 023516 004737 021106    JSR     PC,GTSWRG   ;OPEN SOFTWARE SWITCH REG. FOR CHANGE
5371 023522 000764          BR       7$         ;TAKE (^U) RETURN
5372          ;CHECK FOR RUB-OUT (DELETE)
5373 023524 122702 000177    9$:  CMPB     #177,R2     ;SEE IF RUB-OUT (DEL) TYPED
5374 023530 001020          BNE     14$        ;BR IF NOT RUB-OUT
5375 023532 005700          TST     R0         ;CHECK THE CHARACTER COUNT
5376 023534 001726          BEQ     2$         ;BR IF COUNT = 0
5377 023536 005701          TST     R1         ;CHECK THE RUB-OUT INDICATOR
5378 023540 001003          BNE     11$        ;BR IF WE HAD A PREVIOUS RUB-OUT
5379 023542 005201          INC     R1         ;SET RUB-OUT INDICATOR
5380 023544 104401 012107    TYPE     ,BKSLSH    ;TYPE A BACK-SLASH (\)
5381 023550 005037 005444    11$:  CLR     SCRACH     ;USE SCRATCH WORD FOR TEMP. BUFFER
5382 023554 005300          DEC     R0         ;DECREMENT COUNT
5383 023556 116037 005212 005444  MOV     BUFF0(R0),SCRACH ;GET LAST CHAR. INTO BUFFER
5384 023564 104401 005444    TYPE     ,SCRACH    ;ECHO CHARACTER TO BE DELETED
5385 023570 000710          BR     2$         ;GO READ ANOTHER CHARACTER
5386 023572 005701          14$:  TST     R1         ;CHECK THE RUB-OUT INDICATOR
5387 023574 001403          BEQ     16$        ;BR IF INDICATOR IS NOT SET
5388 023576 104401 012107    TYPE     ,BKSLSH    ;TYPE A BACK-SLASH
5389 023602 005001          CLR     R1         ;CLEAR THE RUB-OUT INDICATOR
5390          ;CHECK FOR CARRIAGE RETURN
5391 023604 122702 000015    16$:  CMPB     #015,R2     ;SEE IF <CR> TYPED
5392 023610 001426          BEQ     19$        ;BR IF <CR>
5393          ;HANDLE POSSIBLE DIGIT
5394 023612 005037 005444    CLR     SCRACH     ;USE SCRATCH WORD FOR TEMP. BUFFER
5395 023616 110237 005444    MOV     R2,SCRACH   ;GET THIS CHARACTER INTO BUFFER
5396 023622 104401 005444    TYPE     ,SCRACH    ;ECHO THE CHARACTER TYPED
5397 023626 110260 005212    MOV     R2,BUFF0(R0) ;PUT CHARACTER INTO BUFFER
5398 023632 005200          INC     R0         ;INCREMENT CHARACTER COUNTER
5399 023634 022700 000120    CMP     #80.,R0    ;SEE IF TOO MANY CHARACTERS TYPED
  
```

```

5400 023640 001264          BNE      2$          ;BR IF NOT TOO MANY
5401 023642 104401 001325    TYPE     ,SCLRF      ;TYPE <CR> AND <LF>
5402 023646 112760 000000 005212    MOVVB   #0,BUFFO(R0) ;PUT TERMINATING NULL INTO BUFFER
5403 023654 104401 005212    TYPE     ,BUFFO      ;ECHO INPUT STRING
5404 023660 104401 001324    TYPE     ,SQUES      ;TYPE <?>,<CR>,<LF>
5405 023664 000703          BR       7$          ;TAKE ERROR EXIT FROM RDCHRS
5406 023666 104401 001325    19$:    TYPE     ,SCLRF      ;TYPE <CR>,<LF>
5407 023672 112760 000000 005212    MOVVB   #0,BUFFO(R0) ;PUT TERMINATING NULL INTO BUFFER
5408 023700 062766 000006 000004    ADD     #6,4(SP)     ;FIX UP RETURN ADDRESS
5409 023706 012602          24$:    MOV      (SP)+,R2    ;RESTORE R2
5410 023710 012601          MOV      (SP)+,R1    ;RESTORE R1
5411 023712 000207          RTS      PC          ;SUBROUTINE EXIT
5412
5413
5414
5415

```

```

*****
.SBTTL DRVSER - TYPE DRIVE SERIAL NUMBER (LOW 3 DIGITS)
*THIS SUBROUTINE TYPES 'DRIVE SER. NO. XXX' (IN DECIMAL), WITH LEADING
*ZEROS SUPPRESSED.
*****

```

```

5419
5420 023714 104407          DRVSER: SAVREG      ;SAVE R0-R5
5421 023716 004737 021140    JSR     PC,INITSS   ;CLEAR S.S. AND PARAMETERS
5422 023722 112765 000141 000001    MOVVB   #RDSTAT,P.CMND(R5) ;SET READ STATUS COMMAND
5423 023730 004737 030124    JSR     PC,DRVCAL   ;READ STATUS OF THIS DRIVE
5424 023734 104401 011701    TYPE     ,DRIV      ;TYPE 'DRIVE'
5425 023740 104401 011717    TYPE     ,SERNM     ;TYPE "SER. NO. "
5426 023744 016501 000054    MOV      P.A11(R5),R1 ;GET 'A' STATUS BYTE 11
5427 023750 012704 043144    MOV      #SOCTVL,R4  ;GET ADDR OF CHAR BUFFER
5428 023754 010446          MOV      R4,-(SP)    ;STORE IT ON STACK FOR $SUPRS
5429 023756 012703 000003    MOV      #3,R3      ;INIT CHAR COUNT
5430 023762 006101          ROL     R1          ;INITIALIZE BIT POSITIONS
5431 023764 006101          ROL     R1
5432 023766 006101          4$:    ROL     R1          ;GET NEXT 4 BITS
5433 023770 006101          ROL     R1
5434 023772 006101          ROL     R1
5435 023774 006101          ROL     R1
5436 023776 010100          MOV     R1,R0       ;GET A WORKING COPY
5437 024000 042700 177760    BIC     #177760,R0   ;CLEAR ALL BUT LOW 4 BITS
5438 024004 052700 000060    BIS     #'0,R0      ;CONVERT A DIGIT TO ASCII
5439 024010 110024          MOVVB   R0,(R4)+    ;PUT ASCII DIGIT INTO CHAR BUFFER
5440 024012 005303          DEC     R3          ;DECREMENT CHAR COUNT
5441 024014 001364          BNE     4$         ;BR IF NOT 3 CHARS YET
5442 024016 105014          CLRB   (R4)        ;INSERT NULL TERMINATOR
5443 024020 004737 043356    JSR     PC,@$SUPRS  ;TYPE DRIVE SER. NUMBER
5444 024024 104401 001325    TYPE     ,SCLRF      ;TYPE <CR> AND <LF>
5445 024030 104410          RESREG          ;RESTORE R0-R5
5446 024032 00C207          RTS      PC          ;RETURN
5447
5448
5449
5450

```

```

*****
.SBTTL CRTSER - TYPE CARTRIDGE SERIAL NUMBER
*THIS SUBROUTINE TYPES 'CART. SER. NO. XXXXXXXXXXXX' (IN OCTAL),
*WITH LEADING ZEROS SUPPRESSED.
*****

```

```

5451
5452
5453
5454
5455 024034 004737 021140    CRTSER: JSR     PC,INITSS ;CLEAR S.S. AND PARAMETERS

```

```

5456 024040 142765 000020 000007      BICB  #B.CFMT,P.CS1H(R5)      ;SET 22 SECTOR FORMAT
5457 024046 105737 003134              TSTB  FORMAT                  ;CHECK THE ACTUAL FORMAT
5458 024052 001402                      BEQ   4$                      ;BR IF 22 SECTORS
5459 024054 105265 000004              INCB  P.SECT(R5)              ;IF 20 SECTORS, READ SECTOR 1
5460 024060 112765 000121 000001 4$:  MOVB  #RDATA,P.CMND(R5)      ;SET READ COMMAND
5461 024066 013765 024362 000002      MOV   LSTCYL,P.CYLN(R5)      ;SET CYL = 632(OCT)/1456(OCT)
5462 024074 112765 000002 000005      MOVB  #LSTTRK,P.TRCK(R5)     ;SET TRACK = 2
5463 024102 012765 053330 000010      MOV   #RWBUF,P.BALO(R5)     ;SET READ BUFFER ADDRESS
5464 024110 012765 177776 000012      MOV   #-2,P.WC(R5)          ;SET WORD COUNT TO READ 2 WORDS
5465 024116 004737 030124              JSR   PC,DRVCAL              ;READ SERIAL NO. IN BSF
5466 024122 104401 011710              TYPE  ,CART                  ;TYPE "CART."
5467 024126 104401 011717              TYPE  ,SERNM                 ;TYPE "SER. NO. "
5468 024132 012746 053330      MOV   #RWBUF,-(SP)          ;GET POINTER FOR $DB20
5469 024136 004737 043042      JSR   PC,@#$DB20            ;CONVERT BINARY TO OCTAL
5470 024142 004737 043356      JSR   PC,@$$SUPRS           ;TYPE CART. SERIAL NO. IN OCTAL
5471 024146 104401 001325              TYPE  ,%CRLF                 ;TYPE <CR> AND <LF>
5472 024152 000207              RTS   PC                     ;RETURN
5473
5474
5475
5476
5477 024154 104401 001325      GETTYP: TYPE  ,%CRLF
5478 024160 104401 007775              TYPE  ,ASKTYP                ;ASK DRIVE TYPE
5479 024164 004737 023402      JSR   PC,RDCHRS             ;READ INPUT STRING
5480 024170 024154              GETTYP                       ;CONT-C RETURN
5481 024172 024154              GETTYP                       ;CONT-Z RET
5482 024174 024154              GETTYP                       ;CONT-U RET
5483 024176 005700      TST   R0                    ;SEE IF CHAR TYPED
5484 024200 001005      BNE   2$                    ;BR IF NO
5485 024202 104401 005212 1$:  TYPE  ,BUFFO                ;ECHO BAD INPUT
5486 024206 104401 001324              TYPE  ,%QUES                 ;TYPE <?>
5487 024212 000760      BR    GETTYP                ;ASK AGAIN
5488
5489 024214 022737 000066 005212 2$:  CMP   #'6,BUFFO             ;SEE IF 6 TYPED
5490 024222 001025      BNE   3$                    ;BR IF NO
5491 024224 105037 003133      CLRB  TYPFMT                 ;SETUP FOR RK06
5492 024230 012737 000632 024362      MOV   #632,LSTCYL
5493 024236 012737 000025 024364      MOV   #25,HOLD1
5494 024244 012737 010025 024366      MOV   #10025,HOLD2
5495 024252 012737 000100 024370      MOV   #100,HOLD3
5496 024260 012737 000060 024372      MOV   #60,HOLD4
5497 024266 012737 000040 024374      MOV   #40,HOLD5
5498 024274 000207      RTS   PC
5499 024276 022737 000067 005212 3$:  CMP   #'7,BUFFO             ;SEE IF 7 TYPED
5500 024304 001336      BNE   1$                    ;BR IF NO
5501 024306 152737 000004 003133      BISB  #B.CDT,TYPFMT         ;SETUP FOR RK07
5502 024314 012737 001456 024362      MOV   #1456,LSTCYL
5503 024322 012737 002025 024364      MOV   #2025,HOLD1
5504 024330 012737 012025 024366      MOV   #12025,HOLD2
5505 024336 012737 000200 024370      MOV   #200,HOLD3
5506 024344 012737 000160 024372      MOV   #160,HOLD4
5507 024352 012737 000100 024374      MOV   #100,HOLD5
5508 024360 000207      RTS   PC
5509
5510 024362 000000      LSTCYL: 0
5511 024364 000000      HOLD1: 0
    
```

5512	024366	000000	HOLD2:	0	
5513	024370	000000	HOLD3:	0	;CYL DIF
5514	024372	000000	HOLD4:	0	;COMPAT START CYL
5515	024374	000000	HOLD5:	0	;HALF CYL DIF

5516
5517
5518
5519
5520
5521
5522
5523

```

:*****
: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
:*****
  
```

5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549

5531	024376	010046			STALL: MOV R0,-(SP)	;SAVE R0
5532	024400	010146			MOV R1,-(SP)	;SAVE R1
5533	024402	032777	000400	154530	BIT #BIT08,@SWR	;APPLY RANDOM STALL ?
5534	024410	001407			BEQ 1\$;BR IF NOT RANDOM
5535	024412	004737	042240		JSR PC,\$RAND	;GENERATE PSEUDO-RANDOM NUMBER
5536	024416	013700	042340		MOV \$LONUM,R0	;GET IT INTO R0
5537	024422	006200			ASR R0	;SCALE IT DOWN
5538	024424	006200			ASR R0	
5539	024426	000403			BR 2\$;GO STALL WITH RANDOM NO.
5540	024430	013700	005432		1\$: MOV STALLS,R0	;GET REQUESTED NO. OF STALLS
5541	024434	001406			BEQ 6\$;RETURN IF NO STALL REQUIRED
5542	024436	012701	000016		2\$: MOV #14.,R1	;SET CONSTANT FOR 40 US
5543	024442	005301			4\$: DEC R1	;INNER LOOP COUNTER
5544	024444	001376			BNE 4\$;INNER LOOP BR
5545	024446	005300			DEC R0	;OUTER LOOP COUNTER
5546	024450	001372			BNE 2\$;OUTER LOOP BR
5547	024452	012601			6\$: MOV (SP)+,R1	;RESTORE R1
5548	024454	012600			MOV (SP)+,R0	;RESTORE R0
5549	024456	000207			RTS PC	;RETURN

5550
5551
5552
5553

```

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

5556
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566
5567

5557	024460	104407			LODSEC: SAVREG	;SAVE R0-R5
5558	024462	012701	053330		MOV #RWBUF,R1	;GET ADRS OF R/W BUF INTO R1
5559	024466	012700	005766		2\$: MOV #TABLEB,R0	;GET DATA PATTERN ADDRESS
5560	024472	012703	000020		MOV #16.,R3	;INIT PATTERN COUNTER
5561	024476	012021			4\$: MOV (R0)+,(R1)+	;MOVE A PATTERN WORD
5562	024500	020127	054330		CMP R1,#RWBUF+512.	;SEE IF DONE YET
5563	024504	103003			BHIS 9\$;BR IF DONE
5564	024506	005303			DEC R3	;DECREMENT PATTERN COUNTER
5565	024510	001372			BNE 4\$;BR IF NOT RESTARTING PATTERN YET
5566	024512	000765			BR 2\$;BR TO RESTART PATTERN
5567	024514	104410			9\$: RESREG	;RESTORE R0-R5

5568 024516 000207 RTS PC ;RETURN

5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5620
5621
5622
5623

024520 104407
024522 013746 005474
024526 005416
024530 005046
024532 116616 000003
024536 005066 000002
024542 116566 000004 000002
024550 066616 000002
024554 005066 000002
024560 012700 000026
024564 105737 003134
024570 001402
024572 012700 000024
024576 020016
024600 101004
024602 160016
024604 005266 000002
024610 000772
024612 112637 005473
024616 005046
024620 116516 000005
024624 066616 000002
024630 005066 000002
024634 122716 000003
024640 101005
024642 162716 000003
024646 005266 000002
024652 000770
024654 112637 005472
024660 066516 000002
024664 011637 005470
024670 005726
024672 104410
024674 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 ;SAVE R0-R5
MOV LASTWC,-(SP) ;STORE WORD COUNT
NEG (SP) ;MAKE IT POSITIVE
18$: CLR -(SP) ;MAKE ROOM ON STACK
MOVB 3(SP),(SP) ;STORE NO. OF SECTORS TRANSFERRED
CLR 2(SP) ;CLEAR LOCATION ON STACK
MOVB P.SECT(R5),2(SP) ;STORE STARTING SECTOR
ADD 2(SP),(SP) ;DETERMINE FINAL SECTOR ADDRESS
CLR 2(SP) ;CLEAR NO. OF TRACKS TRANSFERRED
MOV #22.,R0 ;SET FOR 22 SECTORS
TSTB FORMAT ;DETERMINE THE FORMAT
BEQ 19$ ;BR IF 22 SECTORS
MOV #20.,R0 ;SET FOR 20 SECTORS
19$: CMP R0,(SP) ;CHECK FOR SECTOR OVERFLOW
BHI 20$ ;NO, CHECK IF SECTOR CORRECT
SUB R0,(SP) ;DECREMENT SECTOR COUNT BY 20 OR 22
INC 2(SP) ;INCREMENT TRACKS TRANSFERRED
BR 19$ ;CHECK FOR SECTOR OVERFLOW
20$: MOVB (SP)+,FINSEC ;STORE FINAL SECTOR
CLR -(SP) ;MAKE ROOM FOR TRACKS TRANSFERRED
MOVB P.TRCK(R5),(SP) ;STORE STARTING TRACK
ADD 2(SP),(SP) ;DETERMINE FINAL TRACK ADDRESS
CLR 2(SP) ;CLEAR FINAL CYLINDER
21$: CMPB #3,(SP) ;CHECK FOR TRACK OVERFLOW
BHI 22$ ;NO, CHECK FINAL TRACK
SUB #3,(SP) ;DECREMENT TRACK COUNT BY 3
INC 2(SP) ;INCR CYL COUNT
BR 21$ ;CHECK FOR TRACK OVERFLOW
22$: MOVB (SP)+,FINTRK ;STORE FINAL TRACK
ADD P.CYLN(R5),(SP) ;CALCULATE FINAL CYLINDER
MOV (SP),FINCYL ;STORE FINAL CYLINDER
TST (SP)+ ;CLEAN OFF STACK
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

 .SBTTL LODBUF - LOAD THE READ/WRITE DATA BUFFER

```

5624 ;*THIS SUBROUTINE LOADS THE READ/WRITE DATA BUFFER (POINTED TO BY
5625 ;* PMA AND PMA+2) WITH THE APPROPRIATE REPEATING DATA PATTERN. IF
5626 ;*PARAMETER PATRN = 0 ON ENTRY, THE DATA WHICH IS LOADED IS COMPRISED
5627 ;*OF ALL THE PATTERNS 00-15 (QUICK VERIFY DEFAULT DATA TEST).
5628 ;*IF PATRN IS NOT 0, THE NO. OF WORDS IN THE WORD WDSXFR
5629 ;*OF THE PATTERN IDENTIFIED BY THE NO. OF THE BIT SET IN R1 ON
5630 ;*ENTRY, ARE LOADED INTO THE BUFFER.
5631 ;:*****
5632
5633 024676 104407          LODBUF: SAVREG          ;SAVE R0-R5
5634 024700 013702 005466  MOV      WDSXFR,R2      ;GET NO. OF WORDS
5635 024704 005737 005446  TST      PATRN          ;SEE IF QUICK VERIFY DATA TEST DESIRED
5636 024710 001007          BNE      3$              ;BR IF NOT QUICK VERIFY
5637 024712 012700 005566  MOV      #PAT00,R0      ;SET DATA PATTERN STARTING ADDRESS
5638 024716 012746 000400  MOV      #256.,-(SP)    ;SET PATTERN WORD COUNT
5639 024722 012701 053330  MOV      #RWBUF,R1     ;SET BUFFER ADDRESS
5640 024726 000422          BR       30$             ;PROCEED
5641 024730 005000          3$:   CLR      R0              ;INIT PATTERN NUMBER
5642 024732 032701 000001  4$:   BIT      #BIT0,R1  ;SEE IF THIS BIT IS SET
5643 024736 001003          BNE      6$              ;BR IF THIS BIT IS SET
5644 024740 005200          INC      R0              ;INCREMENT PATTERN NO.
5645 024742 006201          ASR      R1              ;SHIFT TO EXAMINE NEXT BIT
5646 024744 000772          BR       4$              ;BR TO CHECK NEXT BIT
5647 024746 006300          6$:   ASL      R0              ;MULTIPLY PATTERN NO. BY 32(DEC)
5648 024750 006300          ASL      R0
5649 024752 006300          ASL      R0
5650 024754 006300          ASL      R0
5651 024756 006300          ASL      R0
5652 024760 062700 005566  ADD      #PAT00,R0      ;GET ADDRESS OF DESIRED PATTERN
5653 024764 012746 000020  MOV      #16.,-(SP)    ;SET PATTERN WORD COUNT
5654 024770 013701 005504  MOV      PMA,R1         ;SET BUFFER ADDRESS
5655 024774 010003          30$:  MOV      R0,R3         ;GET A COPY OF PATTERN ADDRESS
5656 024776 011604          MOV      (SP),R4        ;INIT PATTERN WORD COUNT
5657 025000 012321          34$:  MOV      (R3)+,(R1)+  ;LOAD A DATA WORD INTO BUFFER
5658 025002 005302          DEC      R2            ;DECREMENT WORD COUNTER
5659 025004 001403          BEQ     44$            ;BR IF ALL DONE
5660 025006 005304          DEC     R4            ;DECREMENT PATTERN WORD COUNT
5661 025010 001373          BNE     34$            ;BR IF NOT DONE WITH PATTERN YET
5662 025012 000770          BR      30$            ;BR TO REPEAT THE PATTERN
5663 025014 005726          44$:  TST      (SP)+        ;POP THE STACK
5664 025016 104410          RESREG          ;RESTORE R0-R5
5665 025020 000207          RTS      PC           ;RETURN
5666
5667
5668
5669 ;:*****
5670 .SBTTL CMPBUF - SOFTWARE COMPARE DATA
5671 ;*THIS SUBROUTINE PERFORMS A SOFTWARE COMPARE OF THE R/W DATA BUFFER (POINTED
5672 ;*TO BY PMA AND PMA+2) TO THE APPROPRIATE REPEATING DATA PATTERN. IF
5673 ;*PARAMETER PATRN = 0 ON ENTRY, THE DATA IS COMPRISED OF ALL THE PATTERNS
5674 ;*00-15 (QUICK VERIFY DEFAULT DATA TEST).
5675 ;*IF PATRN IS NOT 0, THE NO. OF THE BIT SET IN R1 ON ENTRY IDENTIFIES THE
5676 ;*REPEATING DATA PATTERN TO COMPARE AGAINST.
5677 ;:*****
5678 025022 104407          CMPBUF: SAVREG          ;SAVE R0-R5
5679 025024 112737 000040 051653  MOVB    #40,DH701+38.  ;RESTORE ERROR MSG PARAMS
  
```

5680	025032	012737	000007	053222	MOV	#7,DF25+2		
5681	025040	013702	005466		MOV	WDSXFR,R2	:SET NO. OF WORDS	
5682	025044	005037	005444		CLR	SCRACH	:CLEAR COMPARE ERROR COUNT	
5683	025050	004737	031212		JSR	PC,REPSUP	:STORE PREV CMND FOR POSS. PRINT	
5684	025054	005737	005446		TST	PATRN	:SEE IF THIS IS QUICK VERIFY DEFAULT DATA TEST	
5685	025060	001007			BNE	3\$:BR IF NOT	
5686	025062	012700	005566		MOV	#PAT00,R0	:GET DATA PATTERN STARTING ADDR	
5687	025066	012746	000400		MOV	#256.,-(SP)	:SET PATTERN WORD COUNT	
5688	025072	012701	053330		MOV	#RWBUF,R1	:SET BUFFER ADDRESS	
5689	025076	000422			BR	30\$:PROCEED	
5690	025100	005000		3\$:	CLR	R0	:INIT PATTERN NO.	
5691	025102	032701	000001	4\$:	BIT	#BIT0,R1	:SEE IF THIS BIT IS SET	
5692	025106	001003			BNE	6\$:BR IF THIS BIT IS SET	
5693	025110	005200			INC	R0	:INCREMENT PATTERN NUMBER	
5694	025112	006201			ASR	R1	:SHIFT TO EXAMINE NEXT BIT	
5695	025114	000772			BR	4\$:BR TO CHECK NEXT BIT	
5696	025116	006300		6\$:	ASL	R0	:MULTIPLY PATTERN NO. BY 32(DEC)	
5697	025120	006300			ASL	R0		
5698	025122	006300			ASL	R0		
5699	025124	006300			ASL	R0		
5700	025126	006300			ASL	R0		
5701	025130	062700	005566		ADD	#PAT00,R0	:GET ADDRESS OF DESIRED PATTERN	
5702	025134	012746	000020		MOV	#16.,-(SP)	:PATTERN WORD COUNT	
5703	025140	013701	005504		MOV	PMA,R1	:SET BUFFER ADDRESS	
5704	025144	010003		30\$:	MOV	R0,R3	:GET COPY OF PATTERN ADDRESS	
5705	025146	011604			MOV	(SP),R4	:INIT PATTERN WORD COUNT	
5706	025150	022321		34\$:	CMP	(R3)+,(R1)+	:COMPARE DATA WORD TO PATTERN WORD	
5707	025152	001002			BNE	31\$:BR IF COMPARE ERROR	
5708	025154	000137	025514		JMP	40\$:JUMP IF DATA COMPARES OK	
5709	025160	105037	051653	31\$:	CLRB	DH701+38.	:ADJUST DATA HEADER FOR MSG	
5710	025164	012737	000005	053222	MOV	#5,DF25+2	:ADJUST ERROR DATA WORD COUNT	
5711					:COMMON COMPARE	ERROR HANDLER		
5712	025172	010237	001202	35\$:	MOV	R2,\$REG10	:GET WORD NO.	
5713	025176	163737	005466	001202	SUB	WDSXFR,\$REG10		
5714	025204	013737	001202	005474	MOV	\$REG10, LASTWC	:GET 2'S COMP OF WORD NO.	
5715	025212	004737	024520		JSR	PC,FINADR	:COMPUTE ACTUAL PACK ADRS	
5716	025216	104407			SAVREG		:SAVE R0-R5	
5717	025220	013700	005470		MOV	FINCYL,R0	:GET CYL	
5718	025224	113701	005472		MOVB	FINTRK,R1	:GET TRACK	
5719	025230	113702	005473		MOVB	FINSEC,R2	:GET SECTOR	
5720	025234	004737	030020		JSR	PC,BDSRCK	:SEE IF THIS SECTOR LISTED BAD	
5721	025240	104410			RESREG		:RESTORE R0-R5	
5722	025242	032737	001000	005424	BIT	#BADSEC,RECODE		
5723	025250	001116			BNE	50\$:BR IF LISTED- DON'T REPORT ERROR	
5724	025252	005737	005444		TST	SCRACH	:CHECK THE ERROR COUNT	
5725	025256	001010			BNE	36\$:BR IF THIS IS NOT FIRST ERROR	
5726	025260	104034		46\$:	ERROR	34	:TYPE HEADING FOR ERROR MSG	
5727	025262	012737	177777	001174	48\$:	MOV	#-1,\$REG5	:INIT CYL NO.
5728	025270	005037	001176		CLR	\$REG6		
5729	025274	005037	001200		CLR	\$REG7		
5730	025300	005237	005444	36\$:	INC	SCRACH	:INCREMENT THE ERROR COUNT	
5731	025304	032777	000001	153626	BIT	#BIT0,@SWR	:SEE IF ALL ERRORS SHOULD BE REPORTED	
5732	025312	001004			BNE	38\$:BR TO REPORT ALL ERRORS	
5733	025314	022737	000012	005444	CMP	#10.,SCRACH	:SEE IF 10(DEC) ERRORS YET	
5734	025322	002504			BLT	54\$:BR IF ERROR LIMIT EXCEEDED	
5735	025324	023737	001174	005470	38\$:	CMP	\$REG5,FINCYL	:SEE IF DIFFERENT CYL

```

5736 025332 001010          BNE      42$          ;BR IF YES
5737 025334 123737 001176 005472  CMPB    $REG6,FINTRK ;SEE IF DIFFERENT TRACK
5738 025342 001004          BNE      42$          ;BR IF YES
5739 025344 123737 001200 005473  CMPB    $REG7,FINSEC ;SEE IF DIFFERENT SECTOR
5740 025352 001412          BEQ      44$          ;BR IF SAME PACK ADDRESS
5741 025354 013737 005470 001174 42$:    MOV     FINCYL,$REG5  ;SET NEW PACK ADRS FOR PRINTOUT
5742 025362 113737 005472 001176  MOVB    FINTRK,$REG6
5743 025370 113737 005473 001200  MOVB    FINSEC,$REG7
5744 025376 104115          ERROR    115          ;TYPE NEW PACK ADDRESS
5745 025400 005437 001202          44$:    NEG     $REG10        ;GET WORD NO.
5746 025404 016337 177776 001204  MOV     -2(R3),$REG11 ;GET GOOD DATA
5747 025412 016137 177776 001206  MOV     -2(R1),$REG12 ;GET BAD DATA
5748 025420 013737 001202 001212  MOV     $REG10,$REG14 ;COMPUTE PHYSICAL ADDRESS
5749 025426 005037 001210          CLR     $REG13
5750 025432 006137 001212          ROL     $REG14
5751 025436 006137 001210          ROL     $REG13
5752 025442 063737 005504 001212  ADD     PMA,$REG14
5753 025450 005537 001210          ADC     $REG13
5754 025454 063737 005506 001210  ADD     PMA+2,$REG13
5755 025462 010137 001214          MOV     R1,$REG15    ;GET VIRT. ADRS FOR PRINTOUT
5756 025466 162737 000002 001214  SUB     #2,$REG15
5757 025474 104063          ERROR    63          ;TYPE GOOD AND BAD DATA, MEM. ADRS.
5758 025476 032777 000100 153434  BIT     #BIT6,@SWR   ;SEE IF JUST 1 ERROR SHOULD BE REPORTED
5759 025504 001013          BNE     54$          ;BR IF JUST 1 ERROR SHOULD BE REPORTED
5760 025506 005737 005446          50$:    TST     PATRN    ;SEE IF DEFAULT DATA TEST
5761 025512 001400          BEQ     40$          ;BR IF YES
5762 025514 005302          40$:    DEC     R2          ;DECREMENT WORD COUNTER
5763 025516 001406          BEQ     54$          ;BR IF ALL DONE
5764 025520 005304          DEC     R4          ;DECREMENT PATTERN WORD COUNT
5765 025522 001002          BNE     53$          ;BR IF NOT DONE WITH PATTERN YET
5766 025524 000137 025144          JMP     30$          ;JUMP TO REPEAT THE PATTERN
5767 025530 000137 025150          53$:    JMP     34$
5768 025534          54$:
5769 025534 005726          56$:    TST     (SP)+      ;POP THE STACK
5770 025536 104410          RESREG  ;RESTORE R0-R5
5771 025540 000207          RTS     PC          ;RETURN
  
```

```

5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
*****
;* CTLOUT - THIS SUBROUTINE CHECKS FOR (^C) OR (^Z) TTY INPUT.
;* IF (^C) WAS TYPED, THE SUBROUTINE RETURNS TO DRVTST. IF (^Z)
;* WAS TYPED, THE SUBROUTINE RETURNS TO INPUTP. 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
*****
  
```

```

5785 025542 122737 000001 003127 CTLOUT: CMPB    #1,TSTING  ;SEE IF CURRENTLY RUNNING TESTS
5786 025550 001075          BNE     10$          ;BR IF NOT RUNNING TESTS
5787 025552 005737 005442          TST     INTCHR      ;SEE IF ANY TTY INPUT
5788 025556 001472          BEQ     10$          ;BR IF NO INPUT
5789 025560 105737 003126          TSTB   MDFLAG      ;SEE IF DEFAULT MODE RUN
5790 025564 001441          BEQ     12$          ;BR IF YES
5791 025566 122737 000003 005442  CMPB    #003,INTCHR ;SEE IF (^C) TYPED
  
```



```

5792 025574 001026          BNE 4$ ;BR IF NOT (^C)
5793 025576 012700 012752  MOV #ALLSYS,RO ;SET RETURN ADDR = ALLSYS
5794 025602          2$:
5795 025602 000005          6$: RESET ;RESET ALL DEVICES
5796 025604 005037 005442  CLR INTCHR ;CLEAR TTY CHAR BUFFER WORD
5797 025610 105037 003127  CLR TSTING ;CLEAR TTY ESCAPE ALLOW FLAG
5798 025614 012737 031472 003056  MOV #ERRHDL,A.ABNL ;RESTORE ERROR HANDLER ADDRESS
5799 025622 012706 001100  MOV #STACK,SP ;RESET THE STACK
5800 025626 112765 000113 000001  MOVB #RECAL,P.CMND(R5) ;SET RECAL COMMAND
5801 025634 004737 030124  JSR PC,DRVCL ;DO CLEANUP RECALIBRATE
5802 025640 105037 003143  CLR TSITYP ;CLEAR OFFSET TEST IDENTIFIER
5803 025644 005037 001102  CLR $TSTNM ;CLEAR THE TEST NO.
5804 025650 000110  JMP @RO ;EXIT FROM TESTS
5805 025652 122737 000032 005442 4$: CMPB #032,INTCHR ;SEE IF (^Z) TYPED
5806 025660 001021  BNE 7$ ;BR IF NOT (^Z)
5807 025662 012700 013416  MOV #ASKSYS,RO ;SET RETURN ADDR = ASKSYS
5808 025666 000745  BR 2$ ;TAKE EXIT
5809 025670 122737 000003 005442 12$: CMPB #003,INTCHR ;SEE IF (^C) TYPED
5810 025676 001012  BNE 7$ ;BR IF NOT
5811 025700 104401 011537  TYPE ,HLTRQD ;TYPE 'HALT REQUESTED'
5812 025704 104401 011507  TYPE ,CNTRDY ;TYPE 'PRESS CONT WHEN RDY'
5813 025710 012700 033530  MOV #HLTPRG,RO ;SET HALT ADDRESS
5814 025714 112737 000061 003155  MOVB #'1,PASSNO ;RESTORE PASS NO. TO 1
5815 025722 000727  BR 2$ ;TAKE EXIT
5816 025724 122737 000007 005442 7$: CMPB #007,INTCHR ;SEE IF (^G) TYPED
5817 025732 001002  BNE 8$ ;BR IF NOT (^G)
5818 025734 004737 021106  JSR PC,GTSWRG ;OPEN SOFTWARE SWR FOR MODIFICATION
5819 025740 004737 020744 8$: JSR PC,PREPKB ;ENABLE KBD INPUT AGAIN
5820 025744 000207 10$: RTS PC ;RETURN

