

RK611/06

PERFORMANCE EXERCISER
CZR6PC0

AH-9145C-MC

COPYRIGHT ©76-78

FICHE 1 OF 2

MAR 1978

digital

MADE IN USA

RK611/06

PERFORMANCE EXERCISER
CZR6PC0

AH-9145C-MC

COPYRIGHT ©76-78

FICHE 2 OF 2

MAR 1978

digital

MADE IN USA

[Faint, illegible text from a microfiche grid, appearing as a series of small rectangular frames.]

EOF1CZR6LCSEQ
P02600001
CZR6PC.P11

00010000 780223
CZR6PCD RK611/06 PERF EXEC

PDP10 411
MACY11 30(1046) 01-DEC-77 15:04 PAGE 3

00010000

780223

01-DEC-77 14:53

.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-9144C-MC
PRODUCT NAME: CZR6PCD RK611/06 PERF EXERCISER
DATE: FEB 1978
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: ROY SPITZER

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

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

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

COPYRIGHT (C) 1976, 1978 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

- 1.0 ABSTRACT
- 2.0 REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 PRELIMINARY PREREQUISITS
- 3.0 OPERATING PROCEDURES
 - 3.1 LOADING PROCEDURES
 - 3.2 STARTING PROCEDURE
 - 3.3 SYSTEM PARAMETERS
 - 3.4 SWITCHES
 - 3.4.1 SWITCH 15
 - 3.4.2 SWITCH 14
 - 3.4.3 SWITCH 13
 - 3.4.4 SWITCH 12
 - 3.4.5 SWITCH 11
 - 3.4.6 SWITCH 10
 - 3.4.7 SWITCH 9
 - 3.4.8 SWITCH 8
 - 3.4.9 SWITCH 7
 - 3.4.10 SWITCH 6
 - 3.4.11 SWITCH 5
 - 3.4.12 SWITCH 4
 - 3.4.13 SWITCH 3
 - 3.4.14 SWITCH 2
 - 3.4.15 SWITCH 1
 - 3.4.16 SWITCH 0
 - 3.5 RUN TIME
- 4.0 OPERATING PROCEDURE
 - 4.1 OPERATOR COMMANDS
 - 4.2 DRIVE PARAMETER (PER EACH DRIVE)
 - 4.2.1 FORMS FOR PACK AREA EXCLUSIONS
 - 4.3 DIAGNOSTIC DUMP
- 5.0 PROGRAM DESCRIPTION
- 6.0 PRINT OUTS
 - 6.1 PERFORMANCE SUMMARY TYPEOUT
 - 6.2 ERROR REPORTS
- APPENDIX A - DATA WRITTEN ON PACKS

4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92

93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
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
1481.0 ABSTRACT

THE RK06-RK07 PERFORMANCE EXERCISER PROGRAM WILL EXERCISE IN A RANDOM OVERLAPPED MANNER 1 TO 8 RK06-RK07 DISK DRIVES ATTACHED TO THE SAME RK06-RK07 UNIBUS CONTROLLER IN A STAND ALONE MODE.

DRIVES UNDER TEST CAN BE ADDED TO OR DROPPED FROM THE TESTING SEQUENCE BY OPERATOR COMMAND.

AT ANY GIVEN POINT IN TIME, THE NEXT DRIVE ON WHICH AN OPERATION IS TO BE INITIATED IS CHOSEN RANDOMLY FROM AMONG THE RK06 DRIVES UNDER TEST NOT HAVING ANY ERROR CONDITIONS PRESENT AND NOT CURRENTLY UNDERGOING ERROR RECOVERY. THE COMMAND IS THEN CHOSEN RANDOMLY FROM THE FOLLOWING SET OF COMMANDS:

READ
WRITE
WRITE FOLLOWED BY WRITE CHECK

THEN THE CYLINDER, TRACK, SECTOR, WORD COUNT, AND DATA ARE DETERMINED RANDOMLY. THE DATA WRITTEN IS RANDOMLY SELECTED FROM A SET OF FIXED DATA PATTERNS. TO OPTIMIZE THE THROUGHPUT ON THE RK06 CONTROLLER EACH DATA TRANSFER COMMAND IS TRANSLATED TO AN EXPLICIT SEEK COMMAND FOLLOWED BY THE DATA TRANSFER COMMAND.