```

```

5821
5822
5823
5824
5825
5826
5827
5828
5829
5830 025746 104407          *****
5831          ;*WRTSEC - WRITE A BLOCK OF 400(OCT) WORDS ONTO THE DISK.
5832          ;*THE PARAMETER BLOCK MUST BE PRE-LOADED WITH THE PACK
5833          ;*ADDRESS. DATA WORD IS PASSED IN R3 ON ENTRY. RWBUF IS THE BUFFER
5834          ;*USED.
5835          *****
5836          WRTSEC: SAVREG ;SAVE R0-R5
5837          ;LOAD THE R/W BUFFER WITH DATA
5838          MOV #RWBUF,RO ;BUFFER ADDRESS
5839          MOV #400,R1 ;400(OCT) WORDS
5840          4$: MOV R3,(R0)+ ;LOAD A BUFFER WORD
5841          DEC R1 ;DECR COUNTER
5842          BNE 4$ ;BR IF NOT DONE YET
5843          ;SET UP PARAMETERS
5844          MOV #RWBUF,P.BALO(R5) ;SET BUS ADDRESS
5845          MOV #-400,P.WC(R5) ;SET WORD COUNT
5846          ;WRITE THE DATA
5847          MOVB #WRDATA,P.CMND(R5) ;SET WRITE COMMAND
5848          JSR PC,DRVCL ;WRITE THE DATA
5849          RESREG ;RESTORE R0-R5
5850          RTS PC ;RETURN

```

```

5848
5849
5850
5851
5852
5853
5854 026020 104407
5855 026022 005004
5856 026024 012703 000001
5857 026030 000403
5858 026032 104407
5859 026034 012704 177777
5860 026040 013702 005466
5861 026044 013701 005504
5862 026050 010321
5863 026052 005704
5864 026054 001001
5865 026056 005203
5866 026060
5867 026060 005302
5868 026062 001372
5869 026064 104410
5870 026066 000207
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882 026070 104407
5883 026072 005004
5884 026074 012703 000001
5885 026100 000403
5886 026102 104407
5887 026104 012704 177777
5888 026110 013702 005466
5889 026114 013701 005504
5890 026120 005037 005444
5891 026124 112737 000040 051653
5892 026132 012737 000007 053222
5893 026140 004737 031212
5894 026144 020321
5895 026146 001552
5896
5897 026150 010237 001202
5898 026154 163737 005466 001202
5899 026162 013737 001202 005474
5900 026170 004737 024520
5901 026174 104407
5902 026176 013700 005470
5903 026202 113701 005472

```

```

*****
;*LDMEM1 - LOAD MEMORY WITH INCREASING NUMBERS, STARTING WITH 1.
;*LDMEM2 - LOAD MEMORY WITH REPEATED WORD, IN R3 ON ENTRY.
;*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            LDMO        ;PROCEED
LDMEM2: SAVREG          ;SAVE R0-R5
        MOV            #-1,R4      ;R4=177777 INDICATES LDMEM2
LDMO:   MOV            WDSXFR,R2   ;GET NO. OF WORDS
        MOV            PMA,R1     ;GET MEM ADRS
6$:     MOV            R3,(R1)+    ;LOAD A WORD INTO BUFFER
        TST            R4         ;SEE WHICH DATA DESIRED
        BNE            8$        ;BR IF NOT INCREMENTING DATA
        INC            R3         ;INCREMENT THE DATA
8$:
10$:    DEC            R2         ;DECREMENT COUNTER
        BNE            6$        ;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            CKMO        ;PROCEED
CKMEM2: SAVREG          ;SAVE R0-R5
        MOV            #-1,R4      ;R4=177777 INDICATES CKMEM2
CKMO:   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
6$:     CMP            R3,(R1)+    ;COMPARE A DATA WORD
        BEQ            16$        ;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

```

```

5904 026206 113702 005473      MOVB   FINSEC,R2      ;GET SECTOR
5905 026212 004737 030020      JSR    PC,BDSRCK     ;SEE IF THIS SECTOR LISTED BAD
5906 026216 104410      RESREG                                ;RESTORE R0-R5
5907 026220 032737 001000 005424  BIT    #BADSEC,RECODE
5908 026226 001122      BNE    16$           ;BR IF LISTED- DON'T REPORT ERROR
5909 026230 105037 051653      CLRB   DH701+38     ;ADJUST DATA HEADER FOR MSG
5910 026234 012737 000005 053222  MOV    #5,DF25+2    ;ADJ. ERROR DATA WORD COUNT
5911 026242 005737 005444      9$:   TST    SCRACH    ;SEE IF FIRST ERROR IN THIS BLOCK
5912 026246 001010      BNE    10$           ;BR IF NOT FIRST ERROR
5913 026250 104034      17$:  ERROR   34       ;TYPE HEADING FOR ERROR MSG
5914 026252 012737 177777 001174 11$:  MOV    #-1,$REG5    ;INIT CYL NO.
5915 026260 005037 001176      CLR    $REG6
5916 026264 005037 001200      CLR    $REG7
5917 026270 005237 005444      10$:  INC    SCRACH      ;INCREMENT THE ERROR COUNT
5918 026274 032777 000001 152636  BIT    #BIT0,@SWR   ;SEE IF ALL ERRORS SHOULD BE REPORTED
5919 026302 001004      BNE    12$           ;BR TO REPORT ALL ERRORS
5920 026304 022737 000012 005444  CMP    #10.,SCRACH  ;SEE IF 10(DEC) ERRORS YET
5921 026312 002477      BLT    21$           ;BR IF ERROR LIMIT EXCEEDED
5922 026314 023737 001174 005470 12$:  CMP    $REG5,FINCYL ;SEE IF DIFFERENT CYL
5923 026322 001010      BNE    14$           ;BR IF YES
5924 026324 123737 001176 005472  CMPB   $REG6,FINTRK ;SEE IF DIFFERENT TRACK
5925 026332 001004      BNE    14$           ;BR IF YES
5926 026334 123737 001200 005473  CMPB   $REG7,FINSEC ;SEE IF DIFFERENT SECTOR
5927 026342 001412      BEQ    15$           ;BR IF SAME PACK ADDRESS
5928 026344 013737 005470 001174 14$:  MOV    FINCYL,$REG5 ;SET NEW PACK ADR FOR PRINTOUT
5929 026352 113737 005472 001176  MOVB   FINTRK,$REG6
5930 026360 113737 005473 001200  MOVB   FINSEC,$REG7
5931 026366 104115      ERROR  115          ;TYPE NEW PACK ADRS
5932 026370 005437 001202      15$:  NEG    $REG10       ;GET WORD NO.
5933 026374 010337 001204      MOV    R3,$REG11    ;GOOD DATA
5934 026400 016137 177776 001206  MOV    -2(R1),$REG12 ;BAD DATA
5935 026406 013737 001202 001212  MOV    $REG10,$REG14 ;COMPUTE PHYSICAL ADDRESS
5936 026414 005037 001210      CLR    $REG13
5937 026420 006137 001212      ROL    $REG14
5938 026424 006137 001210      ROL    $REG13
5939 026430 063737 005504 001212  ADD    PMA,$REG14
5940 026436 005537 001210      ADC    $REG13
5941 026442 063737 005506 001210  ADD    PMA+2,$REG13
5942 026450 010137 001214      MOV    R1,$REG15
5943 026454 162737 000002 001214  SUB    #2,$REG15
5944 026462 104063      ERROR  63
5945 026464 032777 000100 152446  BIT    #BIT6,@SWR
5946 026472 001007      BNE    21$           ;BR IF JUST 1
5947      ;PROCEED WITH COMPARISONS
5948 026474 005704      16$:  TST    R4           ;SEE WHICH DATA DESIRED
5949 026476 001001      BNE    18$           ;BR IF NOT INCREASING DATA
5950 026500 005203      INC    R3           ;INCREMENT THE DATA WORD
5951 026502      18$:
5952 026502 005302      20$:  DEC    R2           ;DECR COUNTER
5953 026504 001402      BEQ    21$           ;BR IF DONE COMPARING
5954 026506 000137 026144      JMP    6$           ;CONTINUE COMPARING WORDS
5955 026512      21$:
5956 026512 104410      22$:  RESREG
5957 026514 000207      RTS    PC           ;RESTORE R0-R5
5958
5959

```

5960
 5961
 5962
 5963
 5964
 5965
 5966
 5967 026516
 5968 026516 004737 021140
 5969 026522 142765 000020 000007
 5970 026530 112765 000121 000001
 5971 026536 013765 024362 000002
 5972 026544 112765 000002 000005
 5973 026552 012703 000012
 5974 026556 105737 003134
 5975 026562 001403
 5976 026564 105265 000004
 5977 026570 005203
 5978 026572 012765 177400 000012 6\$:
 5979 026600 012765 004162 000010
 5980 026606 012737 026762 003056 8\$:
 5981 026614 105037 003145
 5982 026620 004737 030124
 5983 026624 105737 003145
 5984 026630 001413
 5985 026632 062765 000002 000004
 5986 026640 126503 000004
 5987 026644 001360
 5988 026646 004737 031212 10\$:
 5989 026652 104120
 5990 026654 000137 033530
 5991 026660 012765 003162 000010 12\$:
 5992 026666 110365 000004
 5993 026672 012703 000026
 5994 026676 105737 003134
 5995 026702 001402
 5996 026704 012703 000023
 5997 026710 012737 026762 003056 14\$:
 5998 026716 105037 003145
 5999 026722 004737 030124
 6000 026726 105737 003145
 6001 026732 001407
 6002 026734 062765 000002 000004
 6003 026742 126503 000004
 6004 026746 003760
 6005 026750 000736
 6006 026752 012737 031472 003056 16\$:
 6007 026760 000207
 6008
 6009
 6010
 6011 026762 032765 130600 000034
 6012 026770 001002
 6013 026772 000137 031472
 6014 026776 105237 003145
 6015 027002 000137 033742

```

*****
;*REDBSF - READ FACTORY BSF INTO BSFACT, AND READ SOFTWARE BSF
;*INTO BSSOFT.
*****
REDBSF:
    JSR    PC,INITSS          ;INIT THE S.S.
    BICB   #B.CFMT,P.CS1H(R5) ;SET 22 SECTOR FORMAT
    MOVB   #RDATA,P.CMND(R5) ;SET READ DATA COMMAND
    MOV    LSTCYL,P.CYLN(R5)  ;SET CYL = 632/1456
    MOVB   #2,P.TRCK(R5)     ;SET TRACK = 2
    MOV    #10.,R3           ;SET FACTORY BSF SECTOR LIMIT
    TSTB   FORMAT
    BEQ    6$                ;BR IF 22 SECTORS
    INCB   P.SECT(R5)        ;SET STARTING FACTORY BSF SECTOR NO.
    INC    R3
    MOV    #-400,P.WC(R5)    ;SET WORD COUNT FOR 1 SECTOR
    MOV    #BSFACT,P.BALO(R5) ;SET BA FOR FACTORY BSF
    MOV    #BDSCHD,A.ABNL   ;SET ERROR HANDLER FOR BSF READ
    CLRB   WCEFLG           ;INIT THE ERROR FLAG
    JSR    PC,DRVCAL        ;READ THE FACTORY BSF
    TSTB   WCEFLG           ;SEE IF ANY ERRORS
    BEQ    12$              ;BR IF NOT
    ADD    #2,P.SECT(R5)    ;CHECK NEXT SECTOR
    CMPB   P.SECT(R5),R3    ;SEE IF LIMIT EXCEEDED YET
    BNE    8$                ;BR IF NOT YET
    JSR    PC,REPSUP        ;GATHER STATUS FOR PRINTOUT
    ERROR  120              ;ABORTING- BAD BSF READ
    JMP    HLTPRG           ;HALT- CAN'T PROCEED
    MOV    #BSSOFT,P.BALO(R5) ;SET BA FOR SOFTWARE RECORD
    MOVB   R3,P.SECT(R5)    ;SET STARTING SECTOR NO.
    MOV    #22.,R3         ;SET 22 SECTOR LIMIT
    TSTB   FORMAT          ;SEE IF 22 SECTOR FORMAT
    BEQ    14$              ;BR IF YES
    MOV    #19.,R3         ;SET 20 SECTOR LIMIT
    MOV    #BDSCHD,A.ABNL   ;SET ERROR HANDLER FOR BSF READ
    CLRB   WCEFLG           ;INIT THE ERROR FLAG
    JSR    PC,DRVCAL        ;READ THE SOFTWARE BSF
    TSTB   WCEFLG           ;SEE IF ANY ERRORS
    BEQ    16$              ;BR IF NOT
    ADD    #2,P.SECT(R5)    ;CHECK NEXT SECTOR
    CMPB   P.SECT(R5),R3    ;SEE IF LIMIT EXCEEDED YET
    BLE    14$              ;BR IF NOT YET
    BR     10$              ;REPORT ERROR AND ABORT
    MOV    #ERRHDL,A.ABNL   ;RESTORE ERROR HANDLER ADRS
    RTS    PC                ;RETURN

;*THIS IS THE ABNORMAL DRIVER RETURN WHICH IS USED TO HANDLE
;*POSSIBLE ERRORS IN READING BAD SECTORS.
BDSCHD: BIT   #BSE!HVRC!DTE!OPI!DCK,P.ER(R5) ;SEE IF ANY READ ERRORS
        BNE   4$                ;BR IF A READ ERROR OCCURRED
        JMP   ERRHDL            ;GO HANDLE OTHER ERROR
4$:     INCB  WCEFLG           ;SET ERROR FLAG
        JMP   RETNML           ;TAKE NORMAL DRIVER RETURN
    
```

6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071

027006 104407
027010 005004
027012 012700 004172
027016 104401 011732
027022 104401 011757
027026 104401 051420
027032 104401 001325
027036 005001
027040 005710
027042 100007
027044 005701
027046 001032
027050 104401 011777
027054 104401 001325
027060 000425
027062 012046
027064 104402
027066 104401 012117
027072 011003
027074 105003
027076 000303
027100 010346
027102 104402
027104 104401 012117
027110 111003
027112 010346
027114 104402
027116 104401 001325
027122 005720
027124 005201
027126 020127 000176
027132 002742
027134 005704
027136 001006
027140 005204
027142 012700 003172
027146 104401 011745
027152 000725
027154 104410
027156 000207

```

*****
;*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 R0-R5
          CLR           R4          ;INIT BSF FILE INDICATOR
          MOV          #BSFACT+10,R0 ;ADRS OF DATA IN FACTORY BSF
          TYPE        ,FACTBS      ;TYPE "FACTORY"
          TYPE        ,BDSECT      ;TYPE "BAD SECTORS"
3$:      TYPE        ,DH6041      ;TYPE "CYLNR TRACK SECTOR"
          TYPE        ,$CRLF
          CLR           R1          ;INIT LIMIT COUNTER
4$:      TST           (R0)        ;SEE IF A SECTOR LISTED HERE
          BPL          8$          ;BR IF YES
          TST          R1          ;SEE IF FILE IS EMPTY
          BNE          14$        ;BR IF NOT EMPTY
          TYPE        ,NOFALS      ;TYPE "NONE"
          TYPE        , $CRLF      ;TYPE <CR>,<LF>
          BR           14$
8$:      MOV          (R0)+,-(SP)   ;GET A CYL NO.
          TYPOC
          TYPE        ,SPACE2     ;TYPE SEPARATORS
          MOV          (R0),R3     ;GET TRACK AND SECTOR
          CLRB        R3
          SWAB        R3          ;GET THE TRACK NO.
          MOV          R3,-(SP)
          TYPOC
          TYPE        ,SPACE2     ;TYPE SEPARATORS
          MOV          (R0),R3     ;GET SECTOR NO.
          MOV          R3,-(SP)
          TYPOC
          TYPE        , $CRLF
          TST          (R0)+
          INC          R1          ;INCR THE POINTER
          INC          R1          ;INCR THE COUNTER
          CMP          R1,#126.    ;SEE IF LIMIT REACHED IN THIS FILE
          BLT          4$
14$:     TST          R4          ;BR IF NOT YET
          BNE          18$        ;SEE IF BOTH FILES TYPED YET
          INC          R4          ;BR IF YES
          MOV          #BSOFT+10,R0 ;SET INDICATOR FOR SOFT. FILE
          TYPE        ,SOFTBS     ;ADRS OF DATA IN SOFTWARE BSF
          BR           3$         ;TYPE "SOFTWARE BAD SECTORS :'"
18$:     RESREG
          RTS          PC         ;GO TYPE SOFT. BSF
          ;RESTORE R0-R5
          ;RETURN
*****
;*FINMEM - COMPUTE NO. OF WORDS XFERRED ON PARTIAL XFER
;*(TERMINATED BY BSE ERROR), AND STORE IT IN WDSXFR.
*****

```

6072 027160 104407
 6073 027162 016500 000002
 6074 027166 116501 000005
 6075 027172 116502 000004
 6076 027176 005003
 6077 027200 026500 000030
 6078 027204 001006
 6079 027206 126501 000027
 6080 027212 001003
 6081 027214 126502 000026
 6082 027220 001404
 6083 027222 005203
 6084 027224 004737 027244
 6085 027230 000763
 6086 027232 000303
 6087 027234 010337 005466
 6088 027240 104410
 6089 027242 000207

```

FINMEM: SAVREG          ;SAVE R0-R5
MOV          P.CYL(R5),R0 ;GET ORIG. CYL NO.
MOVB        P.TRCK(R5),R1 ;GET TRACK NO.
MOVB        P.SECT(R5),R2 ;SECTOR NO.
CLR         R3           ;INIT SECTOR COUNT TO 0
6$: CMP      P.DCYL(R5),R0 ;COMPARE ORIG. AND CURRENT CYLS
BNE         8$          ;BR IF NOT EQUAL
CMPB       P.DTS+1(R5),R1 ;COMPARE TRACKS
BNE         8$          ;BR IF NOT EQUAL
CMPB       P.DTS(R5),R2  ;COMPARE SECTORS
BEQ        12$         ;BR IF ADRS ARE EQUAL
8$: INC      R3          ;INCR SECTOR COUNT
JSR        PC,INCRSC    ;INCR PACK ADRS BY 1 SECTOR
BR         6$          ;GO COMPARE ADDRESSES
12$: SWAB   R3           ;COMPUTE NO. OF WORDS XFERRD
MOV        R3,WDSXFR    ;STORE IT
RESREG     PC          ;RESTORE R0-R5
RTS        PC          ;RETURN
  
```

6090
 6091
 6092
 6093
 6094
 6095
 6096
 6097

```

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

6098 027244 012704 000025
 6099 027250 105737 003134
 6100 027254 001402
 6101 027256 012704 000023
 6102 027262 005202
 6103 027264 020204
 6104 027266 003407
 6105 027270 005002
 6106 027272 005201
 6107 027274 020127 000002
 6108 027300 003402
 6109 027302 005001
 6110 027304 005200
 6111 027306 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
4$: INC     R2          ;INCR. SECTOR NO.
CMP       R2,R4       ;SEE IF SECTOR LIMIT EXCEEDED
BLE      6$          ;BR IF NOT
CLR       R2          ;SET SECTOR = 0
INC       R1          ;INCR. TRACK NO.
CMP       R1,#2       ;SEE IF TRACK LIMIT EXCEEDED
BLE      6$          ;BR IF NOT
CLR       R1          ;SET TRACK = 0
INC       R0          ;INCR. CYLINDER NO.
6$: RTS     PC         ;RETURN
  
```

6112
 6113
 6114
 6115
 6116
 6117
 6118
 6119
 6120
 6121

```

;*****
;*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).
;*****
  
```

6122 027310 104407
 6123 027312 004737 027160
 6124 027316 016500 000030
 6125 027322 116501 000027
 6126 027326 116502 000026
 6127 027332 013703 005466

```

MIDXFR: SAVREG          ;SAVE R0-R5
JSR        PC,FINMEM   ;COMPUTE NO. OF WORDS XFERRD
MOV        P.DCYL(R5),R0 ;GET CYL NO.
MOVB      P.DTS+1(R5),R1 ;GET TRACK NO.
MOVB      P.DTS(R5),R2  ;GET SECTOR NO.
MOV        WDSXFR,R3    ;GET NO. OF WORDS TRANSFERRED
  
```

```

6128 027336 032737 000020 005424 BIT #DCKERR,RECODE ;SEE IF DATA CHECK ERROR OCCURRED
6129 027344 001004 BNE 9$ ;BR IF YES
6130 027346 062703 000400 ADD #400,R3 ;SKIP BAD SECTOR
6131 027352 004737 027244 JSR PC,INCRSC ;INCREMENT PACK ADRS TO NEXT SECTOR
6132 027356 010065 000002 9$: MOV R0,P.CYLN(R5) ;UPDATE CYLINDER
6133 027362 110165 000005 MOVVB R1,P.TRCK(R5) ;UPDATE TRACK
6134 027366 110265 000004 MOVVB R2,P.SECT(R5) ;UPDATE SECTOR
6135 027372 060365 000012 ADD R3,P.WC(R5) ;UPDATE P.WC(R5)
6136 027376 032765 100000 000014 BIT #DTBAII,P.PRST(R5) ;SEE IF BUS ADRS INCR INHIB SET
6137 027404 001022 BNE 16$ ;BR IF YES
6138 027406 005004 CLR R4 ;GET BYTES XFERRED
6139 027410 006103 ROL R3
6140 027412 006104 ROL R4
6141 027414 060337 005504 ADD R3,PMA ;UPDATE PMA,PMA+2
6142 027420 005537 005506 ADC PMA+2
6143 027424 060437 005506 ADD R4,PMA+2
6144 027430 013765 005504 000010 MOV PMA,P.BALO(R5) ;UPDATE P.BALO(R5)
6145 027436 013700 005506 MOV PMA+2,R0
6146 027442 042700 177774 BIC #177774,R0
6147 027446 150065 000007 BISB R0,P.BAHI(R5) ;UPDATE P.BAHI(R5)
6148 027452 104410 16$: RESREG ;RESTORE R0-R5
6149 027454 000207 RTS PC ;RETURN
  
```

```

6150
6151
6152
6153
6154
6155
6156
6157 027456 104407
6158 027460 012700 002642
6159 027464 012701 005454
6160 027470 016537 000012 005466
6161 027476 005437 005466
6162 027502 012737 053330 005504
6163 027510 005037 005506
6164 027514 000405
6165 027516 104407
6166 027520 012700 005454
6167 027524 012701 002642
6168 027530 012702 000005
6169 027534 012021
6170 027536 005302
6171 027540 001375
6172 027542 104410
6173 027544 000207
  
```

```

*****
;SVPRMS - SAVE INITIAL PARAMETERS FOR TRANSFER
;GTPRMS - RESTORE SAVED TRANSFER PARAMETERS
*****
SVPRMS: SAVREG ;SAVE R0-R5
MOV #PARMO+2,R0 ;ADRS OF PARAMS
MOV #SAVPRS,R1 ;ADRS OF SAVE AREA
MOV P.WC(R5),WDSXFR ;GET WORD COUNT
NEG WDSXFR ;MAKE IT POSITIVE
MOV #RWBUF,PMA ;INIT PMA TO RWBUF
CLR PMA+2 ;INIT PMA+2 TO 0
BR GT0 ;CONTINUE
GTPRMS: SAVREG ;SAVE R0-R5
MOV #SAVPRS,R0 ;ADRS OF SAVE AREA
MOV #PARMO+2,R1 ;ADRS OF PARAMS
GT0: MOV #5,R2 ;SET FOR 5 WORDS
6$: MOV (R0)+,(R1)+ ;MOVE A WORD
DEC R2 ;SEE IF DONE YET
BNE 6$ ;BR IF NOT YET
RESREG ;RESTORE R0-R5
RTS PC ;RETURN
  
```

```

6174
6175
6176
6177
6178
6179
6180
6181
6182
6183
*****
;TRNSFR - 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
  
```

```

6184
6185
6186 027546 010446
6187 027550 005004
6188 027552 105037 003144
6189 027556 005037 005424
6190 027562 012737 020526 003056
6191 027570 105037 003135
6192 027574 004737 030124
6193 027600 032737 000026 005424
6194 027606 001410
6195 027610 005204
6196 027612 004737 023176
6197 027616 004737 027310
6198 027622 005765 000012
6199 027626 001353
6200 027630 005704
6201 027632 001404
6202 027634 004737 027516
6203 027640 004737 027456
6204 027644 012604
6205 027646 012737 031472 003056
6206 027654 000207
6207
6208
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6220
6221 027656 010237 001266
6222 027662 010337 001270
6223 027666 012637 001272
6224 027672 011402
6225 027674 012437 001264
6226 027700 062737 001000 001264
6227 027706 005003
6228 027710 062702 000010
6229 027714 005712
6230 027716 100430
6231 027720 020012
6232 027722 001406
6233 027724 062702 000004
6234 027730 020237 001264
6235 027734 002021
6236 027736 000766
6237 027740 005722
6238 027742 011237 001302
6239 027746 042737 174377 001302
  
```

```

;*TESTS.
*****
TRNSFR: MOV R4,-(SP) ;SAVE R4 ON STACK
        CLR R4 ;CLEAR ERROR INDICATOR
        CLR DCEFLG ;CLEAR DCK ERROR FLAG
4$: CLR RECODE ;CLEAR ERROR FLAGS
    MOV #DCKHDL,A.ABNL ;SET DCK ERROR HANDLER ADRS
    CLR ERRCNT ;CLEAR ERROR RETRY COUNT
    JSR PC,DRVCL ;PERFORM THE S.S. FUNCTION
    BIT #BSERR!DCKERR!HVR CER,RECODE ;SEE IF BSE OR DCK OR HVRC ERROR OCCURRED
    BEQ 6$ ;BR IF NOT
    INC R4 ;INCR ERROR INDICATOR
    JSR PC,CKSCOR ;DECR. DRIVE SCORE, IF NECESSARY
    JSR PC,MIDXFR ;UPDATE PARAMS TO RESUME XFER
    TST P.WC(R5) ;SEE IF ENTIRE XFER IS COMPLETED
    BNE 4$ ;BR IF NOT
6$: TST R4 ;SEE IF ANY ERRORS OCCURRED
    BEQ 8$ ;BR IF NOT
    JSR PC,GTPRMS ;RESTORE ORIGINAL PARAMS OF XFER
    JSR PC,SVPRMS ;RESTORE WDSXFR,PMA,PMA+2
8$: MOV (SP)+,R4 ;RESTORE R4
    MOV #ERRHDL,A.ABNL ;RESTORE NORMAL ERROR HANDLER ADRS
    RTS PC ;RETURN
  
```

```

*****
.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.
*****
  
```

```

SRHTBS: MOV R2,$TMP2
        MOV R3,$TMP3
        MOV (SP)+,$TMP4 ;STORE RETURN CONTENTS OF R4
        MOV (R4),R2 ;GET ADDRESS OF BAD SECTOR TABLE TO SEARCH
        MOV (R4)+,$TMP1 ;SETUP FOR LENGTH OF BAD SECTOR TABLE
        ADD #1000,$TMP1
        CLR R3 ;CLEAR R3 FOR COUNT
        ADD #10,R2 ;SET R2 FOR POINTER TO CYLINDER ENTRY
1$: TST (R2) ;TEST IF ALL ONES
    BMI 5$ ;YES-DONE
    CMP R0,(R2) ;TEST IF BAD SECTOR IN PRESENT CYL
    BEQ 3$ ;YES-GO CHECK TRACK
    ADD #4,R2 ;ELSE BUMP POINTER
    CMP R2,$TMP1 ;TEST IF OUT OF TABLE
    BGE 5$ ;YES-EXIT
    BR 1$ ;LOOP
3$: TST (R2)+ ;TRACK CHECK - PUT POINTER AT TRK/SEC BYTE
    MOV (R2),$TMP10 ;GET TRK/SEC WORD
    BIC #174377,$TMP10 ;CLEAR ALL BUT TRACK
  
```



```

6240 027754 123701 001303      CMPB   $TMP10+1,R1      ;CHECK IF BAD SECTOR IN THIS TRACK
6241 027760 001402              BEQ    4$              ;YES - GO PUT SECTOR NUMBER ON STACK
6242 027762 005722              TST   (R2)+           ;ELSE BUMP POINTER TO NEXT CYL WORD
6243 027764 000753              BR    1$              ;GO TEST NEXT CYL
6244 027766 005203      4$:   INC    R3          ;BUMP BAD SECTOR COUNT
6245 027770 012246              MOV   (R2)+,-(SP)     ;PUT TRK/SEC WORD ON STACK
6246 027772 042716 177700      BIC   #177700,(SP)    ;CLEAR ALL BUT SECTOR NUMBER
6247 027776 000746              BR    1$              ;GO CHECK REST OF FILE
6248 030000 010346      5$:   MOV   R3,-(SP)      ;EXIT - PUT NUMBER OF BAD
6249 030002 013702 001266      MOV   $TMP2,R2        ;SECTORS ON STACK-RESTORE
6250 030006 013703 001270      MOV   $TMP3,R3        ;REGISTERS
6251 030012 013746 001272      MOV   $TMP4,-(SP)    ;PUT RETURN ON STACK
6252 030016 000204              RTS    R4             ;RETURN
6253
6254      ;*****
6255      .SBTTL  BAD SECTOR CHECK ROUTINE
6256      ;*THIS ROUTINE WILL SEARCH BOTH TABLES TO DETERMINE IF A
6257      ;*SPECIFIC SECTOR IS LISTED AS BAD. IF THE SECTOR IS LISTED IN THE
6258      ;*TABLES THE ROUTINE SETS THE 'BADSEC' FLAG AND RETURNS. IF THE SECTOR
6259      ;*IS NOT LISTED THE FLAG IS RESET.
6260      ;*****
6260 030020 104407      BDSRCK: SAVREG
6261 030022 012737 004162 030034      MOV   #BSFACT,1$     ;SET TABLE TO SEARCH
6262 030030 004437 027656      2$:   JSR   R4,SRHTBS   ;GO SEARCH IT
6263 030034 000000      1$:   .WORD
6264 030036 012603              MOV   (SP)+,R3       ;TABLE ADDRESS GOES HERE
6265 030040 001015              BNE   6$              ;GET NUMBER OF BAD SECTORS, IF ANY
6266 030042 023727 030034 003162 7$:   CMP   1$,#BSSOFT     ;IF ANY, GO TEST WHICH ONES
6267 030050 001404              BEQ   3$              ;ELSE TEST IF OTHER TABLE ALREADY SEARCHED
6268 030052 012737 003162 030034      MOV   #BSSOFT,1$     ;IF YES-EXIT
6269 030060 000763              BR    2$              ;SET OTHER TABLE FOR SEARCH
6270 030062 042737 001000 005424 3$:   BIC   #BADSEC,RECODE ;GO SEARCH IT
6271 030070 104410      4$:   RESREG
6272 030072 000207              RTS   PC              ;CLEAR BAD SECTOR BIT
6273 030074 022602      6$:   CMP   (SP)+,R2     ;RETURN
6274 030076 001403              BEQ   8$              ;THIS SECTOR IN TABLE?
6275 030100 005303              DEC   R3              ;YES-GO SET BIT & EXIT
6276 030102 001374              BNE   6$              ;DECREMENT BAD SECTOR COUNT
6277 030104 000756              BR    7$              ;IF NOT ZERO-CHECK NEXT ENTRY
6278 030106 052737 001000 005424 8$:   BIS   #BADSEC,RECODE ;ELSE GO SEARCH OTHER TABLE
6279 030114 005303      9$:   DEC   R3              ;SET BAD SECTOR BIT
6280 030116 001764              BEQ   4$              ;CLEAR STACK OF OTHER BAD SECTOR
6281 030120 005726              TST   (SP)+           ;NUMBER
6282 030122 000774              BR    9$              ;EXIT
6283
6284
6285
6286
6287
6288      ;*****
6288      .SBTTL  CALL DRIVER ROUTINE
6289      ;*ENTRY JSR PC,DRVCAL
6290      ;*      WITH R5 POINTING TO PARAMETER BLOCK
6291      ;*RETURN RTS PC
6292      ;*
6293      ;*THIS ROUTINE IS USED TO INITIATE A SUBSYSTEM OPERATION BY
6294      ;*CALLING THE DRIVER. THE PARAMETER BLOCK MUST BE SET UP
6295      ;*BY THE CALLER AND R5 MUST POINT TO THE PARAMETER

```

6296
 6297
 6298
 6299
 6300
 6301
 6302
 6303
 6304 030124
 6305 030124 004737 024376
 6306 030130 004737 025542
 6307 030134 105037 003132
 6308 030140 105037 003141
 6309 030144 004737 030176
 6310 030150 010537 030160
 6311 030154 004737 040210
 6312 030160 000000
 6313 030162 004737 034560
 6314 030166 105737 003132
 6315 030172 001773
 6316 030174 000207
 6317
 6318
 6319
 6320
 6321
 6322
 6323 030176 104407
 6324 030200 012701 005162
 6325 030204 012700 005176
 6326
 6327 030210 012021
 6328 030212 012021
 6329 030214 012021
 6330 030216 012021
 6331 030220 012021
 6332 030222 012021
 6333
 6334 030224 012701 005176
 6335 030230 012521
 6336 030232 012521
 6337 030234 012521
 6338 030236 012521
 6339 030240 012521
 6340 030242 012521
 6341 030244 104410
 6342 030246 000207
 6343
 6344
 6345
 6346
 6347
 6348
 6349
 6350
 6351

```

;*BLOCK WHEN THE ROUTINE IS CALLED.
;*
;*THIS ROUTINE WAITS FOR THE OPERATION TO BE COMPLETED.
;*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 (^C) OR (^Z) KBD INPUT
      CLRB     DONE        ;CLEAR DONE FLAG
      CLRB     DRNAFG       ;CLEAR DRIVE SIEZED BY OTHER PORT FLAG
      JSR      PC,STRCMD    ;STORE PREV AND CURRENT COMMANDS
      MOV      R5,4$        ;GET PARAM BLOCK ADDRESS
      JSR      PC,C.INIT    ;CALL DRIVER
4$:   .WORD    ;P.B. ADDRESS GOES HERE
6$:   JSR      PC,W.WTCH    ;CALL WATCH DOG
      TSTB     DONE        ;DONE SET?
      BEQ      6$          ;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)+
;STORE CURRENT COMMAND
4$:   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.

```

```

6352
6353 030250 152737 000377 003132
6354 030256 032737 100000 005424
6355 030264 001403
6356 030266 105037 003135
6357 030272 000413
6358 030274 105737 003135
6359 030300 001410
6360 030302 005037 001174
6361 030306 113737 003135 001174
6362 030314 104101
6363 030316 105037 003135
6364 030322 005037 005424
6365 030326 000207
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376 030330 104407
6377 030332 105237 003137
6378 030336 032777 020000 150574
6379 030344 001402
6380 030346 000137 030752
6381 030352 005000
6382 030354 005005
6383 030356 005105
6384 030360 113700 001114
6385 030364 005300
6386 030366 006300
6387 030370 006300
6388 030372 006300
6389 030374 062700 001410
6390 030400 012037 030420
6391 030404 001406
6392 030406 104401 012103
6393 030412 104401 045400
6394 030416 104401
6395 030420 000000
6396 030422 012037 030576
6397 030426 001467
6398 030430 104401 001325
6399 030434 104401 045366
6400 030440 013746 001102
6401 030444 104403
6402 030446 002
6403 030447 000
6404 030450 104401 012103
6405 030454 032777 010000 150456
6406 030462 001133
6407 030464 104401 050522

```

```

*****
ERRFRE: BISB #377,DONE ;SET THE DONE FLAG
        BIT #ANYDER,RECODE ;TEST !! ANY DATA ERROR
        BEQ 2$ ;IF NO - DO ERROR RECOVERY PRINT TEST
        CLRB ERRCNT ;CLEAR ERROR COUNT
        BR 1$ ;EXIT
2$: TSTB ERRCNT ;CHECK IF ANY ERRORS HAVE OCCURRED
   BEQ 1$ ;NO - SKIP TO EXIT
   CLR $REG5
   MOVB ERRCNT,$REG5 ;GET RETRY COUNT
   ERROR 101 ;PRINT RETRY SUCCESSFUL MESSAGE
   CLRB ERRCNT ;CLEAR ERROR COUNT
1$: CLR RECODE ;CLEAR RECOVERY FLAGS
   RTS PC ;RETURN
*****
.SBTTL TYPE ERROR ROUTINE
;*ENTRY - JSR PC,TYPERR
;*RETURN - RTS PC
;*
;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN USES THE "ERROR TABLE" ($ERRTB)
;*ENTRY TO DEFINE WHAT INFORMATION IS TO BE REPORTED CONCERNING
;*THE ERROR.
*****
TYPERR: SAVREG
        INCB DRVERS ;INCR ERROR COUNT FOR THIS DRIVE
9$: BIT #SW13,@SWR ;INHIBIT ERROR TYPEOUTS?
   BEQ 6$ ;BR IF NO
   JMP 20$
6$: CLR R0 ;CLR R0 FOR ERROR NUMBER
   CLR R5 ;INIT INDENT INDICATOR
   COM R5
   MOVB $ITEMB,R0 ;ENTER ERROR NUMBER
   DEC R0 ;FORM INDEX FOR ERROR TABLE
   ASL R0
   ASL R0
   ASL R0
1$: ADD # $ERRTB,R0 ;FORM ADDRESS OF ERROR ENTRY
   MOV (R0)+,2$ ;GET EM POINTER
   BEQ 3$ ;BRANCH IF THERE ISN'T ONE
   TYPE ,CR2LF
   TYPE ,AS2SP2 ;TYPE "*"
   TYPE ERROR MESSAGE (EM)
2$: .WORD 0 ;EM POINTER GOES HERE
3$: MOV (R0)+,4$ ;GET DH POINTER
   BEQ 5$ ;BR IF THERE ISN'T ONE
   TYPE , $CRLF
   TYPE ,TSTMSG ;TYPE " TEST "
   MOV $TSTNM,-(SP) ;GET TEST NO. ON STACK
   TYPOS ;TYPE IT
   .BYTE 2 ;2 DIGITS
   .BYTE 0 ;SUPPRESS LEADING ZEROS
   TYPE ,CR2LF
   BIT #BIT12,@SWR ;REPORT DESCRIPTION ONLY ?
   BNE 20$ ;BR IF YES
   TYPE ,DH105 ;TYPE "PREVIOUS COMMAND : "

```

6408	030470	104401	001325		TYPE	,\$CRLF	
6409	030474	104401	050714		TYPE	,DH101+10	;TYPE PREV COMMAND HEADER
6410	030500	104401	001325		TYPE	,\$CRLF	
6411	030504	012701	000006		MOV	#6,R1	;SIX COMMAND VALUES
6412	030510	012702	001236		MOV	,\$REG26,R2	;STARTING ADDR OF PREV CMND VALUES
6413	030514	012246		30\$:	MOV	(R2)+,-(SP)	;PUT A WORD ON STACK
6414	030516	104402			TYPOC		;TYPE IT
6415	030520	104401	012117		TYPE	,SPACE2	;TYPE SEPARATORS
6416	030524	005301			DEC	R1	;SEE IF 7 VALUES TYPED YET
6417	030526	001372			BNE	30\$;BR IF NOT
6418	030530	104401	001325		TYPE	,\$CRLF	
6419	030534	104401	012117		TYPE	,SPACE2	;INDENT
6420	030540	104401	050773		TYPE	,DH102	;TYPE HDR FOR BA DATA
6421	030544	104401	001325		TYPE	,\$CRLF	
6422	030550	104401	012117		TYPE	,SPACE2	;INDENT
6423	030554	012246			MOV	(R2)+,-(SP)	
6424	030556	104402			TYPOC		;TYPE PREV. HI BA BITS
6425	030560	104401	012117		TYPE	,SPACE2	
6426	030564	011246			MOV	(R2),-(SP)	
6427	030566	104402			TYPOC		;TYPE PREV. LO BA BITS
6428	030570	104401	012103		TYPE	,CR2LF	
6429	030574	104401			TYPE		;TYPE DATA HEADER
6430	030576	000000		4\$:	.WORD	0	;DH POINTER GOES HERE
6431	030600	104401	001325		TYPE	,\$CRLF	
6432	030604	005005			CLR	R5	;INIT INDENT INDICATOR
6433	030606	032777	010000	150324	5\$:	BIT	#BIT12,@SWR
6434	030614	001056			BNE	20\$;REPORT DESCRIPTION ONLY ?
6435	030616	012001			MOV	(R0)+,R1	;BR IF YES
6436	030620	001454			BEQ	20\$;GET DT POINTER
6437	030622	012000			BEQ	20\$;BRANCH IF THERE ARE NONE
6438	030624	012002			MOV	(R0)+,R0	;GET DF POINTER
6439	030626	112003		10\$:	MOV	(R0)+,R2	;STORE NUMBER OF DH'S
6440	030630	105720			MOVB	(R0)+,R3	;GET & STORE NUMBER OF DATA WORDS
6441	030632	005703			TSTB	(R0)+	;BUMP PAST FORMAT WORD
6442	030634	001417			TST	R3	;TEST IF ANY DATA FOR THIS HEADER
6443	030636	005705			BEQ	14\$;NO - SKIP DATA PRINT
6444	030640	001002			TST	R5	;SEE IF SHOULD INDENT
6445	030642	104401	012117		BNE	11\$;BR IF NOT
6446	030646	013146		11\$:	TYPE	,SPACE2	;INDENT
6447	030650	104402			MOV	@(R1)+,-(SP)	;PUT FIRST DATA WORD ON STACK
6448	030652	005303			TYPOC		;TYPE IT
6449	030654	001403			DEC	R3	;MORE DATA WORDS
6450	030656	104401	012117		BEQ	12\$;NO-BRANCH
6451	030662	000771			TYPE	,SPACE2	;TYPE SEPARATORS
6452	030664	005702		12\$:	BR	11\$;LOOP
6453	030666	001402			TST	R2	;SEE IF <CR>,<LF> NEEDED
6454	030670	104401	001325		BEQ	14\$;BR IF NOT
6455	030674	005302		14\$:	TYPE	,\$CRLF	;TYPE IT
6456	030676	003425			DEC	R2	;MORE DH'S?
6457	030700	012037	030732	15\$:	BLE	20\$;NO-BRANCH
6458	030704	105710			MOV	(R0)+,16\$;GET NEXT DH POINTER
6459	030706	001004			TSTB	(R0)	;SEE IF ANY DATA FOR HDR
6460	030710	104401	001325		BNE	34\$;BR IF YES
6461	030714	005005			TYPE	,\$CRLF	;SKIP EXTRA LINE
6462	030716	000404			CLR	R5	;RE-INIT INDENT INDICATOR
6463	030720	005105		34\$:	BR	36\$	
					COM	R5	;COMPLEMENT INDENT INDICATOR

```

6464 030722 001002          BNE      36$          ;BR IF NO INDENT REQUIRED
6465 030724 104401 012117  TYPE      ,SPACE2    ;INDENT
6466 030730 104401          36$:  TYPE      ;TYPE DH
6467 030732 000000          16$:  .WORD    0        ;DH POINTER GOES HERE
6468 030734 104401 001325  TYPE      ,SCRLF     ;
6469 030740 105710          TSTB    (R0)         ;TYPE A DT?
6470 030742 001331          BNE     10$          ;YES-BRANCH
6471 030744 062700 000002  ADD     #2,R0        ;INCREMENT DF POINTER
6472 030750 000751          BR      14$          ;SEE IF END OF DF BLOCK
6473 030752 104410          20$:  RESREG
6474 030754 000207          RTS     PC
6475                                     ;:*****
6476                                     ;.SBTTL CONTROLLER ERROR REPORTER ROUTINE
6477                                     ;*ENTRY:      JSR PC, CONERR
6478                                     ;*RETURN:     RTS PC
6479                                     ;*
6480                                     ;*THIS ROUTINE DECODES THE CONTROLLER ERROR WORD AND
6481                                     ;*REPORTS THE APPROPRIATE MESSAGE. THE RK611 REGISTERS ARE
6482                                     ;*RETRIEVED FROM THE RK611 AND PLACED IN THE PARAMETER
6483                                     ;*BLOCK. THIS IS DONE BECAUSE PARM 0 MAY NOT BE VALID
6484                                     ;*AT THIS TIME.
6485                                     ;:*****
6486 030756 104407          CONERR: SAVREG      ;SAVE R0-R5
6487 030760 152737 000377 003132  BISB    #377,DONE    ;SET DONE FLAG
6488 030766 105237 003135          INCB    ERRCNT      ;INCREMENT ERROR COUNT
6489 030772 004737 033752          JSR    PC, TOPROC   ;LOAD RK REGS INTO $REGS
6490 030776 032737 000001 003062  BIT     #BIT0,E.CONT ;ERROR 0?
6491 031004 001402          BEQ    1$           ;NO-BRANCH
6492 031006 104064          ERROR  64          ;CLEAR CONT DID NOT CLEAR ERROR
6493 031010 000474          BR     7$           ;
6494 031012 032737 000002 003062  1$:  BIT     #BIT1,E.CONT ;ERROR 1?
6495 031020 001402          BEQ    2$           ;NO-BRANCH
6496 031022 104065          ERROR  65          ;NO ATTENTION IN ATTENTION SUM REG
6497 031024 000466          BR     7$           ;
6498 031026 032737 000004 003062  2$:  BIT     #BIT2,E.CONT ;ERROR 2?
6499 031034 001407          BEQ    3$           ;NO-BRANCH
6500 031036 105737 003141          TSTB  DRNAFG       ;SEE IF DRIVE WAS SIEZED BY OTHER PORT
6501 031042 001402          BEQ    15$          ;BR IF NOT
6502 031044 105237 003142          INCB  REISSU       ;SET FLAG TO RE-ISSUE COMMAND
6503 031050 104066          15$:  ERROR  66          ;UNSOLICITED ATTENTION
6504 031052 000453          BR     7$           ;
6505 031054 032737 000010 003062  3$:  BIT     #BIT3,E.CONT ;ERROR 3?
6506 031062 001402          BEQ    4$           ;NO-BRANCH
6507 031064 104067          ERROR  67          ;UNEXPECTED DATA TYPE ERROR
6508 031066 000445          BR     7$           ;
6509 031070 032737 000020 003062  4$:  BIT     #BIT4,E.CONT ;ERROR 4?
6510 031076 001402          BEQ    5$           ;NO-BRANCH
6511 031100 104070          ERROR  70          ;ATTENTION DID NOT RESET WITH CLEAR
6512 031102 000437          BR     7$           ;
6513 031104 032737 000040 003062  5$:  BIT     #BIT5,E.CONT ;ERROR 5?
6514 031112 001402          BEQ    6$           ;NO-BRANCH
6515 031114 104071          ERROR  71          ;SUBSYS CLEAR DIDN'T CLEAR DRIVE ATTENTION
6516 031116 000431          BR     7$           ;
6517 031120 032737 000400 003062  6$:  BIT     #BIT8,E.CONT
6518 031126 001402          BEQ    8$           ;
6519 031130 104072          ERROR  72          ;DATA LATE WHEN UNLOADING HEADER
  
```

```

6520 031132 000423          BR      7$
6521 031134 032737 001000 003062 8$:  BIT    #BIT9,E.CONT
6522 031142 001402          BEQ    9$
6523 031144 104073          ERROR  73          ;CONTROLLER ERROR DURING DRIVER SERVICE
6524 031146 000415          BR      7$
6525 031150 032737 002000 003062 9$:  BIT    #BIT10,E.CONT
6526 031156 001402          BEQ   10$
6527 031160 104074          ERROR  74          ;DRIVE DETECTED PARITY ERROR
6528 031162 000407          BR      7$
6529 031164 032737 100000 003062 10$: BIT    #BIT15,E.CONT
6530 031172 001402          BEQ   11$
6531 031174 104052          ERROR  52          ;MULTIPLE DRIVE SELECT
6532 031176 000401          BR      7$
6533 031200 104075          11$:  ERROR  75          ;UNDEFINED ERROR
6534 031202 005037 003062  7$:  CLR    E.CONT      ;CLEAR CONTROLLER ERROR WORD
6535 031206 000137 033546          JMP    BGNRTY      ;GO DO RETRY
6536                                     ;*****
6537                                     .SBTTL REPORT SUPPORT ROUTINE
6538                                     ;*THIS ROUTINE LOADS ALL THE PARAMETER BLOCK DATA TO BE REPORTED
6539                                     ;*INTO THE PROPER TEMPORARY REGISTERS FOR REPORTING. ALL THESE MAY
6540                                     ;*NOT BE INCLUDED IN THE REPORT, BUT THEY ARE LOADED ANYWAY.
6541 031212          REPSUP:
6542 031212 104407          SAVREG
6543 031214 005037 005426          CLR    ERRCOM      ;CLEAR ERROR COMMAND STORE
6544 031220 116537 000001 005426          MOVB  P.CMND(R5),ERRCOM ;STORE COMMAND START VALUES
6545 031226 012700 001162          MOV   #SREG0,R0    ;FOR REPORTING
6546 031232 016520 000002          MOV   P.CYLN(R5),(R0)+
6547 031236 116520 000005          MOVB  P.TRCK(R5),(R0)+
6548 031242 105020          CLRB  (R0)+
6549 031244 116520 000004          MOVB  P.SECT(R5),(R0)+
6550 031250 105020          CLRB  (R0)+
6551 031252 016520 000012          MOV   P.WC(R5),(R0)+
6552 031256 012700 001174          MOV   #SREG5,R0
6553 031262 116503 000007          MOVB  P.BAHI(R5),R3
6554 031266 042703 177774          BIC   #177774,R3
6555 031272 010337 001256          MOV   R3,$REG36    ;HI BA BITS
6556 031276 016537 000010 001260          MOV   P.BALO(R5),$REG37 ;LO BA BITS
6557 031304 016520 000016          MOV   P.CS1(R5),(R0)+ ;GET ALL THE VALUES FROM THE
6558 031310 016520 000020          MOV   P.CS2(R5),(R0)+ ;PARAMETER BLOCK AND LOAD
6559 031314 016520 000030          MOV   P.DCYL(R5),(R0)+ ;THE TEMPORARY REGISTERS
6560 031320 016520 000026          MOV   P.DTS(R5),(R0)+ ;FOR REPORTING. ALL THIS
6561 031324 016520 000022          MOV   P.WCR(R5),(R0)+ ;DATA MAY NOT BE VALID
6562                                     ;FOR ALL REPORTS (TO BE
6563 031330 016520 000024          MOV   P.BAR(R5),(R0)+ ;DETERMINED LATER) BUT IT IS
6564 031334 016520 000032          MOV   P.ASOF(R5),(R0)+ ;STORED ANY WAY.
6565 031340 016520 000036          MOV   P.DS(R5),(R0)+
6566 031344 016520 000034          MOV   P.ER(R5),(R0)+
6567 031350 016520 000040          MOV   P.A00(R5),(R0)+
6568 031354 016520 000042          MOV   P.B00(R5),(R0)+
6569 031360 016520 000044          MOV   P.A01(R5),(R0)+
6570 031364 016520 000046          MOV   P.B01(R5),(R0)+
6571 031370 016520 000050          MOV   P.A10(R5),(R0)+
6572 031374 016520 000052          MOV   P.B10(R5),(R0)+
6573 031400 016520 000054          MOV   P.A11(R5),(R0)+
6574 031404 016520 000056          MOV   P.B11(R5),(R0)+
6575                                     ;STORE PREVIOUS COMMAND FOR PRINTOUT
  
```

```

6576 031410 012701 001236      MOV      #$REG26,R1      ;ADRS OF PRINT BUF AREA
6577 031414 012700 005162      MOV      #PRVCM,D,R0     ;ADRS OF PREV CMND STORAGE
6578 031420 112021              MOV      (R0)+,(R1)+     ;DRIVE NO.
6579 031422 105021              CLR      (R1)+
6580 031424 112021              MOV      (R0)+,(R1)+     ;COMMAND
6581 031426 105021              CLR      (R1)+
6582 031430 012021              MOV      (R0)+,(R1)+     ;CYL ADDRESS
6583 031432 116021 000001      MOV      1(R0),(R1)+     ;TRACK
6584 031436 105021              CLR      (R1)+
6585 031440 111021              MOV      (R0),(R1)+      ;SECTOR
6586 031442 105021              CLR      (R1)+
6587 031444 016021 000006      MOV      6(R0),(R1)+     ;WORD COUNT
6588 031450 116003 000003      MOV      3(R0),R3        ;HI BA BITS
6589 031454 042703 177774      BIC      #177774,R3
6590 031460 010321              MOV      R3,(R1)+
6591 031462 016011 000004      MOV      4(R0),(R1)      ;LO BA BITS
6592 031466 104410              RESREG
6593 031470 000207              RTS      PC
6594
6595      ;*****
6596      ;SBTTL REPORT ERROR ROUTINE
6597      ;* ENTRY      JSR      PC,ERRHDL
6598      ;*RETURN      RTS      PC
6599      ;*
6600      ;*THIS ROUTINE IS CALLED BY THE DRIVER WHEN AN ERROR IS DETECTED
6601      ;*IN THE OPERATION. THE ROUTINE DETERMINES WHICH COMMAND WAS
6602      ;*BEING EXECUTED AND GENERATES THE PROPER ERROR MESSAGE.
6603      ;*****
6604 031472 104407      ERRHDL: SAVREG
6605 031474 152737 000377 003132      BISB    #377,DONE        ;SET DONE FLAG
6606 031502 105237 003135              INCB    ERRCNT          ;INCREMENT ERROR COUNT
6607 031506 005037 005424              CLR     RECODE          ;CLEAR RECOVERY CODE WORD
6608 031512 032737 000400 005424      ER2ENT: BIT    #LEV2ER,RECODE ;TEST IF 2ND LEVEL ERROR
6609 031520 001402              BEQ     52$             ;NO - SKIP PARAM BLOCK CHANGE
6610 031522 012705 002724              MOV     #PARM1,R5       ;ELSE SET R5 TO PARAMETER BLOCK 1
6611 031526 012737 033724 003056      52$:  MOV     #RETANL,A.ABNL ;SET NOW ABNORMAL AND NORMAL RETURN FOR
6612 031534 012737 033742 003054      MOV     #RETNML,A.NORM  ;DRIVER OPERATIONS IN ERROR PROCESSING
6613 031542 004737 031212              JSR     PC,REPSUP       ;GO SET UP REGISTERS FOR REPORT
6614
6615      ;NOW BEGIN TESTING THE ERROR
6616      ;BITS. THE SEQUENCE IN
6617      ;WHICH THEY ARE TESTED IS
6618      ;CONSIDERED SIGNIFICANT IN
6619      ;THAT ERRORS OF A MORE
6620      ;CATASTROPHIC NATURE ARE FIRST
6621      ;TESTED.
6622
6623      ;IF AN ERROR IS FOUND SET,
6624      ;THAT ERROR IS REPORTED AND
6625      ;THE REPORTING IS TERMINATED.
6626      ;IF ADDITIONAL ERRORS ARE SET,
6627      ;THE RK611 REGISTER PRINTOUTS
6628      ;WILL SHOW THIS BUT THE
6629      ;REGISTER CONTENTS MUST BE
6630      ;MANUALLY DECODED TO LOCATE THE
6631 031546 016504 000034      MOV     P.ER(R5),R4    ;SET R4 TO ERROR REGISTER
  
```

6632	031552	032765	020000	000020		BIT	#UPE,P.CS2(R5)	:TEST UPE. IF UES-SET
6633	031560	001406				BEQ	1\$:REPORT ERROR
6634	031562	104001				ERROR	1	
6635	031564	052737	000200	005424		BIS	#ABORT,RECODE	
6636	031572	000137	033224			JMP	37\$	
6637	031576	032765	004000	000020	1\$:	BIT	#NEM,P.CS2(R5)	:TEST NON-EXISTANT MEMORY
6638	031604	001406				BEQ	2\$	
6639	031606	104002				ERROR	2	
6640	031610	052737	000200	005424		BIS	#ABORT,RECODE	
6641	031616	000137	033224			JMP	37\$	
6642	031622	032765	010000	000020	2\$:	BIT	#NED,P.CS2(R5)	:TEST NON-EXISTANT DRIVE
6643	031630	001412				BEQ	3\$	
6644	031632	032765	000400	000020		BIT	#UFE,P.CS2(R5)	:TEST IF NED & UFE BOTH SET
6645	031640	001403				BEQ	38\$	
6646	031642	104035				ERROR	35	:NED & UFE BOTH SET ERROR
6647	031644	000137	033224			JMP	37\$	
6648	031650	104003			38\$:	ERROR	3	:NED ONLY
6649	031652	000137	033224			JMP	37\$	
6650	031656	032765	000400	000020	3\$:	BIT	#UFE,P.CS2(R5)	:TEST UNIT FIELD ERROR
6651	031664	001412				BEQ	4\$	
6652	031666	032765	010000	000020		BIT	#NED,P.CS2(R5)	:TEST IF UFE & NED BOTH SET
6653	031674	001403				BEQ	39\$:NO-SKIP
6654	031676	104035				ERROR	35	:REPORT NED & UFE BOTH SET
6655	031700	000137	033224			JMP	37\$	
6656	031704	104004			39\$:	ERROR	4	:REPORT UFE ONLY
6657	031706	000137	033224			JMP	37\$	
6658	031712	032765	000100	000014	4\$:	BIT	#CMDTO,P.PRST(R5)	:
6659	031720	001423				BEQ	5\$	
6660	031722	004737	033752			JSR	PC, TOPROC	:GO PROCESS TIMEOUT
6661	031726	032737	002000	005424		BIT	#TWOTOS,RECODE	:2ND TIMEOUT IN TIMEOUT PROC?
6662	031734	001007				BNE	40\$:YES - SKIP TO ERROR 56
6663	031736	032737	000400	005424		BIT	#LEV2ER,RECODE	:TEST IF LEVEL 2 ERROR
6664	031744	001006				BNE	41\$:YES - SKIP TO ERROR 57
6665	031746	104005				ERROR	5	:ELSE MAKE FULL TIMEOUT REPORT
6666	031750	000137	033224			JMP	37\$	
6667	031754	104056			40\$:	ERROR	56	:TWO TIMEOUTS ERROR REPORT
6668	031756	000137	033224			JMP	37\$	
6669	031762	104057			41\$:	ERROR	57	:2ND LEVEL ERROR REPORT
6670	031764	000137	031512			JMP	ERZENT	:GO BUILD AND MAKE 2ND REPORT
6671	031770	032765	010000	000014	5\$:	BIT	#DRVSZD,P.PRST(R5)	:SEE IF DRIVE SIEZED BY OTHER PORT
6672	031776	001403				BEQ	65\$:BR IF DRIVE IS AVAILABLE
6673	032000	104107				ERROR	107	:DRIVE SIEZED BY OTHER PORT
6674	032002	000137	033224			JMP	37\$	
6675	032006	032765	020000	000016	65\$:	BIT	#SPAR,P.CS1(R5)	:TEST D TO C PARITY ERROR
6676	032014	001406				BEQ	6\$	
6677	032016	104006				ERROR	6	
6678	032020	052737	004000	005424		BIS	#RCLREQ,RECODE	
6679	032026	000137	033224			JMP	37\$	
6680	032032	032704	000010		6\$:	BIT	#DRPAR,R4	:TEST DRIVE DETECTED PARITY ERROR
6681	032036	001406				BEQ	7\$	
6682	032040	104007				ERROR	7	
6683	032042	052737	004000	005424		BIS	#RCLREQ,RECODE	
6684	032050	000137	033224			JMP	37\$	
6685	032054	032765	000010	000036	7\$:	BIT	#ACLO,P.DS(R5)	:TEST AC LOW
6686	032062	001403				BEQ	8\$	
6687	032064	104010				ERROR	10	

6688	032066	000137	033224			JMP	37\$	
6689	032072	032765	000020	000036	8\$:	BIT	#SPDLSS,P.DS(R5)	;TEST SPEED LOSS
6690	032100	001403				BEQ	24\$	
6691	032102	104011				ERROR	11	
6692	032104	000137	033224			JMP	37\$	
6693	032110	032765	004000	000016	24\$:	BIT	#CTO,P.CS1(R5)	;TEST FOR CONTROLLER TIMEOUT
6694	032116	001401				BEQ	25\$	
6695	032120	104027				ERROR	27	
6696	032122	032704	000001		25\$:	BIT	#ILC,R4	;TEST ILLEGAL FUNCTION CODE
6697	032126	001403				BEQ	10\$	
6698	032130	104012				ERROR	12	
6699	032132	000137	033224			JMP	37\$	
6700	032136	032765	002000	000020	10\$:	BIT	#PGE,P.CS2(R5)	;TEST PROGRAMMING ERROR
6701	032144	001403				BEQ	11\$	
6702	032146	104013				ERROR	13	
6703	032150	000137	033224			JMP	37\$	
6704	032154	032704	000004		11\$:	BIT	#ILF,R4	;TEST ILLEGAL DRIVE FUNCTION
6705	032160	001403				BEQ	12\$	
6706	032162	104014				ERROR	14	
6707	032164	000137	033224			JMP	37\$	
6708	032170	032704	000040		12\$:	BIT	#DTYE,R4	;TEST DRIVE TYPE ERROR
6709	032174	001403				BEQ	13\$	
6710	032176	104015				ERROR	15	
6711	032200	000137	033224			JMP	37\$	
6712	032204	032704	000020		13\$:	BIT	#FMTE,R4	;TEST FORMAT ERROR
6713	032210	001403				BEQ	14\$	
6714	032212	104016				ERROR	16	
6715	032214	000137	033224			JMP	37\$	
6716	032220	032704	004000		14\$:	BIT	#WLE,R4	;TEST WRITE LOCK ERROR
6717	032224	001403				BEQ	15\$	
6718	032226	104017				ERROR	17	
6719	032230	000137	033224			JMP	37\$	
6720	032234	032704	040000		15\$:	BIT	#UNS,R4	;TEST DRIVE UNSAFE
6721	032240	001406				BEQ	16\$	
6722	032242	104020				ERROR	20	
6723	032244	052737	000200	005424		BIS	#ABORT,RECODE	
6724	032252	000137	033324			JMP	ALLTRM	
6725	032256	032704	000002		16\$:	BIT	#SKI,R4	;TEST SEEK INCOMPLETE
6726	032262	001406				BEQ	17\$	
6727	032264	104021				ERROR	21	
6728	032266	052737	004000	005424		BIS	#RCLREQ,RECODE	
6729	032274	000137	033224			JMP	37\$	
6730	032300	032704	001000		17\$:	BIT	#COE,R4	;TEST CYLINDER OVERFLOW
6731	032304	001406				BEQ	18\$	
6732	032306	104022				ERROR	22	
6733	032310	052737	004000	005424		BIS	#RCLREQ,RECODE	
6734	032316	000137	033224			JMP	37\$	
6735	032322	032704	002000		18\$:	BIT	#IDAE,R4	;TEST ILLEGAL CYLINDER
6736	032326	001406				BEQ	19\$	
6737	032330	104023				ERROR	23	
6738	032332	052737	004000	005424		BIS	#RCLREQ,RECODE	
6739	032340	000137	033224			JMP	37\$	
6740	032344	032765	000040	000036	19\$:	BIT	#DROT,P.DS(R5)	;TEST DRIVE OFF TRACK
6741	032352	001406				BEQ	20\$	
6742	032354	104024				ERROR	24	
6743	032356	052737	004000	005424		BIS	#RCLREQ,RECODE	

6744	032364	000137	033224			JMP	37\$	
6745	032370	032704	010000	20\$:		BIT	#DTE,R4	;TEST DRIVE TIMING ERROR
6746	032374	001406				BEQ	21\$	
6747	032376	104025				ERROR	25	
6748	032400	052737	000200	005424		BIS	#ABORT,RECODE	
6749	032406	000137	033224			JMP	37\$	
6750	032412	032765	100000	000020	21\$:	BIT	#DLT,P.CS2(R5)	;TEST DATA LATE
6751	032420	001403				BEQ	22\$	
6752	032422	104026				ERROR	26	
6753	032424	000137	033224			JMP	37\$	
6754	032430	032704	020000	22\$:		BIT	#OPI,R4	;TEST IF OPI ERROR
6755	032434	001470				BEQ	29\$	
6756	032436	052737	000010	005424		BIS	#OPIERR,RECODE	
6757	032444	105737	003130			TSTB	DERCNT	;TEST IF FIRST ERROR
6758	032450	001402				BEQ	50\$	
6759	032452	000137	033224			JMP	37\$;NO - SKIP REPORT
6760	032456	032737	000400	005424	50\$:	BIT	#LEV2ER,RECODE	;TEST IF A SECOND LEVEL 2 ERROR
6761	032464	001403				BEQ	26\$;HAS ALREADY OCCURRED
6762	032466	104054			27\$:	ERROR	54	
6763	032470	000137	033224			JMP	37\$;GET OUT OF ERROR REPORT
6764	032474	004737	034432	26\$:		JSR	PC,BLDEXH	;GO BUILD EXPECTED HEADER
6765	032500	004737	034122			JSR	PC,RDHDO	;GET HEADER OF SECTOR 0
6766	032504	032737	000400	005424		BIT	#LEV2ER,RECODE	;TEST IF ERROR GETTING HDR
6767	032512	001036				BNE	28\$;IF YES-GO MAKE ABBREVIATED REPORT
6768	032514	013702	003046			MOV	RKBAS,R2	;STORE HEADER 0 INTO REGISTERS
6769	032520	042762	000100	000000		BIC	#IE,RKCS1(R2)	;RESET INTERRUPT ENABLE
6770	032526	016237	000024	001202		MOV	RKDB(R2),\$REG10	;FOR REPORTING
6771	032534	016237	000024	001204		MOV	RKDB(R2),\$REG11	
6772	032542	016237	000024	001206		MOV	RKDB(R2),\$REG12	
6773	032550	032762	100000	000000		BIT	#CERR,RKCS1(R2)	;TEST IF ERROR DURING STORAGE
6774	032556	001343				BNE	27\$	
6775	032560	105737	003143			TSTB	TSTTYP	;SEE IF READING WITH OFFSET
6776	032564	001403				BEQ	54\$;BR IF NOT
6777	032566	105037	003135			CLRB	ERRCNT	;CLEAR ERROR COUNT
6778	032572	000401				BR	55\$;CONTINUE
6779	032574	104030			54\$:	ERROR	30	;MAKE OPI REPORT
6780	032576	052737	004000	005424	55\$:	BIS	#RCLREQ,RECODE	;SET RECALIBRATE REQUEST BIT
6781	032604	000137	033224			JMP	37\$	
6782	032610	104054			28\$:	ERROR	54	
6783	032612	000137	031512			JMP	ER2ENT	;GO MAKE 2ND LEVEL REPORT
6784	032616	032704	000400	29\$:		BIT	#HVRC,R4	;TEST IF HVRC ERROR
6785	032622	001457				BEQ	23\$	
6786	032624	052737	000004	005424		BIS	#HVRCER,RECODE	
6787	032632	105737	003130			TSTB	DERCNT	;TEST IF FIRST ERROR
6788	032636	001402				BEQ	30\$;YES - REPORT
6789	032640	000137	033224			JMP	37\$;JUMP TO RETURN
6790	032644	032737	000400	005424	30\$:	BIT	#LEV2ER,RECODE	;TEST IF A 2ND LEVEL ERROR HAS ALREADY
6791	032652	001403				BEQ	31\$;OCCURRED. NO-SKIP EXIT
6792	032654	104055			51\$:	ERROR	55	
6793	032656	000137	033224			JMP	37\$;GET OUT OF ERROR REPORT
6794	032662	004737	034432	31\$:		JSR	PC,BLDEXH	;GO BUILD EXPECTED HEADER
6795	032666	004737	034122			JSR	PC,RDHDO	;GO GET HDR 0
6796	032672	032737	000400	005424		BIT	#LEV2ER,RECODE	;TEST IF ERROR IN GETTING HDR
6797	032700	001025				BNE	32\$;IF YES-GO MAKE ABBREVIATED REPORT
6798	032702	013702	003046			MOV	RKBAS,R2	;GET RK611 BASE ADDRESS
6799	032706	042762	000100	000000		BIC	#IE,RKCS1(R2)	;CLEAR INTERRUPT ENABLE

```

6800 032714 016237 000024 001202      MOV      RKDB(R2), $REG10 ;STORE OFF HEADER
6801 032722 016237 000024 001204      MOV      RKDB(R2), $REG11
6802 032730 016237 000024 001206      MOV      RKDB(R2), $REG12
6803 032736 032762 100000 000000      BIT      #CERR, RKCS1(R2) ;TEST IN ANY ERROR IN UNLOAD
6804 032744 001343          BNE      51$             ;IY YES - GO MAKE SHORT REPORT
6805 032746 104031          ERROR   31             ;MAKE FULL REPORT
6806 032750 000137 033224          JMP      37$
6807 032754 104055          32$:    ERROR   55             ;ABREVIATED HVRC ERROR RPORT
6808 032756 000137 031512          JMP      ER2ENT         ;GO REPORT 2ND LEVEL ERROR
6809 032762 032704 000200          23$:    BIT      #BSE, R4         ;TEST FOR BAD SECTOR ERROR
6810 032766 001436          BEQ      33$           ;NO - SKIP
6811 032770 052737 000002 005424          BIS      #BSERR, RECODE ;SET ERROR FLAG
6812 032776 016500 000030          MOV      P.DCYL(R5), R0 ;GET CYL NO.
6813 033002 116501 000027          MOV      P.DTS+1(R5), R1 ;GET TRACK NO.
6814 033006 042701 177774          BIC      #177774, R1    ;CLEAR ALL BITS EXCEPT TRACK
6815 033012 016502 000026          MOV      P.DTS(R5), R2  ;GET SECTOR IN ERROR
6816 033016 042702 177740          BIC      #177740, R2    ;CLEAR ALL BITS EXCEPT SECTOR
6817 033022 004737 030020          JSR      PC, BDSRCK     ;GO SEE IF THIS SECTOR LISTED
6818 033026 032737 001000 005424          BIT      #BADSEC, RECODE ;TEST RESULT
6819 033034 001073          BNE      37$           ;YES - EXIT, NO ERROR OR REPORT
6820 033036 105737 003143          TSTB    TSTTYP         ;SEE IF READING WITH OFFSET
6821 033042 001403          BEQ      56$           ;BR IF NOT
6822 033044 105037 003135          CLRB    ERRCNT         ;CLEAR ERROR COUNT
6823 033050 000465          BR      37$           ;CONTINUE
6824 033052 104104          56$:    ERROR   104         ;ELSE REPORT BAD BSE
6825 033054 042737 000002 005424          BIC      #BSERR, RECODE ;RESET BSE ERROR FLAG
6826 033062 000460          BR      37$           ;GO EXIT
6827 033064 032704 100000          33$:    BIT      #DCK, R4         ;TEST IF DATA CHECK
6828 033070 001435          BEQ      36$
6829 033072 052737 000020 005424          BIS      #DCKERR, RECODE ;SET DATA CHECK ERROR IN RECOVERY CODE
6830 033100 032704 000100          BIT      #ECH, R4         ;TEST IF ECC IS HARD. IF
6831 033104 001406          BEQ      34$           ;YES SET UNCORRECTABLE IN
6832 033106 052737 000040 005424          BIS      #ECCNC, RECODE ;RECOVERY FLAG AND A 0 IN
6833 033114 005037 001206          CLR      $REG12         ;REG12 TO INDICATE UNCORRECTABLE,
6834 033120 000403          BR      35$           ;A 1 IN REG 12 FOR CORRECTABLE
6835 033122 012737 000001 001206          34$:    MOV      #1, $REG12
6836 033130 105737 003130          35$:    TSTB    DERCNT         ;TEST IF FIRST ERROR
6837 033134 001033          BNE      37$           ;NO SKIP REPORT
6838 033136 004737 034276          JSR      PC, GTPKAD     ;GO GET PACK ADDRESS OF ERROR
6839 033142 016537 000060 001202          MOV      P.EPOS(R5), $REG10 ;STORE ECC POSITION &
6840 033150 016537 000062 001204          MOV      P.EPAT(R5), $REG11 ;PATTERN
6841 033156 104032          ERROR   32             ;REPORT DCK ERROR
6842 033160 000137 033224          JMP      37$
6843 033164 032765 040000 000020          36$:    BIT      #WCE, P.CS2(R5) ;TEST WRITE CHECK ERROR
6844 033172 001414          BEQ      37$
6845 033174 042737 000200 005424          BIC      #ABORT, RECODE ;CLEAR ABORT & SET WRITE
6846 033202 052737 000100 005424          BIS      #WCERR, RECODE ;CHECK ERROR IN RECODE
6847 033210 105737 003130          TSTB    DERCNT         ;TEST IF FIRST ERROR
6848 033214 001003          BNE      37$           ;NO - SKIP
6849 033216 004737 034276          JSR      PC, GTPKAD     ;GO GET ADDRESS OF ERROR
6850 033222 104033          ERROR   33             ;REPORT WCE
6851
6852 033224 032765 000020 000014          37$:    BIT      #DRVHRD, P.PRST(R5) ;TEST HARD ERROR
6853 033232 001404          BEQ      43$
6854 033234 104036          ERROR   36
6855 033236 052737 000200 005424          BIS      #ABORT, RECODE