DRIVE ERRORS ARE QUEUED UP AS THEY OCCUR. WHILE ERROR RECOVERY IS BEING PROCESSED ON ONE DRIVE ALL OTHER DRIVES HAVING NO ERRORS WILL BE EXECUTING ORDERS GENERATED RANDOMLY AS DESCRIBED ABOVE. EACH DRIVE ERROR IS PROCESSED ON A FIRST-IN-FIRST-OUT BASIS. THIS MEANS THAT A DRIVE ERROR WILL BE REPORTED AND ERROR RECOVERY COMPLETELY PROCESSED BEFORE THE NEXT DRIVE ERROR IS REPORTED AND ERROR PROCESSING HAS BEGUN. ERROR RECOVERY IS PROCESSED AS DESCRIBED IN THE RK06 DISK DRIVE SPECIFICATION.

REPORTING OF SYSTEM ERRORS SUCH AS CONTROLLER PROBLEMS WILL BE IMMEDIATE UPON OCCURRENCE, NOT DEFERRED, PERHAPS CAUSING PREMATURE PROGRAM TERMINATION.

PERFORMANCE STATISTICS ARE KEPT ON EACH DRIVE. THESE STATISTICS INCLUDE BOTH OPERATION COUNTS AND ERROR SUMMARY INFORMATION. (SEE SECTION 6.1) THESE STATISTICS WILL BE INITIALIZED WHEN TESTING BEGINS ON A DRIVE. AT ANY TIME AFTER TESTING BEGINS ON THE DRIVE(S) UNDER PERFORMANCE EVALUATION, THE OPERATOR CAN DEMAND THESE PERFORMANCE STATISTICS. IF A REAL TIME CLOCK IS AVAILABLE, THE OPERATOR HAS THE OPTION OF HAVING THE PROGRAM GIVE PERIODIC PERFORMANCE SUMMARIES.

ONCE A PARTICULAR DRIVE HAS AN UNRECOVERABLE ERROR (OTHER THAN A DATA TRANSFER AND SEEK INCOMPLETE), THAT DRIVE WILL BE AUTOMATICLY DEASSIGNED.

ONCE A PARTICULAR DRIVE HAS EXCEEDED AN ERROR THRESHOLD (DATA TRANSFER OR SEEK) SPECIFIED FOR A DRIVE, THAT DRIVE WILL BE

EO1

CZR6PCO RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(10⁶) 01-DEC-77 15:04 PAGE 6

SEQ 0004

149

AUTOMATICLY DEASSIGNED UNLESS SWITCH 4 IS SET.

150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

ONCE A PARTICULAR DRIVE HAS EXCEEDED THE MAXIMUM NUMBER OF PERMITTED OPERATIONS FOR THAT DRIVE, THE DRIVE WILL BE AUTOMATICALLY DESELECTED UNLESS SWITCH 5 IS SET.

2.0 REQUIREMENTS

2.1 EQUIPMENT

- PDP-11 WITH AT LEAST 16K OF MEMORY
- CONSOLE TERMINAL
- DECTAPE, PAPER TAPE READER; OR DECDISK
- RK611 CONTROLLER
- 1 TO 8 RK06-RK07 DISK DRIVE WITH FORMATTED PACK (16 BIT FORMAT)
- REAL TIME CLOCK (KW11-P OR KW11-L OPTIONAL) FOR INTERVAL PERFORMANCE REPORTING AND TIME ASSOCIATIONS WITH ERROR REPORTS

2.2 PRELIMINARY PROGRAMS

THE RK06-RK07 SUBSYSTEM IS ASSUMED TO BE BASICALLY OPERATIONAL AND FREE OF HARD FAULTS. THE FOLLOWING RK06 PROGRAMS ARE ASSUMED TO BE RUNNING WITHOUT ERROR BEFORE THIS PROGRAM IS RUN:

- RK611 DISKLESS CONTROLLER DIAGNOSTICS
- RK06-RK07 DRIVE DIAGNOSTICS
- RK611 FUNCTIONAL CONTROLLER DIAGNOSTIC
- RK06-RK07 SUBSYSTEM VERIFICATION
- RK06-RK07 PACK FORMATTER (IF PACKS ARE UNFORMATTED)

201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
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

3.0 OPERATING PROCEDURE

3.1 LOADING PROCEDURE

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

3.2 STARTING PROCEDRES

AFTER THE PROGRAM IS LOADED, THE OPERATOR STARTS THE PROGRAM AT ONE OF THE FOLLOWING LOCATIONS:

LOCATION FUNCTION

200 INITIALIZE ALL DRIVE STATISTIC TABLES, THE RANDOM NUMBER SEED, AND USE DEFAULT SYSTEM PARAMETERS. THE PROGRAM WILL THEN DO AUTO-SIZING OF ALL DRIVES ON THE BUS & BEGIN BY FIRST WRITING THOSE DRIVES & THEN BEGIN PERFORMANCE TESTING.