```

```

6856
6857 033244 032765 000040 000014 43$: BIT #DRVDSCL,P.PRST(R5) ;TEST STATUS CHANGE NOT CLEARED
6858 033252 001404 BEQ 44$
6859 033254 104037 ERROR 37
6860 033256 052737 000200 005424 BIS #ABORT,RECODE
6861
6862 033264 032765 004000 000014 44$: BIT #NODSC,P.PRST(R5) ;IFST ATTENTION BUT NO FAULT OR DSC
6863 033272 001404 BEQ 46$
6864 033274 104040 ERROR 40
6865 033276 052737 000200 005424 BIS #ABORT,RECODE
6866
6867 033304 032765 000010 000014 46$: BIT #UEXATT,P.PRST(R5) ;TEST UNEXPECTED ATTENTION
6868 033312 001404 BEQ ALLTRM
6869 033314 104042 ERROR 42
6870 033316 052737 000200 005424 BIS #ABORT,RECODE
6871
6872
6873 ;ALL ERRORS MUST EXIT THROUGH THIS POINT
6874
6875 033324 012705 002640 ALLTRM: MOV #PARMO,R5 ;RESTORE PARAMETER BLOCK SELECTION
6876 033330 012737 020526 003056 MOV #DCKHDL,A.ABNL ;SET READ ERROR HANDLER
6877 033336 105737 003143 TSTB TSTTYP ;SEE IF READING WITH OFFSETS
6878 033342 001003 BNE 20$ ;BR IF YES
6879 033344 012737 031472 003056 MOV #ERRHDL,A.ABNL ;RESTORE INTERRUPT VECTORS FOR RETRY
6880 033352 012737 030250 003054 20$: MOV #ERRFRE,A.NORM ;DRIVER OPERATIONS, IF ANY
6881 033360 032737 000200 005424 BIT #ABORT,RECODE ;IF ABORT IS NOT SET AND
6882 033366 001057 BNE 48$ ;THE DRIVE IS READY (HAS NOT
6883 ;CYCLED DOWN)
6884 033370 013702 003046 MOV RKBAS,R2 ;GET BASE ADDRESS
6885 033374 032762 000200 000012 BIT #RDY,RKDS(R2) ;TEST IF DRIVE READY SET
6886 033402 001004 BNE 47$ ;RECALIBRATE REQUIRED BIT IS SET
6887 033404 052737 000200 005424 BIS #ABORT,RECODE ;ELSE ABORT WITH ABORT MESSAGE
6888 033412 000445 BR 48$
6889 033414 032737 004000 005424 47$: BIT #RCLREQ,RECODE ;IF RECALIBRATE IS REQUIRED
6890 033422 001451 BEQ BGNRTY ;FOR RETRY SET UP PARAM
6891 033424 013737 002640 002724 MOV PARMO,PARM1 ;SAVE COMMAND
6892 033432 112765 000113 000001 MOV #RECAL,P.CMND(R5) ;BLOCK TO DO IT.
6893 033440 012737 033724 003056 MOV #RETANL,A.ABNL
6894 033446 004737 030124 JSR PC,DRVCAL
6895 033452 013737 002724 002640 MOV PARM1,PARMO ;RESTORE COMMAND
6896 033460 012737 020526 003056 MOV #DCKHDL,A.ABNL ;SET READ ERROR HANDLER
6897 033466 105737 003143 TSTB TSTTYP ;SEE IF READING WITH OFFSETS
6898 033472 001003 BNE 50$ ;BR IF YES
6899 033474 012737 031472 003056 MOV #ERRHDL,A.ABNL ;RESTORE ERROR RETURN
6900 033502 032737 000400 005424 50$: BIT #LEV2ER,RECODE ;IF AN ERROR OCCURRED IN THE
6901 033510 001416 BEQ BGNRTY ;RECAL ATTEMP SET ABORT,
6902 033512 052737 000200 005424 BIS #ABORT,RECODE ;PRINT THE RECAL ERROR MESSAGE, AND
6903 033520 104060 ERROR 60 ;GO REPORT DETAILS
6904 033522 000137 031512 JMP ER2ENT
6905 033526 104061 48$: ERROR 61 ;REPORT ABORT-RETRY FAILED
6906
6907 ;THE PROGRAM WILL HALT HERE IF THE ABORT FLAG HAS BEEN SET OR
6908 ;IF THE DRIVE READY BIT IS RESET.
6909
6910 033530 000005 HLTPRG: RESET ;DISABLE ALL DEVICES
6911 033532 000000 HALT ;HALT THE CPU
  
```

```

6912 033534 105037 003135          CLRB  ERRCNT          ;CLEAR ERROR COUNT
6913 033540 000005          RESET          ;RESET ALL DEVICES
6914 033542 000137 012206          JMP    CMSTRT
6915
6916
6917
6918
6919
6920
6921
6922
6923 033546 032737 000136 005424 BGNRTY: BIT    #BSERR!HVR CER!OPIERR!DCKERR!WCERR,RECODE ;TEST IF ANY DATA ERROR
6924 033554 001404          BEQ    3$
6925 033556 052737 100000 005424 9$:  BIS    #ANYDER,RECODE
6926 033564 000453          BR     2$
6927 033566          3$:
6928 033566 105737 003150          TSTB   NORTRY          ;SEE IF 'NO-RETRY' FLAG SET
6929 033572 001371          BNE    9$              ;BR IF YES
6930 033574 032737 001000 005424 BIT    #BADSEC,RECODE ;TEST IF BAD SECTOR FLAG SET
6931 033602 001044          BNE    2$              ;IF YES-EXIT TO CALLER
6932 033604 123737 003135 003136 CMPB   ERRCNT,ERRLMT ;BEGIN RETRY IF ERROR COUNT HAS
6933 033612 001012          BNE    1$              ;NOT BEEN EXCEEDED
6934 033614 005037 001174          CLR    $REG5
6935 033620 113737 003135 001174 MOVB   ERRCNT,$REG5   ;GET ERROR RETRY COUNT
6936 033626 104102          ERROR 102             ;REPORT RETRY UNSUCCESSFUL
6937 033630 052737 000200 005424 BIS    #ABORT,RECODE  ;SET ABORT & QUIT
6938 033636 000734          BR     HLTPRG
6939 033640 013702 003046          1$:  MOV    RKBAS,R2      ;GET RK BASE ADDRESS
6940 033644 112765 000177 000001 MOVB   #SUBCLR,P.CMND(R5) ;SET UP TO CLEAR SUBSYSTEM
6941 033652 004737 030124          JSR    PC,DRVCAL      ;GO DO IT
6942 033656 012700 005176          MOV    #COMSTR,R0     ;GO AND REESTABLISH THE COMMAND
6943 033662 012025          MOV    (R0)+,(R5)+    ;AS IT WAS ENTERED INTO THE
6944 033664 012025          MOV    (R0)+,(R5)+    ;PARAMETER BLOCK
6945 033666 012025          MOV    (R0)+,(R5)+
6946 033670 012025          MOV    (R0)+,(R5)+
6947 033672 012025          MOV    (R0)+,(R5)+
6948 033674 012025          MOV    (R0)+,(R5)+
6949 033676 012705 002640          MOV    #PARMO,R5
6950 033702 012737 000000 177776 MOV    #PRO,#PSW      ;LOWER PRIORITY TO ALLOW INTERRUPT
6951 033710 004737 030124          JSR    PC,DRVCAL      ;CALL DRIVER
6952 033714 105037 003135          2$:  CLRB   ERRCNT        ;CLEAR ERROR COUNT
6953 033720 104410          RESREG
6954 033722 000207          RTS    PC              ;IF RETURN GETS HERE, NO ERROR
6955
6956
6957 033724 152737 000377 003132 RETANL: BISB   #377,DONE ;SET DONE
6958 033732 052737 000400 005424 BIS    #LEV2ER,RECODE ;SET LEVEL TWO ERROR
6959 033740 000207          RTS    PC
6960 033742 152737 000377 003132 RETNML: BISB   #377,DONE ;SET DONE
6961 033750 000207          RTS    PC

```

```

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

```

6962
6963
6964
6965
6966
6967

```

6968
6969 033752
6970 033752 104407
6971 033754 013702 003046
6972 033760 012701 001174
6973 033764 016221 000000
6974 033770 016221 000010
6975 033774 016221 000020
6976 034000 016221 000006
6977 034004 016221 000002
6978 034010 016221 000004
6979 034014 016221 000016
6980 034020 016221 000012
6981 034024 016221 000014
6982 034030 005000
6983
6984
6985 034032 012705 002724
6986 034036 112765 000141 000001
6987 034044 004737 030124
6988 034050 032765 000100 000014
6989 034056 001403
6990 034060 052737 002000 005424
6991 034066 062705 000040
6992 034072 012521
6993 034074 012521
6994 034076 012521
6995 034100 012521
6996 034102 012521
6997 034104 012521
6998 034106 012521
6999 034110 012521
7000
7001 034112 012705 002640
7002 034116 104410
7003 034120 000207
7004
7005
7006
7007
7008
7009
7010 034122
7011 034122 104407
7012 034124 016501 000026
7013 034130 016500 000052
7014 034134 042700 160017
7015 034140 006200
7016 034142 006200
7017 034144 006200
7018 034146 006200
7019 034150 012705 002724
7020 034154 010165 000004
7021 034160 010065 000002
7022 034164 112765 000177 000001
7023 034172 012737 000000 177776
    
```

```

*****
TOPROC:
    SAVREG
    MOV    RKBAS,R2
    MOV    #SREG5,R1      ;SET UP R1 FOR RK REGISTER STORAGE
    MOV    RKCS1(R2),(R1)+ ;STORE ALL VALID RK611
    MOV    RKCS2(R2),(R1)+ ;REGISTERS
    MOV    RKDC(R2),(R1)+
    MOV    RKDA(R2),(R1)+
    MOV    RKWC(R2),(R1)+
    MOV    RKBA(R2),(R1)+
    MOV    RKASOF(R2),(R1)+
    MOV    RKDS(R2),(R1)+
    MOV    RKER(R2),(R1)+
    CLR    R0              ;THIS CODE WILL ATTEMPT TO
                           ;RETRIEVE THE STATUS FROM THE
                           ;DRIVE.
    MOV    #PARM1,R5      ;SET UP TO USE PARM1
    MOVB   #RDSTAT,P.CMND(R5) ;DO READ DRIVE STATUS CUMMAND
    JSR    PC,DRVCAL      ;CALL DRIVER
    BIT    #CMDTO,P.PRST(R5) ;TEST FOR TIMEOUT
    BEQ    1$             ;NO - SKIP
    BIS    #TWOTOS,RECODE ;SET TWO TIMEOUTS FLAG
    1$:    ADD    #P.A00,R5 ;BUMP R5 TO POINT TO DRIVE STATUS
    MOV    (R5)+,(R1)+    ;MOVE ALL THE DRIVE STATUS INTO THE
    MOV    (R5)+,(R1)+    ;TEMP REGS FOR REPORTING.
    MOV    (R5)+,(R1)+
    MOV    (R5)+,(R1)+
    MOV    (R5)+,(R1)+
    MOV    (R5)+,(R1)+
    MOV    (R5)+,(R1)+
    MOV    (R5)+,(R1)+
    MOV    (R5)+,(R1)+
    MOV    #PARM0,R5      ;RESTORE PARM 0
    RESREG
    RTS    PC
    
```

```

*****
.SBTTL READ HEADER 0 ROUTINE
*****
RDHDO:
    SAVREG
    MOV    P.DTS(R5),R1    ;STORE TRACK AND SECTOR
    MOV    P.B10(R5),R0    ;GET THE CYLINDER ADRS
    BIC    #160017,R0      ;FROM THE DRIVE STATUS. CLEAR
    ASR    R0              ;OFF UNUSED BITS AND POSITION
    ASR    R0              ;FOR USE AS THE DESIRED
    ASR    R0              ;CYLINDER IN THE READ
    ASR    R0              ;HEADER COMMAND.
    MOV    #PARM1,R5      ;SET UP TO USE P.B. 1
    MOV    R1,P.SECT(R5)  ;INSERT TRK,SECT FOR READ HDR
    MOV    R0,P.CYLN(R5)  ;SET CYL NO.
    MOVB   #SUBCLR,P.CMND(R5) ;SET S.S. CLEAR CMND
    MOV    #PRO,2#PSW     ;ALLOW INTERRUPTS
    
```

```

7024 034200 004737 030124 JSR PC,DRVCAL ;INIT THE S.S.
7025 034204 112765 000125 000001 MOVB #RDHEAD,P.CMND(R5) ;SET READ HEADER COMMAND
7026 034212 004737 030124 JSR PC,DRVCAL ;DO A READ HEADER
7027 034216 013762 024364 000000 MOV HOLD1,RKCS1(R2) ;DO A READ HDR, WITH FMT BIT = 0
7028 034224 032762 000200 000000 24$: BIT #RDY,RKCS1(R2) ;WAIT FOR READY
7029 034232 001774 BEQ 24$
7030 034234 013762 024366 000000 MOV HOLD2,RKCS1(R2) ;DO A READ HDR, WITH FMT BIT = 1
7031 034242 032762 000200 000000 26$: BIT #RDY,RKCS1(R2) ;WAIT FOR READY
7032 034250 001774 BEQ 26$
7033 034252 142765 000020 000007 BICB #B.CFMT,P.CS1H(R5) ;CLEAR THE FORMAT BIT
7034 034260 153765 003134 000007 BISB FORMAT,P.CS1H(R5) ;RESTORE TYPE AND FMT IN USE
7035 034266 012705 002640 MOV #PARMO,R5 ;RESTORE P.B. 0 ADDRESS
7036 034272 104410 RESREG ;RESTORE R0-R5
7037 034274 000207 RTS PC ;RETURN
  
```

7038
7039
7040
7041

 .SBTTL GET PACK ADDRESS ROUTINE

```

7042  
7043  
7044 034276 016537 000030 001174 GTPKAD: MOV P.DCYL(R5),$REG5 ;GET CYLINDER NUMBER
7045 034304 005037 001176 CLR $REG6 ;CLEAR REGISTERS FOR
7046 034310 005037 001200 CLR $REG7 ;TRACK & SECTOR STORAGE
7047 034314 116537 000026 001200 MOVB P.DTS(R5),$REG7 ;STORE THE TRACK AND SECTOR
7048 034322 116537 000027 001176 MOVB P.DTS+1(R5),$REG6
7049 034330 032737 000004 005424 BIT #HVR CER,RECODE ;SEE IF HVRC ERROR
7050 034336 001034 BNE 5$ ;BR IF YES
7051 034340 005737 001200 TST $REG7 ;ADJUST THE ADDRESS CONTAINED IN
7052 ;THE RK REGISTERS FOR THE AUTOMATIC
7053 034344 001403 BEQ 1$ ;INCREMENT
7054 034346 005337 001200 DEC $REG7
7055 034352 000426 BR 5$
7056 034354 032765 010000 000016 1$: BIT #CFMT,P.CS1(R5)
7057 034362 001404 BEQ 2$
7058 034364 012737 000023 001200 MOV #19.,$REG7
7059 034372 000403 BR 3$
7060 034374 012737 000025 001200 2$: MOV #21.,$REG7
7061 034402 005737 001176 3$: TST $REG6
7062 034406 001403 BEQ 4$
7063 034410 005337 001176 DEC $REG6
7064 034414 000405 BR 5$
7065 034416 012737 000002 001176 4$: MOV #2,$REG6
7066 034424 005337 001174 DEC $REG5
7067 034430 000207 5$: RTS PC
  
```

7068
7069
7070
7071

 .SBTTL BUILD EXPECTED HEADER
 ;*USES DESIRED CYLINDER, TRACK AND SECTOR REGISTERS TO DETERMINE
 ;*WHICH HEADER WAS EXPECTED. LOADS EXPECTED VALUES IN \$REG5, 6, AND
 ;*7 FOR REPORTING.

```

7072  
7073  
7074  
7075  
7076  
7077 034432 104407 BLDEXH: SAVREG
7078 034434 016537 000030 001174 MOV P.DCYL(R5),$REG5 ;CONSTRUCT EXPECTED HDR
7079 034442 016501 000026 MOV P.DTS(R5),R1 ;DESIRED CYLINDER & DESIRED TRACK
  
```

7080	034446	042701	174377		BIC	#174377,R1		:CLEAR ALL BUT TRACK BITS
7081	034452	006201			ASR	R1		:AND SECTOR. SHIFT THE TRACK
7082	034454	0062C1			ASR	R1		:OVER TO CONFORM TO HEADER FORMAT
7083	034456	006201			ASR	R1		:CHECK THE FORMAT BIT AND
7084								:IF SET, SET THE HEADER FORMAT
7085	034460	016537	000026	001176	MOV	P.DTS(R5), \$REG6		:BIT.
7086	034466	042737	177740	001176	BIC	#177740, \$REG6		:CLEAR ALL BUT SECTOR
7087	034474	060137	001176		ADD	R1, \$REG6		:ADD TRACK AND SECTOR TOGETHER
7088	034500	052737	140000	001176	BIS	#140000, \$REG6		:INSERT BSE BITS
7089	034506	032765	010000	000016	BIT	#CFMT, P.CS1(R5)		
7090	034514	001403			BEQ	23\$		
7091	034516	052737	001000	001176	BIS	#1000, \$REG6		
7092	034524	013737	001174	001200	MOV	\$REG5, \$REG7	23\$:	: COMPUTE THE HEADER VRC
7093	034532	013701	001176		MOV	\$REG6, R1		
7094	034536	043737	001176	001200	BIC	\$REG6, \$REG7		
7095	034544	043701	001174		BIC	\$REG5, R1		
7096	034550	050137	001200		BIS	R1, \$REG7		
7097	034554	104410			RESREG			
7098	034556	000207			RTS	PC		
7099								

7100
7101
7102
7103
7104
7105
7106
7107
7108
7109
7110
7111
7112
7113
7114
7115
7116
7117
7118
7119
7120
7121
7122
7123
7124
7125
7126
7127
7128
7129
7130
7131
7132
7133
7134
7135
7136
7137
7138
7139
7140
7141
7142
7143
7144
7145
7146
7147
7148
7148
7149
7150
7151
7152
7153
7154
7155

.SBTTL RK611/RK06-RK07 UNIBUS DRIVER FOR SEQUENTIAL OPERATIONS (REV. 0.10)

;*COPYRIGHT (C) 1975,1976,1977
 ;*DIGITAL EQUIPMENT CORP.
 ;*MAYNARD, MA. 01754
 ;*AUTHOR: ROY SPITZER

.SBTTL *WATCH-DOG TIMER

```

*****
*
* THE WATCH-DOG TIMER DOES A PSEUDO-TIMING OF RK06-RK07 UNIBUS
* SUBSYSTEM COMMAND. SINCE ONE CAN NOT GUARANTEE THAT A
* REAL-TIME CLOCK (KW11-P OR KW11-L) IS ON THE SYSTEM
* THE RK06-RK07 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 SIMULTEOUSLY. 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 CMDTO 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 20$ ;IF NOT ZERO RETURN
        MOV W.MILI,W.MTIM ;REINITIALIZE MILLISECOND TIMER
        TSTB W.TIME ;CHECK IF DRIVE IS BEING TIMED
        BEQ 20$ ;NO, RETURN
        MOV RKPRI,PS ;LOCK OUT RK06-RK07 INTERRUPTS
        MOV RKBAS,R2 ;LOAD BASE OF RK06-RK07 REGISTERS
        DEC W.DRV ;DECREMENT COMMAND TIMER
        BNE 20$ ;RETURN IF NO TIME OUT
  
```

```

034560 010546
034562 010446
034564 010346
034566 010246
034570 013746 177776
034574 005337 003066
034600 001034
034602 013737 003070 003066
034610 105737 003110
034614 001426
034616 013737 003052 177776
034624 013702 003046
034630 005337 003124
034634 001016
  
```

7156	034636	105037	003110		CLRB	W.TIME		;RESET TIMING INDICATOR
7157	034642	013705	003122		MOV	PBLKT,R5		;LOAD ADDRESS OF PARAMETER BLOCK
7158								; TABLE FOR INDEXING
7159	034646	052765	000100	000014	BIS	#CMDTO,P.PRST(R5)		;SET COMMAND TIME OUT
7160	034654	020537	003064		CMP	R5,O.WAIT		;CHECK IF DRIVER IS WAITING FOR
7161								; COMMAND COMPLETION
7162	034660	001002			BNE	5\$;NO, DO NOT ALTER WAITING FOR
7163								; COMMAND COMPLETION
7164	034662	005037	003064		CLR	O.WAIT		;CLEAR WAIT FOR COMMAND COMPLETION
7165	034666	004737	040142	5\$:	JSR	PC,R.ABNL		;BRANCH TO ERROR ROUTINE
7166	034672	012637	177776	20\$:	MOV	(SP)+,PS		;RESTORE PSW
7167	034676	012602			MOV	(SP)+,R2		;RESTORE R2
7168	034700	012603			MOV	(SP)+,R3		;RESTORE R3
7169	034702	012604			MOV	(SP)+,R4		;RESTORE R4
7170	034704	012605			MOV	(SP)+,R5		;RESTORE R5
7171	034706	000207			RTS	PC		;RETURN

7172
7173
7174
7175
7176
7177
7178
7179
7180
7181
7182
7183
7184
7185
7186
7187
7188
7189
7190
7191
7192
7193
7194
7195
7196
7197
7198
7199
7200
7201
7202
7203
7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214
7215

.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 (SUCESSFUL COMPLETION OF COMMAND)
* 2.) A.ABNL ADDRESS OF ABNORMAL RETURN (UNSUCESSFUL 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
*****
  
```

7216 034710 010546
 7217 034712 010446
 7218 034714 010346
 7219 034716 010246
 7220 034720 010146
 7221 034722 010046
 7222 034724 013702 003046
 7223 034730 016237 000010 003012
 7224 034736 032737 001000 003012
 7225 034744 001407
 7226 034746 052737 100000 003062
 7227 034754 004737 040166

```

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 1$ ;NO, CONTINUE PROCESSING
        BIS #E.MDS,E.CONT ;SET MULTIPLE DRIVE SELECT
        JSR PC,R.CONT ;REPORT ERROR
  
```

```

7228 034760 000137 037140          JMP      I.RTRN          ;RETURN
7229
7230 034764 105737 003104          1$:    TSTB      I.ISRL          ;CHECK IF INTERRUPT OR RELEASE
7231 034770 001410                    BEQ      6$              ;NO, CHECK IF DRIVE AVAILABLE
7232 034772 100403                    BMI      5$              ;CHECK IF RELEASE COMMAND
7233 034774 105037 003104          CLR      I.ISRL          ;YES,CLEAR FLAG
7234 035000 000473                    BR       I.I00           ;CONTINUE PROCESSING INTERRUPT
7235
7236 035002 105037 003104          5$:    CLR      I.ISRL          ;CLEAR FLAG
7237 035006 000137 036122          JMP      I.ATTN          ;GO PROCESS DRIVE ATTENTIONS
7238
7239 035012 032737 010400 003012 6$:    BIT      #NED!UFE,T.CS2 ;CHECK FOR NON-EXISTENT DRIVE OR
7240                                ; UNIT FIELD ERROR
7241 035020 001413                    BEQ      7$              ;NO, WAIT FOR DUAL ACCESS INTERRUPT
7242 035022 013704 003012          MOV      T.CS2,R4        ;LOAD R4 FOR DRIVE NUMBER
7243 035026 042704 177770          BIC      #^C<DRVMSK>,R4 ;KEEP DRIVE BITS
7244 035032 013705 003122          MOV      PBLKT,R5        ;STORE PARAMETER BLOCK ADDRESS
7245 035036 016237 000000 003010  MOV      RKCS1(R2),T.CS1 ;LOAD TEMPORARY CS1 FOR STATUS REPORT
7246 035044 000137 035332          JMP      I.ERRC          ;REPORT ERROR
7247
7248 035050 016237 000012 003030 7$:    MOV      RKDS(R2),T.DS   ;STORE STATUS REGISTER FOR COMPARISON
7249 035056 032737 000001 003030  BIT      #DRA,T.DS       ;CHECK IF DRIVE SEIZED BY OTHER
7250                                ; PORT
7251 035064 001041                    BNE      I.I00           ;NO, CONTINUE PROCESSING INTERRUPT
7252
7253                                ;CHECK IF ANY DATA TRANSFER ERROR EXISTS
7254 035066 032737 164000 003012  BIT      #DLT!WCE!UPE!NEM,T.CS2
7255
7256 035074 001007                    BNE      10$             ;INDICATE ERROR
7257 035076 016237 000014 003026  MOV      RKER(R2),T.ER   ;STORE ERROR REGISTER
7258
7259                                ; CHECK FOR DATA TRANSFER ERROR TYPE ERROR
7260 035104 032737 125700 003026  BIT      #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
7261
7262 035112 001407                    BEQ      11$             ;NO, WAIT FOR RELEASE OF RK06 DRIVE
7263
7264 035114 052737 000010 003062 10$:   BIS      #E.UDAT,E.CONT ;SET UNEXPECTED DATA TYPE ERROR
7265 035122 004737 040166                    JSR      PC,R.CONT       ;REPORT ERROR
7266 035126 000137 037140                    JMP      I.RTRN          ;RESTORE REGISTERS
7267
7268 035132 105037 003110          11$:   CLR      W.TIME          ;RESET TIMING ON THIS DRIVE
7269 035136 005037 003124          CLR      W.DRV           ;CLEAR TIMING COUNT FOR THIS DRIVE
7270 035142 013705 003122          MOV      PBLKT,R5        ;LOAD R5 WITH PARAMETER BLOCK
7271                                ; ADDRESS
7272 035146 052765 010000 000014  BIS      #DRVSZD,P.PRST(R5) ;SET DRIVE SEIZED IN THE
7273                                ; PROGRAM DRIVE STATUS REGISTER
7274 035154 005037 003064          CLR      O.WAIT          ;CLEAR WAIT FOR COMMAND COMPLETION
7275 035160 004737 040142          JSR      PC,R.ABNL       ;INDICATE ABNORMAL TERMINATION
7276 035164 000137 037140          JMP      I.RTRN          ;GO RESTORE REGISTERS
7277
7278 035170 013705 003064          I.I00: MOV      O.WAIT,R5      ;LOAD PARAMETER BLOCK ADDRESS INTO R5
7279 035174 001002                    BNE      2$              ;IS COMMAND WAITING PROCESSING
7280                                ; YES, DO PROCESSING
7281 035176 000137 036122          JMP      I.ATTN          ;NO, PROCESS ATTENTION
7282
7283 035202 013704 003012          2$:    MOV      T.CS2,R4        ;STORE RKCS2 FOR DRIVE NUMBER
  
```

```

7284 035206 042704 177770          BIC      #^C<DRVMSK>,R4 ;MASK OUT UNNECESSARY BITS
7285
7286
7287 035212 126504 000000          CMPB    P.DRVN(R5),R4 ;CHECK IF DRIVE NUMBER IS EXPECTED
7288 035216 001401                    BEQ     3$ ;YES, CONTINUE
7289 035220 000000                    HALT    ;NO, DRIVER ERROR
7290 035222 122765 000164 000001 3$:  CMPB    #RDALHD,P.CMND(R5) ;CHECK IF READ ALL HEADERS
7291 035230 001002                    BNE    10$ ;NO, EXECUTE NORMAL DATA TRANSFER
7292 035232 000137 035570          JMP     I.HDAL ;GO EXECUTE SPECIAL HEADER SEQUENCE
7293
7294 035236 005037 003064          10$:   CLR     O.WAIT ;CLEAR WAIT FOR COMMAND COMPLETION
7295 035242 005037 003124          CLR     W.DRV ;CLEAR WATCH-DOG TIME
7296 035246 105037 003110          CLRB   W.TIME ;RESET TIMING ON THIS DRIVE
7297 035252 016237 000000 003010  MOV     RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS REGISTER 1
7298 035260 032737 100000 003010  BIT     #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR
7299 035266 001021                    BNE    I.ERRC ;YES, PROCESS ERROR
7300 035270 016237 000016 003024  MOV     RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
7301 035276 133737 003111 003025  BITB   INTMSK,T.ASOF+1 ;CHECK IF DRIVE ATTENTION SET
7302 035304 001004                    BNE    15$ ;YES, REPORT ERROR
7303 035306 004737 040154          JSR    PC,R.NORM ;INDICATE NORMAL RETURN
7304 035312 000137 037140          JMP     I.RTRN ;RESTORE REGISTERS
7305
7306 035316 052765 000010 000014 15$:   BIS    #UEXATT,P.PRST(R5) ;SET UNEXPECTED ATTENTION
7307
7308 035324 004737 037610          I.ERRA: JSR    PC,I.CSTS ;STORE CONTROLLER STATUS
7309 035330 000405                    BR     I.ERR ;STORE PATTERN AND POSITION INFORMATION
7310
7311 035332 013765 003010 000016  I.ERRC: MOV     T.CS1,P.CS1(R5) ;GET ERROR RKCS1
7312 035340 004737 037632          JSR    PC,I.CST1 ;GET REST OF CONTROLLER STATUS
7313 035344 016265 000032 000062  I.ERR: MOV     RKECPT(R2),P.EPAT(R5) ;STORE ECC PATTERN
7314 035352 016265 000030 000060  MOV     RKECPS(R2),P.EPOS(R5) ;STORE ECC POSITION
7315 035360 004037 037156          JSR    RO,I.CCLR ;CLEAR CONTROLLER
7316 035364 037140          I.RTRN ;ERROR RETURN
7317 035366 032765 010400 000020  BIT     #NED!UFE,P.CS2(R5) ;CHECK IF IT WAS NON-EXISTENT DRIVE OR
7318 ; UNIT FIELD ERROR
7319 035374 001046                    BNE    5$ ;YES, REPORT ERROR
7320 035376 004037 037714          JSR    RO,I.STAT ;GATHER DRIVE STATUS
7321 035402 037140          I.RTRN ;ERROR RETURN
7322 035404 112737 000005 003010  MOVB   #DR.CLR,T.CS1 ;LOAD COMMAND
7323 035412 004037 037240          JSR    RO,I.ISSU ;ISSUE DRIVE CLEAR
7324 035416 037140          I.RTRN ;ERROR RETURN
7325 035420 133737 003111 003025  BITB   INTMSK,T.ASOF+1 ;CHECK IF ATTENTION RESET
7326 035426 001407                    BEQ    2$ ;NO, INDICATE DRIVE ERROR
7327 035430 052737 000020 003062  BIS    #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
7328 ; WITH CLEAR
7329 035436 004737 040166          JSR    PC,R.CONT ;REPORT CONTROLLER ERROR
7330 035442 000137 037140          JMP     I.RTRN ;GO RESTORE REGISTERS
7331
7332 035446 032737 040000 003034 2$:   BIT     #S.DSC,T.MR2 ;CHECK IF DRIVE STATUS CHANGE CLEARED
7333 035454 001403                    BEQ    3$ ;YES, CHECK FAULT
7334 035456 052765 000040 000014  BIS    #DRVDSC,P.PRST(R5) ;SET DSC DID NOT CLEAR
7335 035464 032737 001000 003036 3$:   BIT     #S.PAR,T.MR3 ;CHECK IF DRIVE PARITY ERROR
7336 035472 001407                    BEQ    5$ ;NO, INDICATE ABNORMAL TERMINATION
7337 035474 052737 002000 003062  BIS    #E.DPAR,E.CONT ;SET DRIVE PARITY ERROR
7338 035502 004737 040166          JSR    PC,R.CONT ;INDICATE CONTROLLER ERROR
7339 035506 000137 037140          JMP     I.RTRN ;RETURN

```

```

7340
7341 035512 032765 000020 000014 5$: BIT #DRVHRD,P.PRST(R5) ;CHECK IF HARD DRIVE ERROR
7342 035520 001017 BNE 10$ ;YES, GO REPORT ERROR
7343 035522 032737 020000 003034 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS CYCLING DOWN
7344 035530 001413 BEQ 10$ ;NO, REPORT ERROR
7345 035532 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING
7346 035540 113737 003111 003110 MOVB INTMSK,W.TIME ;SET UP 8 SECONDS FOR DRIVE TO CYCLE UP
7347 035546 013737 003074 003124 MOV W.8SEC,W.DRV
7348 035554 000137 037140 JMP I.RTRN ;GO RESTORE REGISTERS
7349
7350 035560 004737 040142 10$: JSR PC,R.ABNL ;GO REPORT ERROR
7351 035564 000137 037140 JMP I.RTRN ;GO RESTORE REGISTERS
7352
7353 .SBTTL *READ ALL HEADERS INTERRUPT SEQUENCE
7354
7355 035570 016237 000000 003010 I.HDAL: MOV RKCS1(R2),T.CS1 ;STORE CS1 TO CHECK CONTROLLER
7356 ; ERROR
7357 035576 032737 100000 003010 BIT #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR
7358 035604 001422 BEQ 5$ ;NO, CHECK FOR ATTENTION
7359
7360 035606 005037 003064 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETE
7361 035612 105037 003110 CLR W.TIME ;RESET TIMING ON DRIVE
7362 035616 005037 003124 CLR W.DRV ;CLEAR TIME OUT COUNT
7363 035622 013765 003010 000016 MOV T.CS1,P.CS1(R5) ;STORE ERROR RKCS1
7364 035630 004737 037632 JSR PC,I.CST1 ;STORE CONTROLLER REGISTERS
7365 035634 004037 037156 JSR RO,I.CCLR ;CLEAR CONTROLLER
7366 035640 037140 I.RTRN ;ERROR RETURN
7367 035642 004737 040142 JSR PC,R.ABNL ;INDICATE ERROR RETURN
7368 035646 000137 037140 JMP I.RTRN ;RESTORE REGISTERS
7369
7370 035652 016237 000016 003024 5$: MOV RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
7371 035660 133737 003111 003025 BITB INTMSK,T.ASOF+1 ;CHECK IF DRIVE ATTENTION IS SET
7372 035666 001410 BEQ 7$ ;NO, CHECK IF READ ALL HEADERS
7373 035670 005037 003064 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETION
7374 035674 105037 003110 CLR W.TIME ;RESET TIMING ON DRIVE
7375 035700 005037 003124 CLR W.DRV ;CLEAR TIME OUT COUNT
7376 035704 000137 035324 JMP I.ERRA ;GO REPORT ERROR
7377
7378 035710 013701 003100 7$: MOV HDR.AD,R1 ;GET MAIN MEMORY ADDRESS
7379 035714 016221 000024 MOV RKDB(R2),(R1)+ ;GET FIRST WORD OF HEADER
7380 035720 016221 000024 MOV RKDB(R2),(R1)+ ;GET SECOND WORD OF HEADER
7381 035724 016221 000024 MOV RKDB(R2),(R1)+ ;GET THIRD WORD OF HEADER
7382 035730 010137 003100 MOV R1,HDR.AD ;STORE ADDRESS FOR NEXT HEADER
7383 035734 016237 000010 003012 MOV RKCS2(R2),T.CS2 ;STORE CS2 TO CHECK FOR DATA LATE
7384 035742 032737 100000 003012 BIT #DLT,T.CS2 ;CHECK FOR DATA LATE
7385 035750 001055 BNE 35$ ;YES, REPORT ERROR
7386 035752 005337 003102 DEC HDR.CT ;DECREMENT NUMBER OF HEADER YET TO READ
7387 035756 001026 BNE 25$ ;IF NON-ZERO, GO ISSUE NEXT READ HEADER
7388 035760 005037 003064 CLR O.WAIT ;CLEAR DRIVER WAITING FOR COMMAND COMPLETION
7389 035764 005037 003124 CLR W.DRV ;CLEAR TIME OUT COUNT FOR THIS DRIVE
7390 035770 105037 003110 CLR W.TIME ;CLEAR WATCH DOG TIME ON THIS DRIVE
7391 035774 012762 000003 000026 MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER FOR SECTOR COUNT
7392 036002 112737 000001 003010 MOVB #DR.SEL,T.CS1 ;LOAD SELECT COMMAND
7393 036010 004037 037240 JSR RO,I.ISSU ;GET SECTOR COUNT
7394 036014 037140 I.RTRN ;ERROR RETURN
7395 036016 013765 003036 000056 MOV T.MR3,P.B11(R5) ;LOAD SECTOR COUNT

```

```

7396 036024 004737 040154      JSR    PC,R.NORM      ;INDICATE NORMAL TERMINATION
7397 036030 000137 037140      JMP    I.RTRN        ;RESTORE REGISTERS
7398
7399 036034 016562 000002 000020 25$:  MOV    P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
7400 036042 016562 000004 000006      MOV    P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
7401 036050 116565 000007 000017      MOV    P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
7402 036056 042765 165777 000016      BIC    #^C<CDT!CFMT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT AND
7403                                     ; DRIVE TYPE
7404 036064 112765 000125 000016      MOV    #RDHEAD,P.CS1(R5) ;STORE COMMAND ISSUED
7405 036072 016562 000016 000000      MOV    P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
7406 036100 000137 037140      JMP    I.RTRN        ;RESTORE REGISTERS
7407
7408 036104 052737 000400 003062 35$:  BIS    #E.DLT,E.CONT    ;SET DATA LATE WHILE UNLOADING HEADER
7409 036112 004737 040166      JSR    PC,R.CONT     ;REPORT ERROR
7410 036116 000137 037140      JMP    I.RTRN        ;RESTORE REGISTERS
7411
7412                                     .SBTTL  *DRIVE ATTENTION SCANNER
7413
7414 036122 016237 000000 003010 I.ATTN: MOV    RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS
7415                                     ; REGISTER 1 FOR COMPARISON
7416 036130 032737 100000 003010      BIT    #CERR,T.CS1    ;CHECK IF CONTROLLER ERROR OCCURRED
7417 036136 001441                                     BEQ    5$            ;NO, CHECK IF ATTENTION
7418
7419                                     ;CHECK IF ANY DATA TRANSFER TYPE ERROR EXISTS
7420 036140 032737 164000 003012      BIT    #DLT!WCE!UPE!NEM,T.CS2
7421
7422 036146 001007                                     BNE    1$            ;INDICATE ERROR
7423 036150 016237 000014 003026      MOV    RKER(R2),T.ER  ;STORE ERROR REGISTER
7424
7425                                     ; CHECK FOR DATA TRANSFER ERROR TYPE
7426 036156 032737 125700 003026      BIT    #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
7427
7428 036164 001407                                     BEQ    2$            ;NO DATA TRANSFER ERROR
7429
7430 036166 052737 000010 003062 1$:  BIS    #E.UDAT,E.CONT  ;SET UNEXPECTED DATA TYPE ERROR
7431 036174 004737 040166      JSR    PC,R.CONT     ;REPORT ERROR
7432 036200 000137 037140      JMP    I.RTRN        ;RESTORE REGISTERS
7433
7434 036204 013704 003012 2$:  MOV    T.CS2,R4       ;SAVE CS2 FOR REGISTER NUMBER
7435 036210 042704 177770      BIC    #^C<DRVMSK>,R4 ;STRIP OFF JUNK
7436 036214 105037 003110      CLR    W.TIME        ;CLEAR WATCH DOG TIMER
7437 036220 005037 003124      CLR    W.DRV         ;RESET TIMER VALUE
7438 036224 013705 003122      MOV    PBLKT,R5     ;STORE PARAMETER BLOCK ADDRESS IN R5
7439
7440                                     ; CLEAR DRIVE POSITIONING AND DRIVE POSITIONED FOR DATA TRANSFER
7441                                     ; IN PROGRAM DEVICE STATUS REGISTER
7442 036230 042765 000006 000014      BIC    #DRVPOS!DRVPDT,P.PRST(R5)
7443
7444 036236 000137 035332      JMP    I.ERRC        ;GO REPORT ERROR
7445
7446 036242 032737 040000 003010 5$:  BIT    #DI,T.CS1     ;CHECK IF ANY DRIVE ATTENTION
7447 036250 001002                                     BNE    6$            ;YES, PROCESS INTERRUPT
7448 036252 000137 037140      JMP    I.RTRN        ;RESTORE REGISTERS
7449
7450 036256 016237 000016 003024 6$:  MOV    RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
7451 036264 105737 003025      TSTB   T.ASOF+1     ;CHECK IF ANY ATTENTIONS SET
  
```



```

7508 .SBTTL *ATTENTION ERROR HANDLER
7509
7510 036572 042765 000004 000014 I.AERR: BIC #DRVPT,P.PRST(R5) ;RESET POSITIONING IN PROGRESS BECAUSE
7511 ; OF DATA TRANSFER
7512 036600 105037 003110 CLR W.TIME ;CLEAR TIMING FOR THIS DRIVE
7513 036604 005037 003124 CLR W.DRV ;RESET WATCH-DOG TIME
7514 036610 042765 177741 000016 BIC #177741,P.CS1(R5) ;KEEP COMMAND ISSUED
7515 036616 042737 000036 003010 BIC #36,T.CS1 ;KEEP CURRENT CONTROLLER STATUS
7516 036624 053765 003010 000016 BIS T.CS1,P.CS1(R5) ;MAKE GOOD MESSAGE
7517 036632 013765 003012 000020 MOV T.CS2,P.CS2(R5) ;STORE CONTROLLER REGISTERS
7518 036640 013765 003014 000022 MOV T.WCR,P.WCR(R5)
7519 036646 013765 003016 000024 MOV T.BA,P.BAR(R5)
7520 036654 013765 003020 000026 MOV T.DA,P.DTS(R5)
7521 036662 013765 003022 000030 MOV T.DC,P.DCYL(R5)
7522 036670 013765 003024 000032 MOV T.ASOF,P.ASOF(R5)
7523 036676 013765 003026 000034 MOV T.ER,P.ER(R5)
7524 036704 013765 003030 000036 MOV T.DS,P.DS(R5)
7525 036712 004037 037714 JSR RO,I.STAT ;GATHER DRIVE STATUS
7526 036716 037140 I.RTRN ;ERROR RETURN
7527 036720 112737 000005 003010 MOVB #DR.CLR,T.CS1 ;LOAD COMMAND
7528 036726 004037 037240 JSR RO,I.ISSU ;CLEAR DRIVE ERRORS
7529 036732 037140 I.RTRN ;ERROR RETURN
7530 036734 133737 003111 003025 BITB INTMSK,T.ASOF+1 ;CHECK IF ATTENTION RESET
7531 036742 001407 BEQ 2$ ;YES, FLAG DRIVE ERROR
7532 036744 052737 000020 003062 BIS #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
7533 036752 004737 040166 JSR PC,R.CONT ;REPORT ERROR
7534 036756 000137 037140 JMP I.RTRN ;RESTORE REGISTERS
7535
7536 036762 032765 000020 000014 2$: BIT #DRVHRD,P.PRST(R5) ;CHECK IF AWARD DRIVE ERROR
7537 036770 001017 BNE 10$ ;YES, REPORT ERROR
7538 036772 032737 020000 003034 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS UNLOADING
7539 037000 001413 BEQ 10$ ;NO, REPORT ERROR
7540 037002 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING DUE TO ERROR
7541 037010 113737 003111 003110 MOVB INTMSK,W.TIME ;SET TIMING ON THIS DRIVE
7542 037016 013737 003074 003124 MOV W.8SEC,W.DRV ;LOAD 8 SECONDS FOR CYCLE UP TIME
7543 037024 000137 037140 JMP I.RTRN ;RESTORE REGISTERS
7544
7545 037030 004737 040142 10$: JSR PC,R.ABNL ;REPORT ERROR
7546 037034 000137 037140 JMP I.RTRN ;RESTORE REGISTERS
7547
7548 .SBTTL *ERROR CAUSING DRIVE TO UNLOAD
7549
7550 037040 052765 020000 000014 I.UNLD: BIS #E.UNLD,P.PRST(R5) ;CLEAR DRIVE UNLOADING BECAUSE OF ERROR
7551 037046 112737 000005 003010 MOVB #DR.CLR,T.CS1 ;LOAD IN DRIVE CLEAR
7552 037054 004037 037240 JSR RO,I.ISSU ;GO ISSUE DRIVE CLEAR
7553 037060 037140 I.RTRN ;ERROR RETURN
7554 037062 136437 003111 003025 BITB INTMSK(R4),T.ASOF+1 ;CHECK IF ATTENTION CLEARED
7555 037070 001406 BEQ 15$ ;YES, CONTINUE
7556 037072 012737 000020 003062 MOV #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
7557 037100 004737 040166 JSR PC,R.CONT ;REPORT ERROR
7558 037104 000415 BR I.RTRN ;RESTORE REGISTERS
7559
7560 037106 032737 040000 003034 15$: BIT #S.DSC,T.MR2 ;CHECK IF DRIVE STAUUS CHANGE RESET
7561 037114 001403 BEQ 20$ ;YES, CONTINUE
7562 037116 052765 000040 000014 BIS #DRVDSK,P.PRST(R5) ;SET DRIVE STAUUS CHANGE DID NOT CLEAR
7563 037124 105037 003110 20$: CLRB W.TIME ;RESET TIMING ON THIS DRIVE
  
```

7564	037130	005037	003124	CLR	W.DRV	;CLEAR TIME COUNT
7565	037134	004737	040142	JSR	PC,R.ABNL	;REPORT ERROR
7566						
7567	037140	012600		I.RTRN: MOV	(SP)+,R0	;RESTORE R0
7568	037142	012601		MOV	(SP)+,R1	;RESTORE R1
7569	037144	012602		MOV	(SP)+,R2	;RESTORE R2
7570	037146	012603		MOV	(SP)+,R3	;RESTORE R3
7571	037150	012604		MOV	(SP)+,R4	;RESTORE R4
7572	037152	012605		MOV	(SP)+,R5	;RESTORE R5
7573	037154	000002		RTI		;RETURN
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
7601
7602
7603
7604
7605
7606
7607
7608
7609

.SBTTL *CONTROLLER CLEAR ROUTINE

```

:*****
:
: 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
: R5            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     5$            ;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

5$:     MOV     #IE,RKCS1(R2)  ;SET INTERRUPT ENABLE
        MOVB   #-1,I.ISRL    ;SET INTERRUPT ENABLE ISSUED
        TST    (R0)+         ;ADJUST FOR NORMAL RETURN
        RTS     R0            ;RETURN
  
```

7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638 037240 013746 003010
7639 037244 005037 003012
7640 037250 116537 000000 003012
7641 037256 013762 003012 000010
7642 037264 116537 000007 003011
7643 037272 142737 177753 003011
7644
7645 037300 013762 003010 000000
7646 037306 105762 000000
7647 037312 100375
7648 037314 004737 037462
7649 037320 032737 100000 003010
7650 037326 001437
7651 037330 032737 001000 003012
7652 037336 001406
7653 037340 052737 100000 003062
7654 037346 004737 040166
7655 037352 000440
7656
7657
7658 037354 032737 024000 003010
7659 037362 001027
7660 037364 032737 176400 003012
7661 037372 001023
7662 037374 032737 131761 003026
7663 037402 001017
7664
7665 037404 122716 000005

.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      R0,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
        MOV      P.DRVN(R5),T.CS2 ;LOAD IN DRIVE NUMBER
        MOV      T.CS2,RKCS2(R2) ;LOAD DRIVE NUMBER FOR COMMAND
        MOV      P.CS1H(R5),T.CS1+1 ;STORE BITS 8-15 OF CS1
        BIC      #^C<B.CDT!B.CFMT>,T.CS1+1 ;CLEAR ALL BITS EXCEPT
        ; FORMAT AND DRIVE TYPE
        MOV      T.CS1,RKCS1(R2) ;ISSUE COMMAND
1$:     TSTB     RKCS1(R2)      ;WAIT FOR READY
        BPL      1$
        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      #UFE!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

```

7666	037410	001003				BNE	3\$:NO, DO NOT SET DRIVE HARD ERROR
7667	037412	052765	000020	000014		BIS	#DRVHRD,P.PRST(R5)	;SET HARD DRIVE ERROR
7668	037420	004037	037156		3\$:	JSR	RO,I.CCLR	;GO ISSUE A CONTROLLER CLEAR
7669	037424	037454				10\$;ERROR RETURN
7670	037426	012762	000100	000000	5\$:	MOV	#IE,RKCS1(R2)	;SET INTERRUPT ENABLE
7671	037434	005726				TST	(SP)+	;ADJUST STACK
7672	037436	005720				TST	(RO)+	;ADJUST RO FOR NORMAL RETURN
7673	037440	000200				RTS	RO	;RETURN
7674								
7675	037442	052737	001000	003062	7\$:	BIS	#E.CERR,E.CONT	;SET CONTROLLER ERROR DURING
7676								; DRIVER SERVICING
7677	037450	004737	040166			JSR	PC,R.CONT	;REPORT ERROR
7678	037454	005726			10\$:	TST	(SP)+	;ADJUST STACK
7679	037456	011000				MOV	(RO),RO	;ADJUST RO FOR ERROR RETURN
7680	037460	000200				RTS	RO	;RETURN

7681
 7682
 7683
 7684
 7685
 7686
 7687
 7688
 7689
 7690
 7691
 7692
 7693
 7694
 7695
 7696
 7697
 7698 037462 016237 000000 003010
 7699 037470 016237 000010 003012
 7700 037476 016237 000002 003014
 7701 037504 016237 000004 003016
 7702 037512 016237 000006 003020
 7703 037520 016237 000012 003030
 7704 037526 016237 000014 003026
 7705 037534 016237 000016 003024
 7706 037542 016237 000020 003022
 7707 037550 016237 000026 003032
 7708 037556 016237 000034 003034
 7709 037564 016237 000036 003036
 7710 037572 016237 000030 003040
 7711 037600 016237 000032 003042
 7712 037606 000207

```

.SBTTL *STORE RK611 UNIBUS REGISTERS
:*****
:*
:* 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
  
```

7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757

.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

:*****

```

7743 037610 042765 177741 000016 I.CSTS: BIC #177741,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FUNCTION
7744 ;OF LAST COMMAND ISSUED
7745 037616 042737 000036 003010 BIC #36,T.CS1 ;CLEAR FUNCTION OF CS1 STATUS
7746 037624 053765 003010 000016 BIS T.CS1,P.CS1(R5) ;GENERATE CS1 STATUS INFORMATION
7747 037632 016265 000010 000020 I.CST1: MOV RKCS2(R2),P.CS2(R5) ;STORE COMMAND AND STATUS REGISTER 2
7748 037640 016265 000002 000022 MOV RKWC(R2),P.WCR(R5) ;STORE WORD COUNT REGISTER
7749 037646 016265 000004 000024 MOV RKBA(R2),P.BAR(R5) ;STORE BUS ADDRESS REGISTER
7750 037654 016265 000006 000026 MOV RKDA(R2),P.DTS(R5) ;STORE DESIRED TRACK AND SECTOR
7751 037662 016265 000012 000036 MOV RKDS(R2),P.DS(R5) ;STORE DRIVE STATUS REGISTER
7752 037670 016265 000014 000034 MOV RKER(R2),P.ER(R5) ;STORE ERROR REGISTER
7753 037676 016265 000016 000032 MOV RKASOF(R2),P.ASOF(R5) ;STORE ATTENTION SUMMARY AND
7754 ; OFFSET
7755 037704 016265 000020 000030 MOV RKDCYL(R2),P.DCYL(R5) ;STORE CYLINDER ADDRESS
7756 037712 000207 RTS PC ;RETURN
7757

```

7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813

.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

```

```

7784 037714 012762 000001 000026 I.STAT: MOV #1,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
7785 ; FOR STATUS BYTE 01
7786 037722 112737 000001 003010 MOVB #DR.SEL,T.CS1 ;LOAD COMMAND
7787 037730 004037 037240 JSR RO,I.ISSU ;GET STATUS BYTES 01
7788 037734 040124 3$ ;ERROR RETURN
7789 037736 013765 003034 000044 MOV T.MR2,P.A01(R5) ;STORE STATUS BYTE 01 MESS A
7790 037744 013765 003036 000046 MOV T.MR3,P.B01(R5) ;STORE STATUS BYTE 01 MESS B
7791 037752 012762 000002 000026 MOV #2,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
7792 ; FOR STATUS BYTE 10
7793 037760 112737 000001 003010 MOVB #DR.SEL,T.CS1 ;LOAD COMMAND
7794 037766 004037 037240 JSR RO,I.ISSU ;GET STATUS BYTES 10
7795 037772 040124 3$ ;ERROR RETURN
7796 037774 013765 003034 000050 MOV T.MR2,P.A10(R5) ;STORE STATUS BYTE 10 MESS A
7797 040002 013765 003036 000052 MOV T.MR3,P.B10(R5) ;STORE STATUS BYTE 10 MESS B
7798 040010 012762 000003 000026 MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER
7799 ; FOR STATUS BYTE 11
7800 040016 112737 000001 003010 MOVB #DR.SEL,T.CS1 ;LOAD COMMAND
7801 040024 004037 037240 JSR RO,I.ISSU ;GET STATUS BYTES 11
7802 040030 040124 3$ ;ERROR RETURN
7803 040032 013765 003034 000054 MOV T.MR2,P.A11(R5) ;STORE STATUS BYTE 11 MESS A
7804 040040 013765 003036 000056 MOV T.MR3,P.B11(R5) ;STORE STATUS BYTE 11 MESS B
7805 040046 005062 000026 CLR RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
7806 ; FOR STATUS BYTE 00
7807 040052 112737 000001 003010 MOVB #DR.SEL,T.CS1 ;LOAD COMMAND
7808 040060 004037 037240 JSR RO,I.ISSU ;GET STATUS BYTES 00
7809 040064 040124 3$ ;ERROR RETURN
7810 040066 013765 003034 000040 MOV T.MR2,P.A00(R5) ;STORE STATUS BYTE 00 MESS A
7811 040074 013765 003036 000042 MOV T.MR3,P.B00(R5) ;STORE STATUS BYTE 00 MESS B
7812 040102 032737 001000 003036 BIT #S.PAR,T.MR3 ;CHECK IF BAD PARITY DETECTED BY DRIVE
7813 040110 001407 BEQ 5$ ;NO, RETURN NORMALLY

```


7814	040112	052737	002000	003062		BIS	#E.DPAR,E.CONT	:INDICATE BAD PARITY DETECTED BY DRIVE
7815	040120	004737	040166			JSR	PC,R.CONT	:REPORT ERROR
7816	040124	011000			3\$:	MOV	(R0),R0	:LOAD R0 FOR ERROR RETURN
7817	040126	000200				RTS	R0	:RETURN
7818								
7819	040130	052765	001000	000014	5\$:	BIS	#PBSVAL,P.PRST(R5)	:SET PARAMETER BLOCK STATUS VALID
7820	040136	005720				TST	(R0)+	:ADJUST R0 FOR NORMAL RETURN
7821	040140	000200				RTS	R0	:RETURN
7822								

```
7823 .SBTTL *COMMON DRIVER RETURNS
7824
7825 040142 105037 003111 R.ABNL: CLRB INTMSK ;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
7826 040146 004777 142704 JSR PC,@A.ABNL ;INDICATE ABNORMAL RETURN
7827 040152 000207 RTS PC ;RETURN
7828
7829 040154 105037 003111 R.NORM: CLRB INTMSK ;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
7830 040160 004777 142670 JSR PC,@A.NORM ;INDICATE NORMAL RETURN
7831 040164 000207 RTS PC ;RETURN
7832
7833 040166 105037 003111 R.CONT: CLRB INTMSK ;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
7834 040172 105037 003110 CLRB W.TIME ;RESET WATCH DOG TIMING ON THIS DRIVE
7835 040176 005037 003124 CLR W.DRV ;CLEAR TIMING COUNT FOR THIS DRIVE
7836 040202 004777 142652 JSR PC,@A.CONT ;INDICATE CONTROLLER ERROR RETURN
7837 040206 000207 RTS PC ;RETURN
```

7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893

```
.SBTTL *COMMAND INITATOR

*****
*
* THIS SUBROUTINE WILL INITIATE ALL COMMANDS AS SPECIFIED
* BY THE COMMAND FIELD OF THE PARAMETER BLOCK. THE FOLLOWING
* SPECIAL COMMAND ARE ALSO EXECUTED:
*
*     RELEASE
*     CONROLLER 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    #^C<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.DRV    ;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
        ;
```

7869	040210	010546		
7870	040212	010446		
7871	040214	010346		
7872	040216	010246		
7873	040220	010146		
7874	040222	010046		
7875	040224	013746	177776	
7876	040230	013737	003052	177776
7877	040236	017605	000016	
7878	040242	062766	000002	000016
7879	040250	016504	000000	
7880	040254	042704	177770	
7881	040260	010537	003122	
7882	040264	116437	003112	003111
7883	040272	116437	003112	003110
7884	040300	013737	003072	003124
7885				
7886	040306	013702	003046	

```

7894 040312 042765 075176 000014      BIC      #^C<DRVUSE!W.WCK!NOCHK!DRPDRV!DTBAII>,P.PRST(R5)
7895
7896 040320 010500      MOV      R5,R0      ;STORE PARAMETER BLOCK ADDRESS
7897 040322 062700 000016      ADD      #P.CS1,R0   ;CALCULATE FIRST LOCATION TO BE CLEARED
7898 040326 010501      MOV      R5,R1      ;STORE PARAMETER BLOCK ADDRESS
7899 040330 062701 000062      ADD      #P.EPAT,R1  ;CALCULATE LAST LOCATION TO BE CLEARED
7900
7901 040334 005020      1$:     CLR      (R0)+      ;CLEAR RETURN PARAMETER
7902 040336 020001      CMP      R0,R1      ;CHECK IF FINISHED
7903 040340 101775      BLOS     1$         ;NO, CLEAR NEXT RETURN PARAMETER
7904 040342 105037 003104      CLRB    I.ISRL     ;CLEAR RELEASE OR INTERRUPT ISSUED
7905 040346 010465 000020      MOV      R4,P.CS2(R5) ;STORE DRIVE NUMBER
7906 040352 005062 000026      CLR      RKMRI(R2)  ;CLEAR RK06 MAINTENANCE REGISTER 1
7907 040356 132765 000040 000001      BITB    #BIT5,P.CMND(R5) ;CHECK IF SPECIAL COMMAND
7908 040364 001402      BEQ      3$         ;NO, PROCESS
7909 040366 000137 041102      JMP      C.SPEC     ;JUMP TO SPECIAL COMMAND PROCESSOR
7910
7911 040372 122765 000107 000001 3$:     CMPB    #UNLOAD,P.CMND(R5) ;CHECK IF POSITIONING COMMAND
7912                                     ; START SPINDLE
7913                                     ; RECALIBRATE
7914                                     ; OFFSET
7915                                     ; SEEK
7916                                     ; UNLOAD
7917
7918 040400 101174      BHI      25$        ;NO, DRIVE COMMAND
7919                                     ; SELECT DRIVE
7920                                     ; PACK ACKNOWLEDGE
7921                                     ; CLEAR
7922
7923 040402 122765 000117 000001      CMPB    #SEEK,P.CMND(R5) ;CHECK IF DATA TRANSFER
7924 040410 103540      BLO      20$        ;YES, DATA TRANSFER COMMAND
7925                                     ; READ DATA
7926                                     ; WRITE DATA
7927                                     ; READ HEADER
7928                                     ; WRITE HEADER
7929                                     ; WRITE CHECK
7930 040412 016562 000020 000010      MOV      P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
7931 040420 052765 000002 000014      BIS      #DRVPOS,P.PRST(R5) ;SET DRIVE POSITIONING
7932 040426 005037 003064      CLR      O.WAIT     ;CLEAR WAIT FOR COMMAND
7933 040432 122765 000117 000001      CMPB    #SEEK,P.CMND(R5) ;CHECK IF SEEK
7934 040440 001007      BNE      5$         ;NO, CHECK FOR OFFSET
7935 040442 016562 000002 000020      MOV      P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
7936 040450 016562 000004 000006      MOV      P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
7937 040456 000431      BR       8$         ;GO ISSUE COMMAND
7938
7939 040460 122765 000115 000001 5$:     CMPB    #OFFSET,P.CMND(R5) ;CHECK IF OFFSET
7940 040466 001007      BNE      6$         ;NO, CHECK FOR UNLOAD
7941 040470 116565 000006 000032      MOVB    P.OFST(R5),P.ASOF(R5) ;STORE OFFSET
7942 040476 016562 000032 000016      MOV      P.ASOF(R5),RKASOF(R2) ;LOAD OFFSET REGISTER
7943 040504 000416      BR       8$         ;GO ISSUE COMMAND
7944
7945 040506 122765 000111 000001 6$:     CMPB    #SRTSPL,P.CMND(R5) ;CHECK IF START SPINDLE
7946 040514 001003      BNE      7$         ;NO, CHECK IF RECAL
7947 040516 013737 003076 003124      MOV      W.MIN,W.DRV ;LOAD WATCH DOG TIME FOR 1 MINUTE
7948 040524 122765 000113 000001 7$:     CMPB    #RECAL,P.CMND(R5) ;CHECK IF RECAL
7949 040532 001003      BNE      8$         ;NO, CONTINUE

```

```

7950 040534 013737 003074 003124      MOV      W.8SEC,W.DRV      ;LOAD RECAL TIME FOR 8 SECONDS
7951 040542 116565 000007 000017 8$:     MOVVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
7952 040550 042765 165777 000016      BIC     #^C<CFMT!CDT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT
7953                                     ; AND DRIVE TYPE
7954 040556 116565 000001 000016      MOVVB   P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
7955 040564 042765 000200 000014      BIC     #W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
7956 040572 032765 000400 000014      BIT     #NOCHK,P.PRST(R5) ;CHECK IN NO CHECK MODE
7957 040600 001533                                     BEQ     30$                ;NO, SKIP CLEAR OF INTERRUPT ENABLE
7958 040602 042765 000100 000016      BIC     #IE,P.CS1(R5)      ;CLEAR INTERRUPT ENABLE
7959 040610 016562 000016 000000      MOV     P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
7960 040616 004737 034560 10$:     JSR     PC,W.WTCH          ;CALL WATCH DOG TIMER
7961 040622 016237 000000 003010      MOV     RKCS1(R2),T.CS1    ;STORE COMMAND AND STATUS REGISTER 1
7962 040630 032737 000200 003010      BIT     #RDY,T.CS1        ;WAIT FOR READY
7963 040636 001767                                     BEQ     10$
7964 040640 032737 100000 003010      BIT     #CERR,T.CS1       ;CHECK FOR ERROR
7965 040646 001011                                     BNE     15$                ;YES, GIVE NORMAL RETURN
7966 040650 004737 034560 11$:     JSR     PC,W.WTCH          ;CALL WATCH DOG TIMER
7967 040654 016237 000016 003024      MOV     RKASOF(R2),T.ASOF  ;STORE ATTENTION SUMMARY
7968 040662 133737 003111 003025      BITB   INTMSK,T.ASOF+1    ;CHECK IF INTERRUPT HAS OCCURRED
7969 040670 001767                                     BEQ     11$                ;WAIT FOR DRIVE INTERRUPT
7970 040672 105037 003110 15$:     CLR    W.TIME            ;RESET TIMING ON THIS DRIVE
7971 040676 005037 003124                                     CLR    W.DRV              ;CLEAR DRIVE TIMING COUNT
7972 040702 004737 040154                                     JSR     PC,R.NORM         ;INDICATE COMMAND IS FINISHED
7973 040706 000137 042062                                     JMP     C.RTRN           ;RESTORE REGISTERS
7974
7975 040712 016562 000010 000004 20$:     MOV     P.BALO(R5),RKBA(R2) ;LOAD BUS ADDRESS REGISTER
7976 040720 016562 000012 000002      MOV     P.WC(R5),RKWC(R2)  ;LOAD WORD COUNT REGISTER
7977 040726 016562 000002 000020      MOV     P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
7978 040734 016562 000004 000006      MOV     P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK NUMBER
7979 040742 122765 000131 000001      CMPB   #WRTCHK,P.CMND(R5) ;CHECK IF WRITE CHECK COMMAND
7980 040750 001010                                     BNE     25$                ;NO, GO ISSUE THE COMMAND
7981 040752 032765 000200 000014      BIT     #W.WCK,P.PRST(R5) ;CHECK IF WRITE COMMAND SHOULD BE ISSUED
7982 040760 001404                                     BEQ     25$                ;NO, GO ISSUE THE COMMAND
7983 040762 012765 000123 000016      MOV     #WRDATA,P.CS1(R5) ;ISSUE WRITE COMMAND
7984 040770 000406                                     BR      26$                ;GO ISSUE COMMAND
7985
7986 040772 116565 000001 000016 25$:     MOVVB   P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
7987 041000 042765 000200 000014      BIC     #W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
7988 041006 116565 000007 000017 26$:     MOVVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
7989 041014 142765 177750 000017      BICB   #^C<B.CFMT!B.CDT!B.BA16!B.BA17>,P.CS1+1(R5) ;CLEAR ALL BITS EXCEPT
7990                                     ; FORMAT, DRIVE TYPE, AND BUS ADDRESS
7991                                     ; BITS 16-17
7992 041022 010537 003064      MOV     R5,O.WAIT         ;LOAD WAITING FOR COMMAND
7993 041026 032765 100000 000014      BIT     #DTBA11,P.PRST(R5) ;CHECK IF INHIBIT BUS ADDRESS INCREMENT
7994 041034 001403                                     BEQ     27$                ;NO, LOAD CS2
7995 041036 052765 000020 000020      BIS     #BA1,P.CS2(R5)    ;SET INHIBIT BUS ADDRESS INCREMENT
7996 041044 016562 000020 000010 27$:     MOV     P.CS2(R5),RKCS2(R2) ;LOAD CS2
7997 041052 032765 000400 000014      BIT     #NOCHK,P.PRST(R5) ;CHECK IN NO CHECK MODE
7998 041060 001403                                     BEQ     30$                ;NO, SKIP CLEAR OF INTERRUPT ENABLE
7999 041062 042765 000100 000016      BIC     #IE,P.CS1(R5)     ;CLEAR INTERRUPT ENABLE
8000 041070 016562 000016 000000 30$:     MOV     P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
8001 041076 000137 042062                                     JMP     C.RTRN           ;RESTORE REGISTERS
8002
8003                                     .SBTTL  *SPECIAL COMMAND PROCESSING
8004
8005 041102 122765 000141 000001  C.SPEC: CMPB   #RDSTAT,P.CMND(R5) ;CHECK IF READ DRIVE STATUS

```


8062	041450	112765	000101	000016		MOVB	#SELDIV,P.CS1(R5) ;STORE COMMAND
8063	041456	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
8064	041464	001403				BEQ	11\$;NO, DO NOT RESET INTERRUPT ENABLE
8065	041466	042765	000100	000016		BIC	#IE,P.CS1(R5) ;RESET INTERRUPT ENABLE
8066	041474	016562	000016	000000	11\$:	MOV	P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
8067	041502	000137	042062			JMP	C.RTRN ;RESTORE REGISTERS
8068							
8069	041506	122765	000164	000001	13\$:	CMPB	#RDALHD,P.CMND(R5) ;CHECK IF READ ALL HEADERS
8070	041514	001053				BNE	30\$;NO, CHECK IF CONTROLLER CLEAR
8071	041516	010537	003064			MOV	R5,O.WAIT ;SET WAITING FOR COMMAND COMPLETION
8072	041522	016537	000010	003100		MOV	P.BALO(R5),HDR.AD ;LOAD HEADER ADDRESS
8073	041530	132765	000020	000007		BITB	#B.CFMT,P.CS1H(R5) ;CHECK IF 22 SECTOR FORMANT
8074	041536	001404				BEQ	14\$;YES, LOAD 22 IN HEADER COUNT
8075	041540	012737	000024	003102		MOV	#20.,HDR.CT ;LOAD 20 IN SECTOR COUNT
8076	041546	000403				BR	22\$;GO ISSUE READ HEADER COMMAND
8077							
8078	041550	012737	000026	003102	14\$:	MOV	#22.,HDR.CT ;LOAD 22 IN SECTOR COUNT
8079	041556	016562	000002	000020	22\$:	MOV	P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
8080	041564	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2) ;LOAD TRACK NUMBER
8081	041572	016562	000020	000010		MOV	P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
8082	041600	116565	000007	000017		MOVB	P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
8083	041606	042765	165777	000016		BIC	#^C<CFMT!CDT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT DRIVE TYPE
8084							; AND FORMAT
8085	041614	112765	000125	000016		MOVB	#RDHEAD,P.CS1(R5) ;STORE READ HEADER COMMAND
8086	041622	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
8087	041630	001027				BNE	34\$;YES, INDICATE ILLEGAL DRIVER COMMAND
8088	041632	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
8089	041640	000137	042062			JMP	C.RTRN ;RESTORE REGISTERS
8090							
8091	041644	122765	000176	000001	30\$:	CMPB	#CONCLR,P.CMND(R5) ;CHECK IF CONTROLLER CLEAR
8092	041652	001012				BNE	32\$;NO, CHECK IF SUBSYSTEM CLEAR
8093	041654	004037	037156			JSR	RO,I.CCLR ;CLEAR CONTROLLER
8094	041660	042062				C.RTRN	;ERROR RETURN
8095	041662	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
8096	041670	001472				BEQ	40\$;NO, INDICATE NORMAL RETURN
8097	041672	005062	000000			CLR	RKCS1(R2) ;RESET INTERRUPT ENABLE
8098	041676	000467				BR	40\$;INDICATE NORMAL RETURN
8099							
8100	041700	122765	000177	000001	32\$:	CMPB	#SUBCLR,P.CMND(R5) ;CHECK IF SUBSYSTEM CLEAR
8101	041706	001406				BEQ	36\$;YES, CLEAR SUBSYSTEM
8102	041710	052737	000100	003062	34\$:	BIS	#E.ILLD,E.CONT ;SET ILLEGAL DRIVER COMMAND
8103	041716	004737	040166			JSR	PC,R.CONT ;REPORT ERROR
8104	041722	000457				BR	C.RTRN ;RESTORE REGISTERS
8105							
8106	041724	012762	000040	000010	36\$:	MOV	#SCLR,RKCS2(R2) ;ISSUE SUBSYSTEM CLEAR
8107	041732	016265	000000	000016		MOV	RKCS1(R2),P.CS1(R5) ;STORE COMMAND AND STATUS REGISTER 1
8108	041740	032765	100000	000016		BIT	#CERR,P.CS1(R5) ;CLEAR IF CONTROLLER ERROR RESET
8109	041746	001406				BEQ	37\$;NO, FINISH COMMAND
8110	041750	052737	000001	003062		BIS	#BITO,E.CONT ;SET CLEAR SUBSYSTEM DID NOT CLEAR
8111							; CONTROLLER ERROR
8112	041756	004737	040166			JSR	PC,R.CONT ;REPORT ERROR
8113	041762	000437				BR	C.RTRN ;RESTORE REGISTERS
8114							
8115	041764	013746	003070		37\$:	MOV	W.MILI,-(SP) ;LOAD 16 MILI-SECOND COUNT FOR ATTENTION
8116							; TO DISAPPEAR
8117	041770	016265	000000	000016	38\$:	MOV	RKCS1(R2),P.CS1(R5) ;STORE CS1

```

8118 041776 032765 040000 000016 BIT #DI,P.CS1(R5) ;CHECK IF ATTENTIONS CLEARED
8119 042004 001411 BEQ 39$ ;YES, FINISH COMMAND
8120 042006 005316 DEC (SP) ;DECREMENT 16 MILLISECOND COUNT
8121 042010 001367 BNE 38$ ;CHECK DRIVE INTERRUPT AGAIN
8122 042012 005726 TST (SP)+ ;ADJUST STACK
8123 042014 052737 000040 003062 BIS #E.SCLR,E.CONT ;SET SUBSYSTEM CLEAR DID NOT CLEAR
8124 ; DRIVE ATTENTIONS
8125 042022 004737 040166 JSR PC,R.CONT ;REPORT ERROR
8126 042026 000415 BR C.RTRN ;RESTORE REGISTER
8127
8128 042030 005726 39$: TST (SP)+ ;ADJUST STACK
8129 042032 032765 000400 000014 BIT #NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
8130 042040 001010 BNE C.RTRN ;YES, RESTORE REGISTERS
8131 042042 112737 177777 003104 MOVB #-1,I.ISRL ;SET INTERRUPT ENABLE SET
8132 042050 012762 000100 000000 MOV #IE,RKCS1(R2) ;SET INTERRUPT ENABLE
8133 042056 004737 040154 40$: JSR PC,R.NORM ;INDICATE NORMAL TERMINATION
8134
8135 042062 012637 177776 C.RTRN: MOV (SP)+,PS ;RESTORE PSW
8136 042066 012600 MOV (SP)+,R0 ;RESTORE R0
8137 042070 012601 MOV (SP)+,R1 ;RESTORE R1
8138 042072 012602 MOV (SP)+,R2 ;RESTORE R2
8139 042074 012603 MOV (SP)+,R3 ;RESTORE R3
8140 042076 012604 MOV (SP)+,R4 ;RESTORE R4
8141 042100 012605 MOV (SP)+,R5 ;RESTORE R5
8142 042102 000207 RTS PC ;RETURN
8143 .SBTTL OCTAL TO BINARY CONVERSION ROUTINE
8144
8145 :*****
8146 :*
8147 :* THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
8148 :* WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL
8149 :* IT WILL GENERATE TWO BINARY WORDS PLACING THE LOW 16 BITS
8150 :* ON THE STACK AND THE HIGH 16 BITS IN LOCATION $HIOCT.
8151 :*
8152 :*CALL
8153 :* MOV <ADDRESS OF ASCII STRING>,-(SP)
8154 :* JSR PC,OCTBIN
8155 :* <ADDRESS OF ERROR RETURN>
8156 :* RETURN
8157 :*
8158 :*****
8159
8160 OCTBIN: MOV R0,-(SP) ;SAVE R0
8161 042106 010146 MOV R1,-(SP) ;SAVE R1
8162 042110 010246 MOV R2,-(SP) ;SAVE R2
8163 042112 016600 000010 MOV 10(SP),R0 ;GET ADDRESS OF ASCII STRING
8164 042116 005001 CLR R1 ;CLEAR DATA WORDS
8165 042120 005002 CLR R2
8166 042122 112046 2$: MOV (R0)+,-(SP) ;PICK THIS CHARACTER
8167 042124 001423 BEQ 3$ ;IF ZERO GET OUT
8168 042126 121627 000054 CMPB (SP),#', ;CHECK IF COMMA
8169 042132 001420 BEQ 3$ ;IF COMMA GET OUT
8170 042134 122716 000060 CMPB #'0,(SP) ;MAKE SURE THIS CHARACTER IS
8171 042140 003030 BGT 4$ ; AN OCTAL DIGIT
8172 042142 122716 000067 CMPB #'7,(SP)
8173 042146 002425 BLT 4$
    
```



```

8174 042150 006301          ASL      R1          ; *2
8175 042152 006102          ROL      R2
8176 042154 006301          ASL      R1          ; *4
8177 042156 006102          ROL      R2
8178 042160 006301          ASL      R1          ; *8
8179 042162 006102          ROL      R2
8180 042164 042716 177770  BIC      #'C7,(SP)   ;STRIP THE ASCII JUNK
8181 042170 062601          ADD      (SP)+,R1    ;ADD THIS DIGIT
8182 042172 000753          BR       2$          ;LOOP
8183 042174 005726          3$:     TST      (SP)+   ;CLEAN PARTIAL FROM STACK
8184 042176 010166 000010  MOV      R1,10(SP)   ;SAVE RESULT
8185 042202 010237 042236  MOV      R2,$SHIOCT
8186 042206 012602          MOV      (SP)+,R2    ;RESTORE R2
8187 042210 012601          MOV      (SP)+,R1    ;RESTORE R1
8188 042212 012600          MOV      (SP)+,R0    ;RESTORE R0
8189 042214 062716 000002  ADD      #2,(SP)     ;ADJUST RETURN
8190 042220 000207          RTS      PC          ;RETURN
8191
8192 042222 005726          4$:     TST      (SP)+   ;CLEAN UP PARTIAL FROM STACK
8193 042224 012602          MOV      (SP)+,R2    ;RESTORE R2
8194 042226 012601          MOV      (SP)+,R1    ;RESTORE R1
8195 042230 012600          MOV      (SP)+,R0    ;RESTORE R0
8196 042232 013616          MOV      @ (SP)+,(SP) ;PUT ADDRESS OF ERROR ROUTINE ON STACK
8197 042234 000207          RTS      PC          ;GO PROCESS ERROR
8198 042236 000000          $SHIOCT: .WORD    0   ;HIGH ORDER BITS GO HERE
8199
8200          .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
8201
8202          ;*****
8203          ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
8204          ;*WITH A RANGE OF 0 TO 2(+33)-1.
8205          ;*CALL:
8206          ;*      JSR      PC,$RAND      ;;CALL THE ROUTINE
8207          ;*      RETURN                    ;;RETURN HERE THE RANDOM
8208          ;*                                ;;NUMBER WILL BE IN
8209          ;*                                ;;$HINUM,$LONUM
8210
8211 042240          $RAND:
8212 042240 010046          MOV      R0,-(SP)    ;;PUSH R0 ON STACK
8213 042242 010146          MOV      R1,-(SP)    ;;PUSH R1 ON STACK
8214 042244 010246          MOV      R2,-(SP)    ;;PUSH R2 ON STACK
8215 042246 013700 042340  MOV      $LONUM,R0    ;;SET R0 WITH LOW
8216 042252 013701 042336  MOV      $HINUM,R1    ;;SET R1 WITH HIGH
8217 042256 012702 177771  MOV      #-7,R2      ;;SET SHIFT COUNT
8218 042262 006300          1$:     ASL      R0          ;;SHIFT R0 LEFT AND
8219 042264 006101          ROL      R1          ;;ROTATE CARRY INTO R1 AND
8220 042266 005202          INC      R2          ;;CHECK FOR DONE
8221 042270 001374          BNE     1$          ;;CONTINUE SHIFT LOOP
8222 042272 063700 042340  ADD      $LONUM,R0    ;;ADD NUMBER TO MAKE X 129
8223 042276 005501          ADC      R1          ;;PROPOGATE CARRY
8224 042300 063701 042336  ADD      $HINUM,R1    ;;ADD NUMBER TO MAKE X 129
8225 042304 062700 001057  ADD      #1057,R0     ;;ADD LOW CONSTANT
8226 042310 005501          ADC      R1          ;;PROPOGATE CARRY
8227 042312 062701 047401  ADD      #47401,R1    ;;ADD HIGH CONSTANT
8228 042316 010037 042340  MOV      R0,$LONUM   ;;SAVE R0
8229 042322 010137 042336  MOV      R1,$HINUM   ;;SAVE R1
8230 042326 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
  
```



```

8286
8287 042510 105366 000001 7$:   DECB   1(SP)      ;;AND THE NULL CHAR.
8288 042514 002770          BLT    6$          ;;DOES A NULL NEED TO BE TYPED?
8289 042516 004737 042554   JSR   PC,$TYPEC   ;;BR IF NO--GO POP THE NULL OFF OF STACK
8290 042522 105337 042620   DECB  $CHARCNT   ;;GO TYPE A NULL
8291 042526 000770          BR     7$          ;;DO NOT COUNT AS A COUNT
                        ;;LOOP
8292
8293           ;HORIZONTAL TAB PROCESSOR
8294
8295 042530 112716 000040 8$:   MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
8296 042534 004737 042554 9$:   JSR   PC,$TYPEC   ;;TYPE A SPACE
8297 042540 132737 000007 042620 BITB  #7,$CHARCNT   ;;BRANCH IF NOT AT
8298 042546 001372          BNE   9$          ;;TAB STOP
8299 042550 005726          TST   (SP)+       ;;POP SPACE OFF STACK
8300 042552 000724          BR     2$          ;;GET NEXT CHARACTER
8301 042554 105777 136370 $TYPEC: TSTB  @STPS      ;;WAIT UNTIL PRINTER IS READY
8302 042560 100375          BPL  $TYPEC
8303 042562 116677 000002 136362 MOVB  2(SP),@STPB   ;;LOAD CHAR TO BE TYPED INTO DATA REG.
8304 042570 122766 000015 000002 CMPB  #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
8305 042576 001003          BNE  1$          ;;BRANCH IF NO
8306 042600 105037 042620   CLRB  $CHARCNT   ;;YES--CLEAR CHARACTER COUNT
8307 042604 000406          BR     $TYPEX
8308 042606 122766 000012 000002 1$:   CMPB  #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
8309 042614 001402          BEQ  $TYPEX      ;;BRANCH IF YES
8310 042616 105227          INCB  (PC)+      ;;COUNT THE CHARACTER
8311 042620 000000          $CHARCNT: WORD 0  ;;CHARACTER COUNT STORAGE
8312 042622 000207          $TYPEX: RTS    PC
8313
8314
8315
8316
8317
8318           ;*****
8319           .SBTTL  DOUBLE-PRECISION MULTIPLY SUBROUTINE
8320           ;*      SUBROUTINE TO MULTIPLY TWO DOUBLE PRECISION INTEGERS
8321           ;*      USES ALL REGISTERS (R0-R5)
8322           ;*
8323           ;*      ENTER WITH JSR PC,M.DPIM
8324           ;*      MULTIPLIER IN R2-R3
8325           ;*      MULTIPLICAND IN R4-R5
8326           ;*      PRODUCT RETURNED IN R0-R1-R2-R3
8327           ;*****
8328 042624 005000 M.DPIM: CLR    R0      ;CLEAR HI ORDER WORDS
8329 042626 005001      CLR    R1
8330 042630 012746 000041      MOV    #41,-(SP)  ;MOVE 33 (DEC) TO COUNTER
8331 042634 006000 M.DP01: ROR    R0
8332 042636 006001      ROR    R1
8333 042640 006002      ROR    R2          ;SHIFT TO ADD
8334 042642 006003      ROR    R3
8335 042644 103003      BCC   M.DP02     ;NO CARRY NO ADD
8336 042646 060501      ADD   R5,R1
8337 042650 005500      ADC   R0          ;ADD DOUBLE PRECISION TO OBTAIN NEW PARTIAL
8338 042652 060400      ADD   R4,R0      ;PRODUCT
8339 042654 005316 M.DP02: DEC   @SP   ;DECREMENT COUNTER
8340 042656 001366      BNE  M.DP01
8341 042660 005726      TST   (SP)+      ;REMOVE THE COUNTER
  
```

8342 042662 000207

RTS PC

8343
8344
8345
8346
8347
8348
8349
8350
8351
8352
8353
8354
8355
8356
8357
8358
8359
8360
8361
8362
8363
8364
8365
8366
8367
8368
8369
8370
8371
8372
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

042664 012746 000040
042670 010446
042672 010546
042674 005466 000002
042700 005416
042702 005666 000002
042706 061601
042710 005500
042712 066600 000002
042716 103445
042720 005046
042722 006103
042724 006102
042726 006101
042730 006100
042732 005716
042734 001410
042736 005016
042740 066601 000002
042744 005500
042746 005516
042750 066600 000004
042754 000404
042756 060501
042760 005500
042762 005516
042764 060400
042766 005516
042770 005716
042772 001401
042774 005203
042776 005366 000006
043002 003347
043004 006003
043006 103404
043010 060501
043012 005500
043014 060400
043016 000241
043020 006103
043022 062706 000010

```

:*****
:SBTTL DOUBLE-PRECISION DIVIDE SUBROUTINE
:*      SUBROUTINE TO PERFORM DOUBLE-PRECISION INTEGER DIVISION
:*      USES ALL REGISTERS (R0-R5)
:*
:*      ENTER WITH JSR PC,M.DPID
:*      DIVIDEND IN R0-R1-R2-R3
:*      DIVISOR IN R4-R5
:*      REMAINDER RETURNED IN R0-R1
:*      QUOTIENT RETURNED IN R2-R3
:*****
M.DPID: MOV #40,-(SP) ;COUNTER FOR DIVISION CYCLES
        MOV R4,-(SP) ;HI ORDER
        MOV R5,-(SP) ;LO ORDER DIVISOR TO THE STACK
        NEG 2(SP) ;FORM NEGATIVE
        NEG @SP ; VERSION OF THE DIVISOR
        SBC 2(SP)
        ADD @SP,R1
        ADC R0 ;PERFORM THE INITIAL SUBTRACTION
        ADD 2(SP),R0
        BCS M.DP50 ;IF CARRY THEN OVERFLOW HAS OCCURRED
        CLR -(SP) ;THIS IS A LONGER LASTING CARRY BIT
M.DP40: ROL R3
        ROL R2
        ROL R1
        ROL R0
        TST @SP ;TEST "CARRY INDICATOR"
        BEQ M.DP41 ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
        CLR @SP ;CLEAR UP FOR NEXT TIME
        ADD 2(SP),R1
        ADC R0 ;ADD -(DIVISOR)
        ADC @SP ;SET "CARRY"
        ADD 4(SP),R0 ;<
        BR M.DP42
M.DP41: ADD R5,R1
        ADC R0 ;ADD +(DIVISOR)
        ADC @SP ;SET "CARRY"
        ADD R4,R0 ;<
M.DP42: ADC @SP ;SET "CARRY"
        TST @SP ;TEST THE UPDATE INDICATOR
        BEQ .+4 ;> ;IF ZERO FORGET IT
        INC R3 ;> ;NO CARRY POSSIBLE HERE
        DEC 6(SP) ;< ;DECREMENT COUNTER
        BGT M.DP40 ;BR IF MORE TO DO
        ROR R3
        BCS M.DP44
        ADD R5,R1
        ADC R0
        ADD R4,R0
        CLC
M.DP44: ROL R3
        ADD #10,SP ;ADJUST STACK BY 4 WORDS
    
```

8398 043026 000242
 8399 043030 000207
 8400 043032 062706 000006
 8401 043036 000262
 8402 043040 000207
 8403
 8404
 8405
 8406
 8407
 8408
 8409
 8410
 8411
 8412
 8413
 8414
 8415
 8416

CLV
 RTS PC
 M.DP50: ADD #6,SP
 SEV
 RTS PC

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

 *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,@#\$DB20 ;; CALL THE ROUTINE
 * RETURN ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

8417 043042 104407
 8418 043044 016601 000002
 8419 043050 012705 043161
 8420 043054 012704 000014
 8421 043060 012703 177770
 8422 043064 012100
 8423 043066 012101
 8424 043070 005002
 8425 043072 110245
 8426 043074 010002
 8427 043076 005304
 8428 043100 003007
 8429 043102 001405
 8430 043104 005205
 8431 043106 010566 000002
 8432 043112 104410
 8433 043114 000207
 8434 043116 006203
 8435 043120 006001
 8436 043122 006000
 8437 043124 006001
 8438 043126 006000
 8439 043130 006001
 8440 043132 006000
 8441 043134 040302
 8442 043136 062702 000060
 8443 043142 000753
 8444 043144 000016
 8445
 8446
 8447
 8448
 8449
 8450
 8451
 8452
 8453

\$DB20: SAVREG ;; SAVE ALL REGISTERS
 MOV 2(SP),R1 ;; PICKUP THE POINTER TO LOW WORD
 MOV #SOCTVL+13.,R5 ;; POINTER TO DATA TABLE
 MOV #12.,R4 ;; DO ELEVEN CHARACTERS
 MOV #^C7,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
 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
 \$SOCTVL: .BLKB 14. ;; RESERVE DATA TABLE

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

 *THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
 *DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
 *POSITIVE.
 *CALL

* MOV #PNTR,-(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
 * JSR PC,@#\$DB20

```

8454 ;* RETURN ;:THE FIRST ADDRESS OF ASCIIZ
8455 ;:IS ON THE STACK
8456
8457
8458 043162 104407 $DB2D: SAVREG ;:SAVE REGISTERS
8459 043164 016602 000002 MOV 2(SP),R2 ;:PICKUP THE DATA POINTER
8460 043170 012700 043342 MOV #$DECVL,R0 ;:GET ADDRESS OF "$DECVL" STRING
8461 043174 010066 000002 MOV R0,2(SP) ;:PUT ADDRESS OF ASCIIZ STRING ON STACK
8462 043200 012201 MOV (R2)+,R1 ;:PICKUP THE BINARY NUMBER
8463 043202 012202 MOV (R2)+,R2
8464 043204 012737 000012 043260 MOV #10,,4$ ;:SET UP TO DO 10 CONVERSIONS
8465 043212 012704 043272 MOV #$STNPWR,R4 ;:ADDRESS OF TEN POWER
8466 043216 012705 043274 MOV #$STNPWR+2,R5
8467 043222 005003 1$: CLR R3 ;:CLEAR PARTIAL
8468 043224 161401 2$: SUB (R4),R1 ;:SUBTRACT TEN POWER
8469 043226 005602 SBC R2
8470 043230 161502 SUB (R5),R2
8471 043232 002402 BLT 3$ ;:BR IF TEN POWER TO LARGE
8472 043234 005203 INC R3 ;:ADD 1 TO PARTIAL
8473 043236 000772 BR 2$ ;:LOOP
8474 043240 062401 3$: ADD (R4)+,R1 ;:RESTORE SUBTRACTED VALUE
8475 043242 005502 ADC R2
8476 043244 062402 ADD (R4)+,R2
8477 043246 022525 CMP (R5)+,(R5)+ ;:MOVE TO NEXT TEN POWER
8478 043250 052703 000060 BIS #'0,R3 ;:CHANGE PARTIAL TO ASCII
8479 043254 110320 MOVB R3,(R0)+ ;:SAVE IT
8480 043256 005327 DEC (PC)+ ;:DONE?
8481 043260 000000 4$: .WORD 0
8482 043262 001357 BNE 1$ ;:BR IF NO
8483 043264 105020 CLRB (R0)+ ;:TERMINATOR
8484 043266 104410 RESREG ;:RESTORE REGISTERS
8485 043270 000207 RTS PC ;:RETURN
8486 043272 145000 $STNPWR: 145000 ;:1.0E09
8487 043274 035632 35632
8488 043276 160400 ;:1.0E08
8489 043300 002765 2765
8490 043302 113200 ;:1.0E07
8491 043304 000230 230
8492 043306 041100 ;:1.0E06
8493 043310 000017 17
8494 043312 103240 ;:1.0E05
8495 043314 000001 1
8496 043316 023420 ;:1.0E04
8497 043320 000000 0
8498 043322 001750 ;:1.0E03
8499 043324 000000 0
8500 043326 000144 ;:1.0E02
8501 043330 000000 0
8502 043332 000012 ;:1.0E01
8503 043334 000000 0
8504 043336 000001 ;:1.0E00
8505 043340 000000 0
8506 043342 000014 $DECVL: .BLKB 12. ;:RESERVE STORAGE FOR ASCIIZ STRING
8507 .SBTTL TYPE NUMERICAL ASCIIZ STRING SUPPRESS LEADING ZEROS
8508
8509 ;:*****

```

```

8510 ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
8511 ;*LEADING NUMBERS.
8512 ;*CALL
8513 ;*      MOV      #NUMADR,-(SP)      ;;FIRST ADDRESS OF ASCIZ STRING
8514 ;*      JSR      PC,@#$$SUPRS
8515
8516
8517 043356 010046 000004 $$SUPRS: MOV      R0,-(SP)      ;;SAVE R0
8518 043360 016600 000004      MOV      4(SP),R0      ;;PICKUP THE POINTER
8519 043364 105710      TSTB      (R0)      ;;TERMINATEOR?
8520 043366 001403      BEQ      2$      ;;BR IF YES
8521 043370 122720 000060      CMPB      #'0,(R0)+      ;;IS THIS AN ASCII '0' ?
8522 043374 001773      BEQ      1$      ;;BR IF YES
8523 043376 005300      DEC      R0      ;;BACKUP BY '1'
8524 043400 010037 043406      MOV      R0,3$      ;;SAVE FOR TYPING
8525 043404 104401      TYPE      ;;GO TYPE
8526 043406 000000      .WORD      0      ;;ASCIZ POINTER GOES HERE
8527 043410 012600      MOV      (SP)+,R0      ;;RESTORE R0
8528 043412 012616      MOV      (SP)+,(SP)      ;;RESTORE THE STACK
8529 043414 000207      RTS      PC      ;;RETURN
8530 .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

```

```

8531
8532 ;*****
8533 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
8534 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
8535 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
8536 ;*CALL:
8537 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
8538 ;*      TYPOS      ;;CALL FOR TYPEOUT
8539 ;*      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
8540 ;*      .BYTE      M      ;;M=1 OR 0
8541 ;*      ;;1=TYPE LEADING ZEROS
8542 ;*      ;;0=SUPPRESS LEADING ZEROS
8543
8544 ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
8545 ;*$TYPOS OR $TYPOC
8546 ;*CALL:
8547 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
8548 ;*      TYPON      ;;CALL FOR TYPEOUT
8549
8550 ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
8551 ;*CALL:
8552 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
8553 ;*      TYPOC      ;;CALL FOR TYPEOUT
8554
8555 043416 017646 000000 043641 $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
8556 043422 116637 000001      MOVB      1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
8557 043430 112637 043643      MOVB      (SP)+,$SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
8558 043434 062716 000002      ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
8559 043440 000406      BR      $TYPON
8560 043442 112737 000001 043641 $TYPOC: MOVB      #1,$OFILL      ;;SET THE ZERO FILL SWITCH
8561 043450 112737 000006 043643      MOVB      #6,$SOMODE+1      ;;SET FOR SIX(6) DIGITS
8562 043456 112737 000005 043640 $TYPON: MOVB      #5,$SOCNT      ;;SET THE ITERATION COUNT
8563 043464 010346      MOV      R3,-(SP)      ;;SAVE R3
8564 043466 010446      MOV      R4,-(SP)      ;;SAVE R4
8565 043470 010546      MOV      R5,-(SP)      ;;SAVE R5

```

```

8566 043472 113704 043643          MOVB    $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
8567 043476 005404                   NEG     R4
8568 043500 062704 000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
8569 043504 110437 043642          MOVB    R4,$OMODE      ;;SAVE IT FOR USE
8570 043510 113704 043641          MOVB    $OFILL,R4      ;;GET THE ZERO FILL SWITCH
8571 043514 016605 000012          MCV     12(SP),R5      ;;PICKUP THE INPUT NUMBER
8572 043520 005003                   CLR     R3              ;;CLEAR THE OUTPUT WORD
8573 043522 006105                   1$:    ROL     R5        ;;ROTATE MSB INTO 'C'
8574 043524 000404                   BR      3$              ;;GO DO MSB
8575 043526 006105                   2$:    ROL     R5        ;;FORM THIS DIGIT
8576 043530 006105                   ROL     R5
8577 043532 006105                   ROL     R5
8578 043534 010503                   MOV     R5,R3
8579 043536 006103                   3$:    ROL     R3        ;;GET LSB OF THIS DIGIT
8580 043540 105337 043642          DECB    $OMODE          ;;TYPE THIS DIGIT?
8581 043544 100016                   BPL     7$              ;;BR IF NO
8582 043546 042703 177770          BIC     #177770,R3     ;;GET RID OF JUNK
8583 043552 001002                   BNE     4$              ;;TEST FOR 0
8584 043554 005704                   TST     R4              ;;SUPPRESS THIS 0?
8585 043556 001403                   BEQ     5$              ;;BR IF YES
8586 043560 005204                   4$:    INC     R4        ;;DON'T SUPPRESS ANYMORE 0'S
8587 043562 052703 000060          BIS     #'0,R3         ;;MAKE THIS DIGIT ASCII
8588 043566 052703 000040          5$:    BIS     #' ,R3   ;;MAKE ASCII IF NOT ALREADY
8589 043572 110337 043636          MOVB    R3,8$          ;;SAVE FOR TYPING
8590 043576 104401 043636          TYPE   8$              ;;GO TYPE THIS DIGIT
8591 043602 105337 043640          7$:    DECB    $OCNT     ;;COUNT BY 1
8592 043606 003347                   BGT     2$              ;;BR IF MORE TO DO
8593 043610 002402                   BLT     6$              ;;BR IF DONE
8594 043612 005204                   INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
8595 043614 000744                   BR      2$              ;;GO DO THE LAST DIGIT
8596 043616 012605                   6$:    MOV     (SP)+,R5   ;;RESTORE R5
8597 043620 012604                   MOV     (SP)+,R4       ;;RESTORE R4
8598 043622 012603                   MOV     (SP)+,R3       ;;RESTORE R3
8599 043624 016666 000002 000004   MOV     2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
8600 043632 012616                   MOV     (SP)+,(SP)
8601 043634 000002                   RTI
8602 043636 000                   8$:    .BYTE   0         ;;RETURN
8603 043637 000                   .BYTE   0         ;;STORAGE FOR ASCII DIGIT
8604 043640 000                   $OCNT:  .BYTE   0         ;;TERMINATOR FOR TYPE ROUTINE
8605 043641 000                   $OFILL: .BYTE   0         ;;OCTAL DIGIT COUNTER
8606 043642 000000                   $OMODE: .WORD   0         ;;ZERO FILL SWITCH
8607                                     .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
8608                                     ;;NUMBER OF DIGITS TO TYPE
8609                                     *****
8610                                     ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
8611                                     ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
8612                                     ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
8613                                     ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
8614                                     ;;*REPLACED WITH SPACES.
8615                                     ;;*CALL:
8616                                     ;;*   MOV     NUM,-(SP)    ;;PUT THE BINARY NUMBER ON THE STACK
8617                                     ;;*   TYPDS                    ;;GO TO THE ROUTINE
8618
8619 043644                                     $TYPDS:
8620 043644 010046                   MOV     R0,-(SP)       ;;PUSH R0 ON STACK
8621 043646 010146                   MOV     R1,-(SP)       ;;PUSH R1 ON STACK

```



```

8622 043650 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
8623 043652 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
8624 043654 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
8625 043656 012746 020200  MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
8626 043662 016605 000020  MOV      20(SP),R5    ;;GET THE INPUT NUMBER
8627 043666 100004      BPL      1$           ;;BR IF INPUT IS POS.
8628 043670 005405      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
8629 043672 112766 000055 000001  MOVVB   #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
8630 043700 005000 1$:      CLR      R0           ;;ZERO THE CONSTANTS INDEX
8631 043702 012703 044060  MOV      #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
8632 043706 112723 000040  MOVVB   #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
8633 043712 005002 2$:      CLR      R2           ;;CLEAR THE BCD NUMBER
8634 043714 016001 044050  MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
8635 043720 160105 3$:      SUB      R1,R5        ;;FORM THIS BCD DIGIT
8636 043722 002402      BLT      4$           ;;BR IF DONE
8637 043724 005202      INC      R2           ;;INCREASE THE BCD DIGIT BY 1
8638 043726 000774      BR       3$
8639 043730 060105 4$:      ADD      R1,R5        ;;ADD BACK THE CONSTANT
8640 043732 005702      TST      R2           ;;CHECK IF BCD DIGIT=0
8641 043734 001002      BNE      5$           ;;FALL THROUGH IF 0
8642 043736 105716      TSTB    (SP)         ;;STILL DOING LEADING 0'S?
8643 043740 100407      BMI      7$           ;;BR IF YES
8644 043742 106316 5$:      ASLB    (SP)         ;;MSD?
8645 043744 103003      BCC      6$           ;;BR IF NO
8646 043746 116663 000001 177777  MOVVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
8647 043754 052702 000060 6$:      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
8648 043760 052702 000040 7$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
8649 043764 110223      MOVVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
8650 043766 005720      TST      (R0)+       ;;JUST INCREMENTING
8651 043770 020027 000010  CMP      R0,#10     ;;CHECK THE TABLE INDEX
8652 043774 002746      BLT      2$           ;;GO DO THE NEXT DIGIT
8653 043776 003002      BGT      8$           ;;GO TO EXIT
8654 044000 010502      MOV      R5,R2       ;;GET THE LSD
8655 044002 000764      BR       6$           ;;GO CHANGE TO ASCII
8656 044004 105726 8$:      TSTB    (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
8657 044006 100003      BPL      9$           ;;BR IF NO
8658 044010 116663 177777 177776  MOVVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
8659 044016 105013 9$:      CLRB    (R3)        ;;SET THE TERMINATOR
8660 044020 012605      MOV      (SP)+,R5    ;;POP STACK INTO R5
8661 044022 012603      MOV      (SP)+,R3    ;;POP STACK INTO R3
8662 044024 012602      MOV      (SP)+,R2    ;;POP STACK INTO R2
8663 044026 012601      MOV      (SP)+,R1    ;;POP STACK INTO R1
8664 044030 012600      MOV      (SP)+,R0    ;;POP STACK INTO R0
8665 044032 104401 044060  TYPE    $DBLK        ;;NOW TYPE THE NUMBER
8666 044036 016666 000002 000004  MOV      2(SP),4(SP) ;;ADJUST THE STACK
8667 044044 012616      MOV      (SP)+,(SP)
8668 044046 000002      RTI                    ;;RETURN TO USER
8669 044050 023420  $DTBL: 10000.
8670 044052 001750      1000.
8671 044054 000144      100.
8672 044056 000012      10.
8673 044060 000004  $DBLK: .BLKW 4
      .SBTTL ERROR HANDLER ROUTINE
8674
8675
8676
8677

```

THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,

```

8678      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
8679      ;*AND GO TO TYPERR ON ERROR
8680      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
8681      ;*SW15=1      HALT ON ERROR
8682      ;*SW13=1      INHIBIT ERROR TYPEOUTS
8683      ;*SW10=1      BELL ON ERROR
8684      ;*SW09=1      LOOP ON ERROR
8685      ;*CALL
8686      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
8687
8688      $ERROR:
8689      044070      105237      001103      7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
8690      044074      001775      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
8691      044076      013777      001102      135036      MOV      $STNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
8692      044104      032777      002000      135026      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
8693      044112      001402      BEQ      1$      ;;NO - SKIP
8694      044114      104401      001320      TYPE      ,SBELL      ;;RING BELL
8695      044120      005237      001112      1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
8696      044124      011637      001116      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
8697      044130      162737      000002      001116      SUB      #2,$ERRPC
8698      044136      117737      134754      001114      MOVB    @$ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
8699      044144      032777      020000      134766      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
8700      044152      001004      BNE      20$      ;;SKIP TYPEOUTS
8701      044154      004737      030330      JSR      PC,TYPERR      ;;GO TO USER ERROR ROUTINE
8702      044160      104401      001325      TYPE      ,SCLRF
8703
8704      044164      122737      000001      001350      20$:      CMPB    #APTENV,$ENV      ;;RUNNING IN APT MODE
8705      044172      001007      BNE      2$      ;;NO,SKIP APT ERROR REPORT
8706      044174      113737      001114      044206      MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
8707      044202      004737      044762      JSR      PC,$ATY4      ;;REPORT FATAL ERROR TO APT
8708      044206      000      21$:      .BYTE  0
8709      044207      000      .BYTE  0
8710      044210      000777      22$:      BR      22$      ;;APT ERROR LOOP
8711      044212      005777      134722      2$:      TST    @SWR      ;;HALT ON ERROR
8712      044216      100001      BPL      3$      ;;SKIP IF CONTINUE
8713      044220      000000      HALT      ;;HALT ON ERROR!
8714      044222      032777      001000      134710      3$:      BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
8715      044230      001402      BEQ      4$      ;;BR IF NO
8716      044232      013716      001110      MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
8717      044236      005737      001316      4$:      TST    $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
8718      044242      001402      BEQ      5$      ;;BR IF NONE
8719      044244      013716      001316      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
8720      044250      5$:
8721      044250      022737      017234      000042      CMP      #$ENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
8722      044256      001001      BNE      6$      ;;BRANCH IF NO
8723      044260      000000      HALT      ;;YES
8724      044262      6$:
8725      044262      000002      RTI      ;;RETURN
8726      .SBTTL  TTY INPUT ROUTINE
8727
8728      ;:*****
8729      .ENABL  LSB
8730
8731      .DSABL  LSB
8732
8733
    
```

```

8734
8735
8736
8737
8738
8739
8740
8741
8742 044264 011646
8743 044266 016666 000004 000002
8744 044274 105777 134644
8745 044300 100375
8746 044302 117766 134640 000004
8747 044310 042766 177600 000004
8748 044316 026627 000004 000023
8749 044324 001013
8750 044326 105777 134612
8751 044332 100375
8752 044334 117746 134606
8753 044340 042716 177600
8754 044344 022627 000021
8755 044350 001366
8756 044352 000750
8757 044354 026627 000004 000140
8758 044362 002407
8759 044364 026627 000004 000175
8760 044372 003003
8761 044374 042766 000040 000004
8762 044402 000002
8763 044404 052536 005015 000
8764 044411 136 006507 000012
8765 044416 005015 053523 020122
8766 044424 020075 000
8767 044427 040 047040 053505
8768 044434 036440 000040
8769
8770
8771
8772
8773
8774
8775 044440 012737 044452 000024
8776 044446 000000
8777 044450 000776
8778
8779
8780 044452 005037 044524
8781 044456 005237 044524
8782 044462 001375
8783 044464 012737 044440 000024
8784 044472 012737 000340 000026
8785 044500 012737 000340 000036
8786 044506 012706 001100
8787 044512 104401 044526
8788 044516 000005
8789 044520 000177 134362

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

$RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
MOV 4(SP),2(SP) ;;SAVE THE PS
1$: TSTB @STKS ;;WAIT FOR
BPL 1$ ;;A CHARACTER
MOVB @STKB,4(SP) ;;READ THE TTY
BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
CMP 4(SP),#23 ;;IS IT A CONTROL-S?
BNE 3$ ;;BRANCH IF NO
2$: TSTB @STKS ;;WAIT FOR A CHARACTER
BPL 2$ ;;LOOP UNTIL ITS THERE
MOVB @STKB,-(SP) ;;GET CHARACTER
BIC #^C177,(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 #40,4(SP) ;;MAKE IT UPPER CASE
4$: RTI ;;GO BACK TO USER
$CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
$CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
$MSWR: .ASCIZ <15><12>/SWR = /
$MNEW: .ASCIZ / NEW = /

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE
$PWRDN: MOV $SPWRUP,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 @SLPADR ;;RESTART THE CURRENT TEST
  
```

```

8790 044524 000000 $PWRCT: .WORD 0
8791 044526 005015 047520 042527 PWRMSG: .ASCIZ <15><12>/POWER FAILED/<15><12>
8792 044534 020122 040506 046111
8793 044542 042105 005015 000
8794 044550 .EVEN
8795
8796
8797
8798
8799
8800
8801
8802 044550 105737 001103 SCOPE1: TSTB $ERFLG ;SEE IF AN ERROR HAS OCCURRED
8803 044554 001406 BEQ 6$ ;BR IF NOT
8804 044556 032777 001000 134354 BIT #BIT9,@SWR ;SEE IF LOOP ON ERROR DESIRED
8805 044564 001402 BEQ 6$ ;BR IF NOT
8806 044566 013716 001110 MOV $LPERR,(SP) ;SET ERROR LOOP ADDRESS ON STACK
8807 044572 000002 6$: RTI ;RETURN
8808
8809
8810
8811 .SBTTL SCOPE HANDLER ROUTINE
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823 044574 $SCOPE:
8824 044574 032777 040000 134336 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
8825 044602 001052 BNE $OVER ;;YES IF SW14=1
8826 :####START OF CODE FOR THE XOR TESTER####
8827 044604 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
8828 :THIS INSTRUCTION TO A "NOP" (NOP=240)
8829 044606 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
8830 044612 012737 044632 000004 MOV #5$,@#ERRVEC ;;SET FOR TIMEOUT
8831 044620 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
8832 044624 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
8833 044630 000421 BR $SVLAD ;;GO TO THE NEXT TEST
8834 044632 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
8835 044634 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
8836 044640 000407 BR 7$ ;;LOOP ON THE PRESENT TEST
8837 044642 6$:;####END OF CODE FOR THE XOR TESTER####
8838 044642 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
8839 044646 001412 BEQ $SVLAD ;;BR IF NO
8840 044650 032777 001000 134262 BIT #BIT09,@SWR ;;LOOP ON ERROR?
8841 044656 001404 BEQ 4$ ;;BR IF NO
8842 044660 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
8843 044666 000420 BR $OVER
8844 044670 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
8845 044674 105237 001102 $SVLAD: INCB $STSTM ;;COUNT TEST NUMBERS
    
```

```

8846 044700 113737 001102 001334      MOV      $STNM,$TESTN      ;;SET TEST NUMBER IN APT MAILBOX
8847 044706 011637 001106      MOV      (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
8848 044712 011637 001110      MOV      (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
8849 044716 005037 001316      CLR      $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
8850 044722 112737 000001 001115      MOV      #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
8851 044730 013777 001102 134204 $OVER:  MOV      $STNM,@DISPLAY  ;;DISPLAY TEST NUMBER
8852 044736 013716 001106      MOV      $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
8853 044742 000002      RTI                      ;;FIXES PS
8854                                     .SBTTL  APT COMMUNICATIONS ROUTINE
8855
8856                                     ;*****
8857 044744 112737 000001 045210 $ATY1:  MOV      #1,$FFLG        ;;TO REPORT FATAL ERROR
8858 044752 112737 000001 045206 $ATY3:  MOV      #1,$MFLG        ;;TO TYPE A MESSAGE
8859 044760 000403      BR                      ;;
8860 044762 112737 000001 045210 $ATY4:  MOV      #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
8861 044770 $ATYC:
8862 044770 010046      MOV      RO,-(SP)         ;;PUSH RO ON STACK
8863 044772 010146      MOV      R1,-(SP)         ;;PUSH R1 ON STACK
8864 044774 105737 045206      TSTB     $MFLG           ;;SHOULD TYPE A MESSAGE?
8865 045000 001450      BEQ      5$              ;;IF NOT: BR
8866 045002 122737 000001 001350 CMPB     #APTENV,$ENV     ;;OPERATING UNDER APT?
8867 045010 001031      BNE      3$              ;;IF NOT: BR
8868 045012 132737 000100 001351 BITB     #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
8869 045020 001425      BEQ      3$              ;;IF NOT: BR
8870 045022 017600 000004      MOV      @4(SP),RO        ;;GET MESSAGE ADDR.
8871 045026 062766 000002 000004 ADD      #2,4(SP)         ;;BUMP RETURN ADDR.
8872 045034 005737 001330 1$:     TST      $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
8873 045040 001375      BNE      1$              ;;IF NOT: WAIT
8874 045042 010037 001344      MOV      RO,$MSGAD        ;;PUT ADDR IN MAILBOX
8875 045046 105720 2$:     TSTB     (RO)+           ;;FIND END OF MESSAGE
8876 045050 001376      BNE      2$              ;;
8877 045052 163700 001344      SUB      $MSGAD,RO        ;;SUB START OF MESSAGE
8878 045056 006200      ASR      RO              ;;GET MESSAGE LNTH IN WORDS
8879 045060 010037 001346      MOV      RO,$MSGGLT        ;;PUT LENGTH IN MAILBOX
8880 045064 012737 000004 001330 MOV      #4,$MSGTYPE      ;;TELL APT TO TAKE MSG.
8881 045072 000413      BR      5$              ;;
8882 045074 017637 000004 045120 3$:     MOV      @4(SP),4$        ;;PUT MSG ADDR IN JSR LINKAGE
8883 045102 062766 000002 000004 ADD      #2,4(SP)         ;;BUMP RETURN ADDRESS
8884 045110 013746 177776      MOV      177776,-(SP)     ;;PUSH 177776 ON STACK
8885 045114 004737 042342      JSR      PC,$TYPE        ;;CALL TYPE MACRO
8886 045120 000000 4$:     .WORD    0
8887 045122 5$:
8888 045122 105737 045210 10$:    TSTB     $FFLG           ;;SHOULD REPORT FATAL ERROR?
8889 045126 001416      BEQ      12$            ;;IF NOT: BR
8890 045130 005737 001350      TST      $ENV            ;;RUNNING UNDER APT?
8891 045134 001413      BEQ      12$            ;;IF NOT: BR
8892 045136 005737 001330 11$:    TST      $MSGTYPE        ;;FINISHED LAST MESSAGE?
8893 045142 001375      BNE      11$            ;;IF NOT: WAIT
8894 045144 017637 000004 001332 MOV      @4(SP),$FATAL    ;;GET ERROR #
8895 045152 062766 000002 000004 ADD      #2,4(SP)         ;;BUMP RETURN ADDR.
8896 045160 005237 001330      INC      $MSGTYPE        ;;TELL APT TO TAKE ERROR
8897 045164 105037 045210 12$:    CLRB     $FFLG           ;;CLEAR FATAL FLAG
8898 045170 105037 045207      CLRB     $LFLG           ;;CLEAR LOG FLAG
8899 045174 105037 045206      CLRB     $MFLG           ;;CLEAR MESSAGE FLAG
8900 045200 012601      MOV      (SP)+,R1        ;;POP STACK INTO R1
8901 045202 012600      MOV      (SP)+,RO        ;;POP STACK INTO RO
    
```

```

8902 045204 000207
8903 045206 000
8904 045207 000
8905 045210 000
8906 045212
8907 000200
8908 000001
8909 000100
8910 000040
8911
8912
8913
8914
8915
8916
8917
8918
8919
8920
8921
8922
8923
8924
8925
8926
8927
8928 045212
8929 045212 010046
8930 045214 010146
8931 045216 010246
8932 045220 010346
8933 045222 010446
8934 045224 010546
8935 045226 016646 000022
8936 045232 016646 000022
8937 045236 016646 000022
8938 045242 016646 000022
8939 045246 000002
8940
8941
8942
8943
8944 045250
8945 045250 012666 000022
8946 045254 012666 000022
8947 045260 012666 000022
8948 045264 012666 000022
8949 045270 012605
8950 045272 012604
8951 045274 012603
8952 045276 012602
8953 045300 012601
8954 045302 012600
8955 045304 000002
8956
8957
  
```

```

      RTS      PC      ;;RETURN
$MFLG: .BYTE  0      ;;MESSG. FLAG
$LFLG: .BYTE  0      ;;LOG FLAG
$FFLG: .BYTE  0      ;;FATAL FLAG
      .EVEN
APTSIZE=200
APTENV=001
APTSPool=100
APTCSUP=040
.SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

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

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

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

```

8958
8959
8960
8961
8962
8963
8964 045306 010046
8965 045310 016600 000002
8966 045314 005740
8967 045316 111000
8968 045320 006300
8969 045322 016000 045342
8970 045326 000200
8971
8972
8973
8974
8975 045330 011646
8976 045332 016666 000004 000002
8977 045340 000002
8978
8979
8980
8981
8982
8983
8984
8985
8986 045342 045330
8987 045344 042342
8988 045346 043442
8989 045350 043416
8990 045352 043456
8991 045354 043644
8992
8993
8994 045356 044264
8995 045360 045212
8996 045362 045250
8997 045364 044550
8998
8999 045366 020040 020040 042524 1STMSG: .ASCIZ / TEST /
9000 045374 052123 000040
9001 045400 025052 020040 000 AS2SP2: .ASCIZ /** /
9002 045405 125 044516 052502 EM1: .ASCIZ /UNIBUS PARITY ERROR/
9003 045412 020123 040520 044522
9004 045420 054524 042440 051122
9005 045426 051117 000
9006 045431 116 047117 042455 EM2: .ASCIZ /NON-EXISTANT MEMORY/
9007 045436 044530 052123 047101
9008 045444 020124 042515 047515
9009 045452 054522 000
9010 045455 116 047117 042455 EM3: .ASCIZ /NON-EXISTANT DRIVE/
9011 045462 044530 052123 047101
9012 045470 020124 051104 053111
9013 045476 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 R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;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 R0-R5 ROUTINE
$RESREG ;;CALL=RESREG TRAP+10(104410) RESTORE R0-R5 ROUTINE
SCOPE1 ;;CALL=SCOPER TRAP+11(104411) INTERNAL LOOP ON ERROR ROUTINE

```

9014	045500	047125	052111	043040	EM4:	.ASCIZ	/UNIT FIELD ERROR/
9015	045506	042511	042114	042440			
9016	045514	051122	051117	000			
9017	045521	123	041125	054523	EM5:	.ASCIZ	/SUBSYS TIMEOUT/
9018	045526	020123	044524	042515			
9019	045534	052517	000124				
9020	045540	020104	047524	041440	EM6:	.ASCIZ	/D TO C PARITY ERROR/
9021	045546	050040	051101	052111			
9022	045554	020131	051105	047522			
9023	045562	000122					
9024	045564	051104	053111	020105	EM7:	.ASCIZ	/DRIVE DETECTED PARITY ERROR/
9025	045572	042504	042524	052103			
9026	045600	042105	050040	051101			
9027	045606	052111	020131	051105			
9028	045614	047522	000122				
9029	045620	041501	046040	053517	EM10:	.ASCIZ	/AC LOW/
9030	045626	000					
9031	045627	123	042520	042105	EM11:	.ASCIZ	/SPEED LOSS/
9032	045634	046040	051517	000123			
9033	045642	046111	042514	040507	EM12:	.ASCIZ	/ILLEGAL FUNCTION/
9034	045650	020114	052506	041516			
9035	045656	044524	047117	000			
9036	045663	120	047522	051107	EM13:	.ASCIZ	/PROGRAMMING ERROR/
9037	045670	046501	044515	043516			
9038	045676	042440	051122	051117			
9039	045704	000					
9040	045705	116	047117	042455	EM14:	.ASCIZ	/NON-EXISTANT FUNCTION/
9041	045712	044530	052123	047101			
9042	045720	020124	052506	041516			
9043	045726	044524	047117	000			
9044	045733	104	044522	042526	EM15:	.ASCIZ	/DRIVE TYPE ERROR/
9045	045740	052040	050131	020105			
9046	045746	051105	047522	000122			
9047	045754	047506	046522	052101	EM16:	.ASCIZ	/FORMAT ERROR/
9048	045762	042440	051122	051117			
9049	045770	000					
9050	045771	127	044522	042524	EM17:	.ASCIZ	/WRITE LOCK ERROR/
9051	045776	046040	041517	020113			
9052	046004	051105	047522	000122			
9053	046012	051104	053111	020105	EM20:	.ASCIZ	/DRIVE UNSAFE/
9054	046020	047125	040523	042506			
9055	046026	000					
9056	046027	123	042505	020113	EM21:	.ASCIZ	/SEEK INCOMPLETE/
9057	046034	047111	047503	050115			
9058	046042	042514	042524	000			
9059	046047	103	046131	047111	EM22:	.ASCIZ	/CYLINDER OVERFLOW/
9060	046054	042504	020122	053117			
9061	046062	051105	046106	053517			
9062	046070	000					
9063	046071	111	046114	043505	EM23:	.ASCIZ	/ILLEGAL CYLINDER ADDRESS/
9064	046076	046101	041440	046131			
9065	046104	047111	042504	020122			
9066	046112	042101	051104	051505			
9067	046120	000123					
9068	046122	051104	053111	020105	EM24:	.ASCIZ	/DRIVE OFF TRACK/
9069	046130	043117	020106	051124			

9070	046136	041501	000113						
9071	046142	051104	053111	020105	EM25:	.ASCIZ	/DRIVE TIMING ERROR/		
9072	046150	044524	044515	043516					
9073	046156	042440	051122	051117					
9074	046164	000							
9075	046165	104	052101	020101	EM26:	.ASCIZ	/DATA LATE/		
9076	046172	040514	042524	000					
9077	046177	103	047117	051124	EM27:	.ASCIZ	/CONTROLLER TIMEOUT/		
9078	046204	046117	042514	020122					
9079	046212	044524	042515	052517					
9080	046220	000124							
9081	046222	050117	051105	052101	EM30:	.ASCIZ	/OPERATION INCOMPLETE/		
9082	046230	047511	020116	047111					
9083	046236	047503	050115	042514					
9084	046244	042524	000						
9085	046247	110	040505	042504	EM31:	.ASCIZ	/HEADER VRC ERROR/		
9086	046254	020122	051126	020103					
9087	046262	051105	047522	000122					
9088	046270	040504	040524	041440	EM32:	.ASCIZ	/DATA CHECK ERROR/		
9089	046276	042510	045503	042440					
9090	046304	051122	051117	000					
9091	046311	127	044522	042524	EM33:	.ASCIZ	/WRITE CHECK ERROR/		
9092	046316	041440	042510	045503					
9093	046324	042440	051122	051117					
9094	046332	000							
9095	046333	104	052101	020101	EM34:	.ASCIZ	/DATA MISCOMPARE/		
9096	046340	044515	041523	046517					
9097	046346	040520	042522	000					
9098	046353	116	020117	051104	EM35:	.ASCIZ	/NO DRIVE RESPONSE-UFE & NXD/		
9099	046360	053111	020105	042522					
9100	046366	050123	047117	042523					
9101	046374	052455	042506	023040					
9102	046402	047040	042130	000					
9103	046407	104	044522	042526	EM36:	.ASCIZ	/DRIVE ERROR WILL NOT CLEAR/		
9104	046414	042440	051122	051117					
9105	046422	053440	046111	020114					
9106	046430	047516	020124	046103					
9107	046436	040505	000122						
9108	046442	051104	053111	020105	EM37:	.ASCIZ	/DRIVE STATUS CHANGE WILL NOT CLEAR/		
9109	046450	052123	052101	051525					
9110	046456	041440	040510	043516					
9111	046464	020105	044527	046114					
9112	046472	047040	052117	041440					
9113	046500	042514	051101	000					
9114	046505	101	052124	047047	EM40:	.ASCIZ	/ATT'N BUT NO STATUS CHANGE OR FAULT/		
9115	046512	041040	052125	047040					
9116	046520	020117	052123	052101					
9117	046526	051525	041440	040510					
9118	046534	043516	020105	051117					
9119	046542	043040	052501	052114					
9120	046550	000							
9121	046551	101	052124	047047	EM41:	.ASCIZ	/ATT'N BUT DRIVE NOT AVAILABLE/		
9122	046556	041040	052125	042040					
9123	046564	044522	042526	047040					
9124	046572	052117	040440	040526					
9125	046600	046111	041101	042514					

9126	046606	000				
9127	046607	101	052124	047047	EM42:	.ASCIZ /ATT'N WHEN NOT EXPECTED/
9128	046614	053440	042510	020116		
9129	046622	047516	020124	054105		
9130	046630	042520	052103	042105		
9131	046636	000				
9132	046637	105	051122	051117	EM43:	.ASCIZ /ERROR GATHERING DRIVE STATUS/
9133	046644	043440	052101	042510		
9134	046652	044522	043516	042040		
9135	046660	044522	042526	051440		
9136	046666	040524	052524	000123		
9137	046674	052515	052114	050111	EM52:	.ASCIZ /MULTIPLE DRIVE SELECT/
9138	046702	042514	042040	044522		
9139	046710	042526	051440	046105		
9140	046716	041505	000124			
9141	046722	042510	042101	051105	EM53:	.ASCIZ /HEADER COMPARE ERROR/
9142	046730	041440	046517	040520		
9143	046736	042522	042440	051122		
9144	046744	051117	000			
9145	046747	123	041125	054523	EM56:	.ASCIZ /SUBSYS TIMEOUT/
9146	046754	020123	044524	042515		
9147	046762	052517	000124			
9148	046766	051105	047522	020122	EM60:	.ASCIZ /ERROR IN RECAL FOR RECOVERY/
9149	046774	047111	051040	041505		
9150	047002	046101	043040	051117		
9151	047010	051040	041505	053117		
9152	047016	051105	000131			
9153	047022	051120	043517	040522	EM61:	.ASCIZ /PROGRAM ABORTING FATAL ERROR IN RETRY/
9154	047030	020115	041101	051117		
9155	047036	044524	043516	043040		
9156	047044	052101	046101	042440		
9157	047052	051122	051117	044440		
9158	047060	020116	042522	051124		
9159	047066	000131				
9160	047070	054503	044514	042116	EM62:	.ASCIZ /CYLINDER MISCOMPARE/
9161	047076	051105	046440	051511		
9162	047104	047503	050115	051101		
9163	047112	000105				
9164	047114	046103	040505	020122	EM63:	.ASCIZ /CLEAR CONTROLLER DID NOT CLEAR ERROR/
9165	047122	047503	052116	047522		
9166	047130	046114	051105	042040		
9167	047136	042111	047040	052117		
9168	047144	041440	042514	051101		
9169	047152	042440	051122	051117		
9170	047160	000				
9171	047161	116	020117	052101	EM64:	.ASCIZ /NO ATT'N IN ATT'N SUMMARY REGISTER/
9172	047166	023524	020116	047111		
9173	047174	040440	052124	047047		
9174	047202	051440	046525	040515		
9175	047210	054522	051040	043505		
9176	047216	051511	042524	000122		
9177	047224	047125	047523	044514	EM65:	.ASCIZ /UNSOLICITED ATTENTION/
9178	047232	044503	042524	020104		
9179	047240	052101	042524	052116		
9180	047246	047511	000116			
9181	047252	047125	054105	042520	EM66:	.ASCIZ /UNEXPECTED DATA TYPE ERROR/

9182	047260	052103	042105	042040		
9183	047266	052101	020101	054524		
9184	047274	042520	042440	051122		
9185	047302	051117	000			
9186	047305	101	052124	047047	EM67:	.ASCIZ /ATT'N DID NOT RESET WITH CLEAR/
9187	047312	042040	042111	047040		
9188	047320	052117	051040	051505		
9189	047326	052105	053440	052111		
9190	047334	020110	046103	040505		
9191	047342	000122				
9192	047344	052523	051502	051531	EM70:	.ASCIZ /SUBSYS CLEAR DID NOT CLEAR DRIVE ATT'N/
9193	047352	041440	042514	051101		
9194	047360	042040	042111	047040		
9195	047366	052117	041440	042514		
9196	047374	051101	042040	044522		
9197	047402	042526	040440	052124		
9198	047410	047047	000			
9199	047413	104	052101	020101	EM71:	.ASCIZ /DATA LATE WHEN UNLOADING HEADER/
9200	047420	040514	042524	053440		
9201	047426	042510	020116	047125		
9202	047434	047514	042101	047111		
9203	047442	020107	042510	042101		
9204	047450	051105	000			
9205	047453	103	047117	051124	EM72:	.ASCIZ /CONTROLLER ERROR WHEN DRIVER SERVICING/
9206	047460	046117	042514	020122		
9207	047466	051105	047522	020122		
9208	047474	044127	047105	042040		
9209	047502	044522	042526	020122		
9210	047510	042523	053122	041511		
9211	047516	047111	000107			
9212	047522	051104	053111	020105	EM73:	.ASCIZ /DRIVE DETECTED PARITY ERROR/
9213	047530	042504	042524	052103		
9214	047536	042105	050040	051101		
9215	047544	052111	020131	051105		
9216	047552	047522	000122			
9217	047556	047125	042504	044506	EM74:	.ASCIZ /UNDEFINED ERROR/
9218	047564	042516	020104	051105		
9219	047572	047522	000122			
9220	047576	040515	045522	047111	EM75:	.ASCIZ /MARKING THIS SECTOR BAD/
9221	047604	020107	044124	051511		
9222	047612	051440	041505	047524		
9223	047620	020122	040502	000104		
9224	047626	040502	020104	040504	EM76:	.ASCIZ /BAD DATA VERIF'N WITH READ. ECC OF LAST RETRY IS:/
9225	047634	040524	053040	051105		
9226	047642	043111	047047	053440		
9227	047650	052111	020110	042522		
9228	047656	042101	020056	041505		
9229	047664	020103	043117	046040		
9230	047672	051501	020124	042522		
9231	047700	051124	020131	051511		
9232	047706	000072				
9233	047710	042522	051124	020131	EM77:	.ASCIZ /RETRY SUCCESSFUL/
9234	047716	052523	041503	051505		
9235	047724	043123	046125	000		
9236	047731	122	052105	054522	EM100:	.ASCIZ /RETRY UNSUCCESSFUL/
9237	047736	052440	051516	041525		

9238	047744	042503	051523	052506	
9239	047752	000114			
9240	047754	040503	047116	052117	EM101: .ASCIZ /CANNOT FIND A VALID HEADER IN TRACK JUST READ/
9241	047762	043040	047111	020104	
9242	047770	020101	040526	044514	
9243	047776	020104	042510	042101	
9244	050004	051105	044440	020116	
9245	050012	051124	041501	020113	
9246	050020	052512	052123	051040	
9247	050026	040505	000104		
9248	050032	040502	020104	042523	EM102: .ASCIZ /BAD SECTOR ERROR ON SECTOR NOT LISTED BAD/
9249	050040	052103	051117	042440	
9250	050046	051122	051117	047440	
9251	050054	020116	042523	052103	
9252	050062	051117	047040	052117	
9253	050070	046040	051511	042524	
9254	050076	020104	040502	000104	
9255	050104	044524	042515	026504	EM103: .ASCIZ /TIMED-OUT ON READ HDR/
9256	050112	052517	020124	047117	
9257	050120	051040	040505	020104	
9258	050126	042110	000122		
9259	050132	044524	042515	026504	EM104: .ASCIZ /TIMED-OUT ON SEEK/
9260	050140	052517	020124	047117	
9261	050146	051440	042505	000113	
9262	050154	051104	053111	020105	EM105: .ASCIZ /DRIVE SIEZED BY OTHER PORT/
9263	050162	044523	055105	042105	
9264	050170	041040	020131	052117	
9265	050176	042510	020122	047520	
9266	050204	052122	000		
9267	050207	104	052101	020101	EM106: .ASCIZ /DATA MISCMPR WHILE BAI SET/
9268	050214	044515	041523	050115	
9269	050222	020122	044127	046111	
9270	050230	020105	040502	020111	
9271	050236	042523	000124		
9272	050242	047516	047040	046505	EM107: .ASCIZ /NO NEM ERROR WHEN REF'ING LOC 760000/
9273	050250	042440	051122	051117	
9274	050256	053440	042510	020116	
9275	050264	042522	023506	047111	
9276	050272	020107	047514	020103	
9277	050300	033067	030060	030060	
9278	050306	000			
9279	050307	111	052116	050122	EM110: .ASCIZ /INTRPT WHEN CNTRLR NOT RDY/
9280	050314	020124	044127	047105	
9281	050322	041440	052116	046122	
9282	050330	020122	047516	020124	
9283	050336	042122	000131		
9284	050342	047516	040440	052124	EM111: .ASCIZ /NO ATT'N ON SEEK/
9285	050350	047047	047440	020116	
9286	050356	042523	045505	000	
9287	050363	104	044522	042526	EM112: .ASCIZ /DRIVE'S CYLINDER INCORRECT/
9288	050370	051447	041440	046131	
9289	050376	047111	042504	020122	
9290	050404	047111	047503	051122	
9291	050412	041505	000124		
9292	050416	041101	051117	026524	EM113: .ASCIZ /ABORT- CAN'T READ BSF/
9293	050424	041440	047101	052047	

9294	050432	051040	040505	020104	
9295	050440	051502	000106		
9296	050444	052113	030461	043040	EM114: .ASCIZ /KT11 FAILURE/
9297	050452	044501	052514	042522	
9298	050460	000			
9299	050461	115	046505	050040	EM115: .ASCIZ /MEM PARITY ERR/
9300	050466	051101	052111	020131	
9301	050474	051105	000122		
9302	050500	052503	051122	047105	DH100: .ASCIZ /CURRENT COMMAND :/
9303	050506	020124	047503	046515	
9304	050514	047101	020104	000072	
9305	050522	051120	053105	047511	DH105: .ASCIZ /PREVIOUS COMMAND :/
9306	050530	051525	041440	046517	
9307	050536	040515	042116	035040	
9308	050544	000			
9309	050545	122	033113	030461	DH200: .ASCIZ /RK611 REGISTERS :/
9310	050552	051040	043505	051511	
9311	050560	042524	051522	035040	
9312	050566	000			
9313	050567	122	046505	044501	DH500: .ASCIZ /REMAINING REGISTERS NOT VALID/
9314	050574	044516	043516	051040	
9315	050602	043505	051511	042524	
9316	050610	051522	047040	052117	
9317	050616	053040	046101	042111	
9318	050624	000			
9319	050625	124	042510	043040	DH501: .ASCIZ /THE FOLLOWING REGISTER DATA MAY BE INCORRECT :/
9320	050632	046117	047514	044527	
9321	050640	043516	051040	043505	
9322	050646	051511	042524	020122	
9323	050654	040504	040524	046440	
9324	050662	054501	041040	020105	
9325	050670	047111	047503	051122	
9326	050676	041505	020124	000072	
9327	050704	051105	020122	041520	DH101: .ASCIZ /ERR PC DRIVE CMND CYLNDR TRACK SECTOR WD CNT/
9328	050712	020040	051104	053111	
9329	050720	020105	020040	046503	
9330	050726	042116	020040	020040	
9331	050734	054503	047114	051104	
9332	050742	020040	051124	041501	
9333	050750	020113	020040	042523	
9334	050756	052103	051117	020040	
9335	050764	042127	041440	052116	
9336	050772	000			
9337	050773	110	020111	040502	DH102: .ASCIZ /HI BA LO BA/
9338	051000	020040	046040	020117	
9339	051006	040502	000		
9340	051011	120	042522	027126	DH103: .ASCIZ /PREV. UNIBUS MAP :/
9341	051016	052440	044516	052502	
9342	051024	020123	040515	020120	
9343	051032	000072			
9344	051034	044510	051040	043505	DH104: .ASCIZ /HI REGO LO REGO/
9345	051042	020060	047514	051040	
9346	051050	043505	000060		
9347	051054	052503	027122	052440	DH106: .ASCIZ /CUR. UNIBUS MAP :/
9348	051062	044516	052502	020123	
9349	051070	040515	020120	000072	

9406	051560	051117	031504	000	
9407	051565	102	042101	044040	DH607: .ASCIZ /BAD HEADER IS :/
9408	051572	040505	042504	020122	
9409	051600	051511	035040	000	
9410	051605	127	020104	020043	DH701: .ASCIZ /WD # GOOD BAD HI PHY LO PHY VRT AD KIPAR6/
9411	051612	020040	043440	047517	
9412	051620	020104	020040	041040	
9413	051626	042101	020040	020040	
9414	051634	044040	020111	044120	
9415	051642	020131	046040	020117	
9416	051650	044120	020131	053040	
9417	051656	052122	040440	020104	
9418	051664	045440	050111	051101	
9419	051672	000066			
9420	051674	047507	042117	020040	DH702: .ASCIZ /GOOD BAD/
9421	051702	020040	040502	000104	
9422	051710	040506	046111	047111	DH703: .ASCIZ /FAILING DATA WORD :/
9423	051716	020107	040504	040524	
9424	051724	053440	051117	020104	
9425	051732	000072			
9426	051734	051123	004460	051123	DH704: .ASCIZ /SRO SR1 SR2 SR3/
9427	051742	004461	051123	004462	
9428	051750	051123	000063		
9429	051754	052113	030461	051040	DH705: .ASCIZ /KT11 REGS :/
9430	051762	043505	020123	000072	
9431	051770	042515	020115	042522	DH706: .ASCIZ /MEM REGS :/
9432	051776	051507	035040	000	
9433	052003	120	004503	047514	DH707: .ASCIZ /PC LOERAD HIERAD MEMSYS/
9434	052010	051105	042101	044011	
9435	052016	042511	040522	004504	
9436	052024	042515	051515	051531	
9437	052032	000			
9438	052033	120	004503	051503	DH710: .ASCIZ /PC CSR AD CSR/
9439	052040	020122	042101	041411	
9440	052046	051123	000		
9441	052051	103	046131	047111	DH711: .ASCIZ /CYLINDERS :/
9442	052056	042504	051522	035040	
9443	052064	000			
9444	052065	122	042113	020123	DH204: .ASCIZ /RKDS RKER RKMR2 RKMR3/
9445	052072	020040	051040	042513	
9446	052100	020122	020040	051040	
9447	052106	046513	031122	020040	
9448	052114	051040	046513	031522	
9449	052122	000			
9450	052123	105	051122	051117	DH502: .ASCIZ /ERROR TRYING TO GET FAILURE INFO/
9451	052130	052040	054522	047111	
9452	052136	020107	047524	043440	
9453	052144	052105	043040	044501	
9454	052152	052514	042522	044440	
9455	052160	043116	000117		
9456	052164	042523	047503	042116	DH503: .ASCIZ /SECOND TIMEOUT OCCURRED GETTING DRIVE STATUS/
9457	052172	052040	046511	047505	
9458	052200	052125	047440	041503	
9459	052206	051125	042522	020104	
9460	052214	042507	052124	047111	
9461	052222	020107	051104	053111	

9462	052230	020105	052123	052101		
9463	052236	051525	000			
9464	052241	116	046525	042502	DH800:	.ASCIZ /NUMBER OF RETRIES:/
9465	052246	020122	043117	051040		
9466	052254	052105	044522	051505		
9467	052262	000072				
9468						.EVEN
9469	052264	001116	005430	005426	DT100:	.WORD \$ERRPC,DRIVE,ERRCOM,\$REG0,\$REG1,\$REG2,\$REG3
9470	052272	001162	001164	001166		
9471	052300	001170				
9472	052302	001256	001260		DT102:	.WORD \$REG36,\$REG37
9473	052306	001174	001176	001200	DT201:	.WORD \$REG5,\$REG6,\$REG7,\$REG10,\$REG11,\$REG12,\$REG13
9474	052314	001202	001204	001206		
9475	052322	001210				
9476	052324	001212	001214		DT202:	.WORD \$REG14,\$REG15
9477	052330	001216	001220	001222	DT203:	.WORD \$REG16,\$REG17,\$REG20,\$REG21,\$REG22,\$REG23,\$REG24,\$REG25
9478	052336	001224	001226	001230		
9479	052344	001232	001234			
9480	052350	001174	001176	001200	DT601:	.WORD \$REG5,\$REG6,\$REG7
9481	052356	001202	001204	001206	DT602:	.WORD \$REG10,\$REG11,\$REG12,\$REG13,\$REG14,\$REG15,\$REG16
9482	052364	001210	001212	001214		
9483	052372	001216				
9484						
9485	052374	000006			DF01:	.WORD 6 :NUMBER OF HEADER LINES
9486	052376	000				.BYTE 0 :NUMBER OF COL FOR FIRST HDR
9487	052377	000				.BYTE 0 :ALL CHARACTERS OCTAL
9488	052400	050704				.WORD DH101 :SECOND HDR LINE
9489	052402	007	000			.BYTE 7,0 :NUM OF COL-ALL OCTAL
9490	052404	050773				.WORD DH102
9491	052406	002	000			.BYTE 2,0
9492	052410	050545				.WORD DH200
9493	052412	000	000			.BYTE 0,0
9494	052414	051076				.WORD DH201
9495	052416	007	000			.BYTE 7,0
9496	052420	050567				.WORD DH500
9497	052422	000	000			.BYTE 0,0
9498						
9499	052424	000007			DF02:	.WORD 7
9500	052426	000	000			.BYTE 0,0
9501	052430	050704				.WORD DH101
9502	052432	007	000			.BYTE 7,0
9503	052434	050773				.WORD DH102
9504	052436	002	000			.BYTE 2,0
9505	052440	050545				.WORD DH200
9506	052442	000	000			.BYTE 0,0
9507	052444	051076				.WORD DH201
9508	052446	007	000			.BYTE 7,0
9509	052450	051165				.WORD DH202
9510	052452	002	000			.BYTE 2,0
9511	052454	051202				.WORD DH203
9512	052456	010	000			.BYTE 10,0
9513						
9514	052460	000010			DF03:	.WORD 10
9515	052462	000	000			.BYTE 0,0
9516	052464	050704				.WORD DH101
9517	052466	007	000			.BYTE 7,C

9518	052470	050773		.WORD	DH102
9519	052472	002	000	.BYTE	2,0
9520	052474	050545		.WORD	DH200
9521	052476	000	000	.BYTE	0,0
9522	052500	051076		.WORD	DH201
9523	052502	007	000	.BYTE	7,0
9524	052504	050625		.WORD	DH501
9525	052506	000	000	.BYTE	0,0
9526	052510	051165		.WORD	DH202
9527	052512	002	000	.BYTE	2,0
9528	052514	051202		.WORD	DH203
9529	052516	010	000	.BYTE	10,0
9530					
9531	052520	000003		DF04: .WORD	3
9532	052522	000	000	.BYTE	0,0
9533	052524	050704		.WORD	DH101
9534	052526	007	000	.BYTE	7,0
9535	052530	050773		.WORD	DH102
9536	052532	002	000	.BYTE	2,0
9537					
9538	052534	000007		DF05: .WORD	7
9539	052536	000	000	.BYTE	0,0
9540	052540	050704		.WORD	DH101
9541	052542	007	000	.BYTE	7,0
9542	052544	050773		.WORD	DH102
9543	052546	002	000	.BYTE	2,0
9544	052550	051276		.WORD	DH601
9545	052552	000	000	.BYTE	0,0
9546	052554	051537		.WORD	DH606
9547	052556	003	000	.BYTE	3,0
9548	052560	051320		.WORD	DH602
9549	052562	000	000	.BYTE	0,0
9550	052564	051537		.WORD	DH606
9551	052566	003	000	.BYTE	3,0
9552					
9553	052570	000007		DF07: .WORD	7
9554	052572	000	000	.BYTE	0,0
9555	052574	050704		.WORD	DH101
9556	052576	007	000	.BYTE	7,0
9557	052600	050773		.WORD	DH102
9558	052602	002	000	.BYTE	2,0
9559	052604	051365		.WORD	DH604
9560	052606	000	000	.BYTE	0,0
9561	052610	051420		.WORD	DH6041
9562	052612	003	000	.BYTE	3,0
9563	052614	051465		.WORD	DH605
9564	052616	000	000	.BYTE	0,0
9565	052620	051502		.WORD	DH6051
9566	052622	003	000	.BYTE	3,0
9567					
9568	052624	000005		DF10: .WORD	5
9569	052626	000	000	.BYTE	0,0
9570	052630	050704		.WORD	DH101
9571	052632	007	000	.BYTE	7,0
9572	052634	050773		.WORD	DH102
9573	052636	002	000	.BYTE	2,0

;"THE FOLLOWING REG DATA MB INCORRECT"

;OPI, HVRC

9574	052640	051365		.WORD	DH604	
9575	052642	000	000	.BYTE	0,0	
9576	052644	051420		.WORD	DH6041	
9577	052646	003	000	.BYTE	3,0	
9578						
9579	052650	000004		DF11: .WORD	4	
9580	052652	000	000	.BYTE	0,0	
9581	052654	051365		.WORD	DH604	
9582	052656	000	000	.BYTE	0,0	
9583	052660	051420		.WORD	DH6041	
9584	052662	003	000	.BYTE	3,0	
9585	052664	051605		.WORD	DH701	
9586	052666	000	000	.BYTE	0,0	
9587						
9588	052670	000006		DF12: .WORD	6	
9589	052672	000	000	.BYTE	0,0	
9590	052674	050704		.WORD	DH101	
9591	052676	007	000	.BYTE	7,0	
9592	052700	050773		.WORD	DH102	
9593	052702	002	000	.BYTE	2,0	
9594	052704	050545		.WORD	DH200	
9595	052706	000	000	.BYTE	0,0	
9596	052710	051076		.WORD	DH201	
9597	052712	007	000	.BYTE	7,0	
9598	052714	052065		.WORD	DH204	
9599	052716	004	000	.BYTE	4,0	
9600						
9601	052720	000006		DF13: .WORD	6	
9602	052722	000	000	.BYTE	0,0	
9603	052724	050704		.WORD	DH101	
9604	052726	007	000	.BYTE	7,0	
9605	052730	050773		.WORD	DH102	
9606	052732	002	000	.BYTE	2,0	
9607	052734	051276		.WORD	DH601	
9608	052736	000	000	.BYTE	0,0	
9609	052740	051537		.WORD	DH606	
9610	052742	003	000	.BYTE	3,0	
9611	052744	052123		.WORD	DH502	
9612	052746	000	000	.BYTE	0,0	
9613						
9614	052750	000006		DF14: .WORD	6	
9615	052752	000	000	.BYTE	0,0	
9616	052754	050704		.WORD	DH101	
9617	052756	007	000	.BYTE	7,0	
9618	052760	050773		.WORD	DH102	
9619	052762	002	000	.BYTE	2,0	
9620	052764	051276		.WORD	DH601	
9621	052766	000	000	.BYTE	0,0	
9622	052770	051537		.WORD	DH606	
9623	052772	003	000	.BYTE	3,0	
9624	052774	052123		.WORD	DH502	
9625	052776	000	000	.BYTE	0,0	
9626						
9627	053000	000011		DF15: .WORD	11	
9628	053002	000	000	.BYTE	0,0	
9629	053004	050704		.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

9630	053006	007	000		.BYTE	7,0
9631	053010	050773			.WORD	DH102
9632	053012	002	000		.BYTE	2,0
9633	053014	050545			.WORD	DH200
9634	053016	000	000		.BYTE	0,0
9635	053020	051076			.WORD	DH201
9636	053022	007	000		.BYTE	7,0
9637	053024	051165			.WORD	DH202
9638	053026	002	000		.BYTE	2,0
9639	053030	052164			.WORD	DH503
9640	053032	000	000		.BYTE	0,0
9641	053034	050625			.WORD	DH501
9642	053036	000	000		.BYTE	0,0
9643	053040	051202			.WORD	DH203
9644	053042	010	000		.BYTE	10,0
9645						
9646	053044	000011		DF16:	.WORD	11
9647	053046	000	000		.BYTE	0,0
9648	053050	050704			.WORD	DH101
9649	053052	007	000		.BYTE	7,0
9650	053054	050773			.WORD	DH102
9651	053056	002	000		.BYTE	2,0
9652	053060	050545			.WORD	DH200
9653	053062	000	000		.BYTE	0,0
9654	053064	051076			.WORD	DH201
9655	053066	007	000		.BYTE	7,0
9656	053070	051165			.WORD	DH202
9657	053072	002	000		.BYTE	2,0
9658	053074	052123			.WORD	DH502
9659	053076	000	000		.BYTE	0,0
9660	053100	050625			.WORD	DH501
9661	053102	000	000		.BYTE	0,0
9662	053104	051202			.WORD	DH203
9663	053106	010	000		.BYTE	10,0
9664						
9665	053110	000005		DF17:	.WORD	5
9666	053112	000	000		.BYTE	0,0
9667	053114	050704			.WORD	DH101
9668	053116	007	000		.BYTE	7,0
9669	053120	050773			.WORD	DH102
9670	053122	002	000		.BYTE	2,0
9671	053124	052051			.WORD	DH711
9672	053126	000	000		.BYTE	0,0
9673	053130	051674			.WORD	DH702
9674	053132	002	000		.BYTE	2,0
9675						
9676	053134	000002		DF20:	.WORD	2
9677	053136	000	000		.BYTE	0,0
9678	053140	051034			.WORD	DH104
9679	053142	002	000		.BYTE	2,0
9680						
9681	053144	000002		DF21:	.WORD	2
9682	053146	000	000		.BYTE	0,0
9683	053150	051502			.WORD	DH6051
9684	053152	003	000		.BYTE	3,0
9685						

9686	053154	000006		DF22:	.WORD	6
9687	053156	000	000		.BYTE	0,0
9688	053160	050500			.WORD	DH100
9689	053162	000	000		.BYTE	0,0
9690	053164	050704			.WORD	DH101
9691	053166	007	000		.BYTE	7,0
9692	053170	050773			.WORD	DH102
9693	053172	002	000		.BYTE	2,0
9694	053174	051054			.WORD	DH106
9695	053176	000	000		.BYTE	0,0
9696	053200	051034			.WORD	DH104
9697	053202	002	000		.BYTE	2,0
9698						
9699	053204	000002		DF23:	.WORD	2
9700	053206	000	000		.BYTE	0,0
9701	053210	052241			.WORD	DH800
9702	053212	001	000		.BYTE	1,0
9703						
9704	053214	000001		DF24:	.WORD	1
9705	053216	002	000		.BYTE	2,0
9706						
9707	053220	000000		DF25:	.WORD	0
9708	053222	007	000		.BYTE	7,0
9709						
9710	053224	000002		DF26:	.WORD	2
9711	053226	002	000		.BYTE	2,0
9712	053230	051202			.WORD	DH203
9713	053232	010	000		.BYTE	10,0
9714						
9715	053234	000005		DF27:	.WORD	5
9716	053236	000	000		.BYTE	0,0
9717	053240	050704			.WORD	DH101
9718	053242	007	000		.BYTE	7,0
9719	053244	050773			.WORD	DH102
9720	053246	002	000		.BYTE	2,0
9721	053250	051710			.WORD	DH703
9722	053252	000	000		.BYTE	0,0
9723	053254	051674			.WORD	DH702
9724	053256	002	000		.BYTE	2,0
9725						
9726	053260	000005		DF30:	.WORD	5
9727	053262	000	000		.BYTE	0,0
9728	053264	050704			.WORD	DH101
9729	053266	007	000		.BYTE	7,0
9730	053270	050773			.WORD	DH102
9731	053272	002	000		.BYTE	2,0
9732	053274	051754			.WORD	DH705
9733	053276	000	000		.BYTE	0,0
9734	053300	051734			.WORD	DH704
9735	053302	004	000		.BYTE	4,0
9736						
9737	053304	000005		DF31:	.WORD	5
9738	053306	000	000		.BYTE	0,0
9739	053310	050704			.WORD	DH101
9740	053312	007	000		.BYTE	7,0
9741	053314	050773			.WORD	DH102

9742	053316	002	000		.BYTE	2,0	
9743	053320	051770			.WORD	DH706	
9744	053322	000	000		.BYTE	0,0	
9745	053324	052033			.WORD	DH710	
9746	053326	003	000		.BYTE	3,0	
9747							
9748	053330	000400			RWBUF: .BLKW	256.	;READ/WRITE DATA BUFFER
9749							
9750	054330	005015	020040	020040	NOTMSG: .ASCII	<15><12>/	*** NOTE ***/<15><12><12>
9751	054336	020040	020040	020040			
9752	054344	025052	020052	047516			
9753	054352	042524	025040	025052			
9754	054360	005015	012				
9755	054363	117	046116	020131	.ASCII	/ONLY ALL RK06 OR ALL RK07'S MUST BE UNDER TEST/<15><12><12>	
9756	054370	046101	020114	045522			
9757	054376	033060	047440	020122			
9758	054404	046101	020114	045522			
9759	054412	033460	051447	046440			
9760	054420	051525	020124	042502			
9761	054426	052440	042116	051105			
9762	054434	052040	051505	006524			
9763	054442	005012					
9764	054444	046101	020114	051104	.ASCII	/ALL DRIVES TO BE TESTED MUST HAVE :/<15><12><12>	
9765	054452	053111	051505	052040			
9766	054460	020117	042502	052040			
9767	054466	051505	042524	020104			
9768	054474	052515	052123	044040			
9769	054502	053101	020105	006472			
9770	054510	005012					
9771	054512	027061	042040	051505	.ASCII	/1. DESIRED PORT SELECTED/<15><12>	
9772	054520	051111	042105	050040			
9773	054526	051117	020124	042523			
9774	054534	042514	052103	042105			
9775	054542	005015					
9776	054544	027062	053440	044522	.ASCII	/2. WRITE LOCK DISABLED/<15><12>	
9777	054552	042524	046040	041517			
9778	054560	020113	044504	040523			
9779	054566	046102	042105	005015			
9780	054574	051104	053111	051505	.ASCII	/DRIVES NOT TO BE TESTED MUST HAVE BOTH/<15><12>	
9781	054602	047040	052117	052040			
9782	054610	020117	042502	052040			
9783	054616	051505	042524	020104			
9784	054624	052515	052123	044040			
9785	054632	053101	020105	047502			
9786	054640	044124	005015				
9787	054644	047520	052122	020123	.ASCII	/PORTS DE-SELECTED./<15><12>	
9788	054652	042504	051455	046105			
9789	054660	041505	042524	027104			
9790	054666	005015					
9791	054670	044124	020105	040503	.ASCII	/THE CARTRIDGE MUST BE FORMATTED ON A KNOWN GOOD/<15><12>	
9792	054676	052122	044522	043504			
9793	054704	020105	052515	052123			
9794	054712	041040	020105	047506			
9795	054720	046522	052101	042524			
9796	054726	020104	047117	040440			
9797	054734	045440	047516	047127			

9798	054742	043440	047517	006504
9799	054750	012		
9800	054751	127	046105	026514
9801	054756	046101	043511	042516
9802	054764	020104	051104	053111
9803	054772	026105	050040	044522
9804	055000	051117	052040	020117
9805	055006	042524	052123	047111
9806	055014	027107	005015	
9807	055020	044124	020105	047503
9808	055026	050115	042514	042524
9809	055034	051040	047125	052040
9810	055042	046511	020105	051511
9811	055050	034040	030455	020060
9812	055056	044515	027116	050040
9813	055064	051105	042040	044522
9814	055072	042526	006456	012
9815	055077	124	020117	041101
9816	055104	051117	020124	042524
9817	055112	052123	047111	026107
9818	055120	052040	050131	020105
9819	055126	047503	052116	047522
9820	055134	026514	020103	057050
9821	055142	024503	006456	005012
9822	055150	000		
9823				
9824				
9825				
9826				
9827				
9828				
9829	000001			

.ASCII /WELL-ALIGNED DRIVE, PRIOR TO TESTING./<15><12>

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

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

.END

C.INIT	040210	6311	7869#											
C.RTRN	042062	7973	8001	8029	8036	8039	8046	8050	8067	8089	8094	8104	8113	8126
		8130	8135#											
C.SPEC	041102	7909	8005#											
DCEFLG	003144	2930#	4720*	6188*										
DCK =	100000	2554#	4716	6011	6827	7260	7426	7662						
DCKERR=	000020	3240#	4719	5271	6128	6193	6829	6923						
DCKHDL	020526	4711#	6190	6876	6896									
DCLO =	000020	2563#												
DDISP =	177570	1349#	1526	3542										
DDT =	000400	2568#												
DERCNT	003130	2918#	6757	6787	6836	6847								
DESL =	000010	2518#												
DFLTR	013744	3602	3784#											
DFSTR	012146	1453	3505#	5053	5125									
DF01	052374	1657	1663	1669	1675	1717	1825	2095	2131	9485#				
DF02	052424	1681	1699	1705	1711	1723	1729	1735	1741	1747	1753	1759	1765	1771
		1777	1783	1789	1843	1849	1855	1963	1969	1975	1981	1987	1993	1999
		2005	2011	2017	2059	2077	2101	9499#						
DF03	052460	1687	1693	2065	2071	9514#								
DF04	052520	1819	9531#											
DF05	052534	1795	1801	9538#										
DF07	052570	1807	9553#											
DF10	052624	1813	9568#											
DF11	052650	2113	9579#											
DF12	052670	1861	1867	1873	1879	1885	1891	1897	1903	9588#				
DF13	052720	1909	1921	9601#										
DF14	052750	1915	9614#											
DF15	053000	1927	9627#											
DF16	053044	1933	9646#											
DF17	053110	1951	9665#											
DF20	053134	2119	9676#											
DF21	053144	2029	9681#											
DF22	053154	2125	9686#											
DF23	053204	2035	2041	2047	9699#									
DF24	053214	2053	9704#											
DF25	053220	1957	5680*	5710*	5892*	5910*	9707#							
DF26	053224	2107	9710#											
DF27	053234	2083	9715#											
DF30	053260	2137	9726#											
DF31	053304	2143	9737#											
DH100	050500	1655	1661	1667	1673	1679	1685	1691	1697	1703	1709	1715	1721	1727
		1733	1739	1745	1751	1757	1763	1769	1775	1781	1787	1793	1799	1805
		1811	1817	1823	1841	1847	1853	1859	1865	1871	1877	1883	1889	1895
		1901	1907	1913	1919	1925	1931	1949	1961	1967	1973	1979	1985	1991
		1997	2003	2009	2015	2057	2063	2069	2075	2081	2093	2099	2129	2135
		2141	9302#	9688										
DH101	050704	6409	9327#	9488	9501	9516	9533	9540	9555	9570	9590	9603	9616	9629
		9648	9667	9690	9717	9728	9739							
DH102	050773	6420	9337#	9490	9503	9518	9535	9542	9557	9572	9592	9605	9618	9631
		9650	9669	9692	9719	9730	9741							
DH103	051011	2117	9340#											
DH104	051034	9344#	9678	9696										
DH105	050522	6407	9305#											
DH106	051054	9347#	9694											
DH200	050545	9309#	9492	9505	9520	9594	9633	9652						

EM113	050416	2128	9292#	
EM114	050444	2134	9296#	
EM115	050461	2140	9299#	
EM12	045642	1708	9033#	
EM13	045663	1714	9036#	
EM14	045705	1720	9040#	
EM15	045733	1726	9044#	
EM16	045754	1732	9047#	
EM17	045771	1738	9050#	
EM2	045431	1660	9006#	
EM20	046012	1744	9053#	
EM21	046027	1750	9056#	
EM22	046047	1756	9059#	
EM23	046071	1762	9063#	
EM24	046122	1768	9068#	
EM25	046142	1774	9071#	
EM26	046165	1780	9075#	
EM27	046177	1786	9077#	
EM3	045455	1666	2146	9010#
EM30	046222	1792	1912	9081#
EM31	046247	1798	1918	9085#
EM32	046270	1804	9088#	
EM33	046311	1810	9091#	
EM34	046333	1816	2116	9095#
EM35	046353	1822	9098#	
EM36	046407	1828	9103#	
EM37	046442	1834	9108#	
EM4	045500	1672	9014#	
EM40	046505	1840	9114#	
EM41	046551	1846	9121#	
EM42	046607	1852	9127#	
EM43	046637	1858	9132#	
EM5	045521	1678	9017#	
EM52	046674	1900	9137#	
EM53	046722	1906	9141#	
EM56	046747	1924	1930	9145#
EM6	045540	1684	9020#	
EM60	046766	1936	9148#	
EM61	047022	1942	9153#	
EM62	047070	1948	9160#	
EM63	047114	1864	1960	9164#
EM64	047161	1870	1966	9171#
EM65	047224	1876	1972	9177#
EM66	047252	1882	1978	9181#
EM67	047305	1888	1984	9186#
EM7	045564	1690	9024#	
EM70	047344	1894	1990	9192#
EM71	047413	1996	9199#	
EM72	047453	2002	9205#	
EM73	047522	2008	9212#	
EM74	047556	2014	9217#	
EM75	047576	2020	9220#	
EM76	047626	2026	9224#	
EM77	047710	2032	2038	9233#
ENDTST	011140	3387#	4408	
EQUALS	012134	3496#		

RKPRI	003052	2840#	3575	3600*	3751	3767*	7152	7876						
RKPRTY	007677	3264#	3756											
RKVADR	007654	3260#	3747											
RKVEC	003050	2839#	3573	3596*	3597*	3746	3749*							
RKWC =	000002	2442#	6977	7700	7748	7976*								
RLS =	000010	2519#	8056											
ROTBLD	023000	3960	4115	4247	5200#									
RWBUF	053330	3912	4088*	4089	4158	4234	4982	4990	5463	5468	5558	5562	5639	5688
		5832	5838	6162	9748#									
R.ABNL	040142	7165	7275	7350	7367	7545	7565	7825#	8035					
R.CONT	040166	7227	7265	7329	7338	7409	7431	7454	7460	7493	7533	7557	7602	7654
		7677	7815	7833#	8049	8103	8112	8125						
R.NORM	040154	7303	7396	7505	7829#	7972	8045	8133						
SAVPRS	005454	2974#	6159	6166										
SAVREG=	104407	4457	4530	4572	4625	5146	5200	5229	5268	5420	5557	5585	5633	5678
		5716	5830	5854	5858	5882	5886	5901	6025	6072	6122	6157	6165	6260
		6323	6376	6486	6542	6604	6970	7011	7077	8417	8458	8995#		
SAVWRD	003160	2951#												
SCLR =	000040	2522#	8106											
SCNDRV	021274	4916#	5065											
SCOPE=	104411	8997#												
SCOPE1	044550	8802#	8997											
SCRACH	005444	2970#	3675*	3678*	3683	3823*	3824*	4925*	4926*	5381*	5383*	5384	5394*	5395*
		5396	5682*	5724	5730*	5733	5890*	5911	5917*	5920				
SCRLST	005556	2989#	3648	3656*	3662	3678	3790*	3793*	3794*	3812	3847*			
SECLST	006572	3130#	5240	5287										
SEEK =	000117	2469#	4079	4171	4466	7923	7933							
SELDRV=	000101	2462#	3828	4930	8062									
SELF	011205	3394#	4338											
SERNM	011717	3462#	5425	5467										
SETTUP	022020	3868	3891	3939	3994	4039	4073	4145	4222	4319	5007#			
SKI =	000002	2538#	6725											
SOFTBS	011745	3466#	6061											
SPACE1	012120	3492#	4351											
SPACE2	012117	3491#	4356	6042	6048	6415	6419	6422	6425	6445	6450	6465		
SPACE3	012116	3490#												
SPACE4	012115	3489#												
SPACE6	012113	3488#												
SPAR =	020000	2502#	6675	7658										
SPDLSS=	000020	2562#	6689											
SRHTBS	027656	6221#	6262											
SRPASS	010257	3308#	3774*	3775	4014*	4015								
SRTSPL=	000111	2466#	7945											
STACK =	001100	1335#	3522	5007*	5008	5050	5057	5122	5129	5799	8786			
STALL	024376	5531#	6305											
STALLS	005432	2965#	3583*	5540										
STDLS	006552	3120#	5148											
STKLMT=	177774	1346#												
STRCMD	030176	6309	6323#											
STR204	010420	3325#	4007*	4008										
STR220	010475	3333#	4017*	4018	4404*	4405								
SUBCLR=	000177	2483#	4508	4870	5012	6940	7022	8100						
SUBSYS	005510	2983#	3601*	3729*	3731	3802	4003	4400						
SVAL =	100000	2572#												
SVPRMS	027456	4095	4506	6157#	6203									
SWR	001140	1525#	3520	3541*	3543	3549*	3556*	4839	5091	5533	5731	5758	5918	5945

S.REV = 004000	2627#																			
S.RTZ = 020000	2629#																			
S.SECT= 000020	2634#																			
S.SKI = 002000	2611#																			
S.SPIN= 010000	2600#																			
S.SPLS= 010000	2613#																			
S.SPOK= 001000	2625#																			
S.TYPE= 000400	2596#																			
S.UNLD= 040000	2630#																			
S.UNS = 040000	2616#																			
S.VV = 000100	2594#																			
S.WCLK= 000040	2635#																			
S.WGAT= 000100	2636#																			
S.WLE = 004000	2612#																			
S.WRL = 004000	2599#	4967																		
S.XDOK= 000020	2620#																			
S.XERR= 001000	2639#																			
TAB 011213	3396#	4360																		
TABLEA 005566	2997#	3903	3905																	
TABLEB 005766	3037#	5559																		
TABLEC 006026	3058#	3946	3949	3965																
TABLED 006112	3079#	5149	5164	5178	5181	5201	5213	5233	5285											
TABLEG 006512	3099#	4087	5232																	
TABL7A 005666	3017#	3908	3910																	
TABL7C 006060	3068#	3952	3955	3968																
TBITVE= 000014	1432#																			
TKVEC = 000060	1439#	3571*	3572*																	
TOPROC 033752	6489	6660	6969#																	
TPVEC = 000064	1440#																			
TRACK 005440	2968#																			
TRAPVE= 000034	1438#	3528*	3529*	8785*																
TRKAVG 020232	4374	4625#																		
TRKBUF 011174	3392#	4331*	4334																	
TRNSFR 027546	4096	4507	6186#																	
TRTVEC= 000014	1433#																			
TSTDRN 011654	3454#																			
TSTING 003127	2917#	3576*	3894*	3997*	4076*	4322*	4394*	5785	5797*											
TSTMSG 045366	6399	8999#																		
TSTTYP 003143	2929#	3561*	4101*	4120*	4169*	4176*	4189*	4225*	4253*	4459	4478	4531	4548							
	4573	4583	5269	5275	5802*	6775	6820	6877	6897											
TST1 014316	3778	3866#	4006																	
TST10 015766	4220#																			
TST11 016404	4317#																			
TST2 014346	3889#																			
TST3 014540	3937#																			
TST4 014732	3992#																			
TST5 015116	3780	4016	4037#	4403																
TST6 015154	4071#																			
TST7 015460	4143#																			
TWOTOS= 002000	3246#	6661	6990																	
TYPBSF 027006	5093	6025#																		
TYPDS = 104405	8991#																			
TYPE = 104401	3566	3569	3612	3623	3632	3633	3668	3674	3682	3683	3689	3697	3709							
	3710	3715	3724	3725	3736	3742	3747	3756	3768	3770	3771	3775	4008							
	4015	4018	4327	4334	4338	4343	4351	4354	4356	4360	4365	4368	4381							
	4382	4387	4392	4405	4408	4740	4746	4750	4754	4756	4757	4788	4789							

\$DECBN	1#														
\$DOCTBN	1#	8143													
\$SCMRE	1498#	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550
	1551	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565
	1566	1567	1568												
\$SCMTM	1498#	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582
\$SESCA	1#	1442#													
\$SNEWT	1#	1442#	3854	3877	3928	3973	4024	4047	4124	4193	4297				
\$SSET	8979#	8988	8989	8990	8991	8994	8995	8996	8997						
\$SSETM	3553#														
\$SSKIP	1#	1442#													
.EQUAT	1#	1292#	1332												
.HEADE	1#	1292#	1307												
.KT11	1#														
.SETUP	1#	1292#	3501												
.SWRHI	1#	1292#	1319												
.SWRLO	1292#	1330#													
.SACT1	1#	1292#	1462												
.SAPTB	1#	1589#													
.SAPTH	1#	1292#	1473												
.SAPTY	1#	1292#	8854												
.SASTA	1#														
.SCATC	1#	1292#	1443												
.SCMTA	1#	1292#	1498												
.SDB2D	1#	1292#	8445												
.SDB2O	1#	1292#	8406												
.SDIV	1#														
.SEOP	1#	1292#	4418												
.SERRO	1#	1292#	8674												
.SERRT	1#														
.SMULT	1#														
.SPOWE	1#	1292#													
.SRAND	1#	1292#	8199												
.SRDDE	1#														
.SRDOC	1#														
.SREAD	1#	1292#	8726												
.SR2AZ	1#														
.SSAVE	1#	1292#	8911												
.SSB2D	1#														
.SSB2O	1#														
.SSCOP	1#	1292#	8811												
.SSIZE	1#	1292#													
.SSUPR	1#	1292#	8507												
.STRAP	1#	1292#	8956												
.STYPB	1#														
.STYPD	1#	1292#	8607												
.STYPE	1#	1292#	8235												
.STYPO	1#	1292#	8530												
.S4OCA	1#														
.1170	1#														

. ABS. 055151 000

ERRORS DETECTED: 0

CZR6QCO RK6 DR CPT PROG MACY11 30A(1052) 02-JUL-80 09:52 PAGE 217^{H 1}
CZR6QC.P11 02-JUL-80 09:50 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0213

DSKZ:CZR6QC.BIN,DSKZ:CZR6QC.LST/CRF/SOL/NL:TOC/EQ:QNEWSW=DSKZ:CZR6QC.SML,DRIV10.P11,CZR6QC.P11
RUN-TIME: 71 105 8 SECONDS
RUN-TIME RATIO: 314/186=1.6
CORE USED: 54K (107 PAGES)