204 RESTART PROGRAM. IF DRIVE IS UNDER TEST, CONTINUE TESTING DRIVE AND DO NOT DESTROY STATISTICS. IF PACK IS BEING WRITTEN, START WRITING PACK FROM THE BEGINNING. THE PROGRAM WILL THEN TYPE READY TO BEGIN PERFORMANCE TESTING.

214 START PROGRAM AND ALTER SYSTEM PARMETERS. THE PROGRAM WILL THEN IDENTIFY ITSELF AND ASK FOR THE PARAMETERS AS DESCRIBED IN SECTION 3.3.2. THE PROGRAM WILL THEN TYPE READY TO BEGIN PERFORMANCE TESTING.

240 INITIALIZE ALL DRIVE STATISTIC TABLES, THE RANDOM NUMBER SEED, AND USE DEFAULT SYSTEM PARAMETERS. THE PROGRAM WILL THEN TYPE READY TO BEGIN PERFORMANCE TESTING.

NOTE: IF A POWER FAIL OCCURS, THE PROGRAM WILL AUTOMATICALLY PERFORM THE RESTART FUNCTION.

257
258
259
260
261
262
263
264

THE PROGRAM IS NOW IN AN IDLE LOOP WAITING FOR THE INITIATION OF RK06 DRIVE TESTING.

THE OPERATOR THEN STARTS PERFORMANCE EVALUATION OF THE RK06 DRIVE(S) BY TYPING A CONTROL-C <C> FOLLOWED BY A TN OR PN AS DESCRIBED IN SECTION 4.1. IF THE PACK DOES NOT HAVE THE PROPER RANDOMLY CHOSEN

265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307

PREDETERMINED TEST PATTERNS. THE OPERATOR MUST USE THE WN COMMAND AS DESCRIBED IN SECTION 4.1 BEFORE INITIATING THE PERFORMANCE EVALUATION SEQUENCE ON THAT DRIVE.

3.3 SYSTEM PARAMETERS

3.3.1 DIRECT MEMORY ALTERATION

AS PART OF THE PROGRAM START-UP PROCEDURE, THE FOLLOWING SYSTEM PARAMETERS WILL BE LOADED BY THE OPERATOR OR DEFAULTED BY THE PROGRAM. THE OPERATOR MUST PHYSICALLY ALTER THE APPROPRIATE MEMORY LOCATIONS TO CHANGE THESE VALUES.

PARAMETER -----	TAG ---	DEFAULT VALUE -----
- KW11-P STATUS REGISTER ADDRESS	\$LKCSR	172540
- KW11-P COUNTER BUFFER ADDRESS	\$LKCSB	172542
- KW11-P VECTOR ADDRESS	\$LPVEC	104
- KW11-L STATUS REGISTER ADDRESS	\$LKS	177506
- KW11-L VECTOR ADDRESS	\$LLVEC	100
- SYSTEM POWER (HERTZ)	HZ	60(DECIMAL)

3.3.2 PROGRAM MODIFIED PARAMETERS

THE FOLLOWING PARAMETERS ARE MODIFIED BY A 214 PROGRAM START.

A <CR> WILL DEFAULT CURRENT PARAMETERS. A CONTROL Z <↑Z> <CR> WILL DEFAULT ALL THE REST OF THE PARAMETERS. A CONTROL C <↑C> <CR> WILL RETURN TO FIRST PARAMETER ENTRY REQUEST FOR THIS DRIVE. A RUBOUT CANCELS THE LAST CHARACTER TYPED. A CONTROL U <↑U> CANCELS LINE BEING TYPED. ALL ENTRIES MUST BE TERMINATED WITH CARRIAGE RETURN.

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

PARAMETER -----	DEFAULT VALUE -----
- CPU IDENTIFIER (OCTAL NUMBER 0-377 FOR DUAL ACCESS PORT DISCRIMINATION)	0
- OVERLAY OF LOADER BY PROGRAM? (ALLOWS GREATER EXPANSION OF MEMORY FOR BUFFER SPACE IF LOADER IS OVERLAYED)	NO
- MAXIMUM DATA BUFFER (SEE NOTE) UP TO 5888 (23 SECTORS)	SEE NOTE
- NUMBER OF DATA WORDS COMPARED PER READ (0 - MAXIMUM BUFFER LENGTH) EACH READ COMMAND ISSUED BY THE PROGRAM HAS AN IMPLICIT DATA COMPARISON ASSOCIATED WITH IT VERIFYING THAT THE DATA READ IS ONE OF LEGAL RANDOM PATTERNS.	3 WORDS
- INTERVAL BETWEEN AUTOMATIC PERFORMANCE TYPE-OUTS (EVERY 1-255 MINUTES OR NO AUTOMATIC PERFORMANCE TYPE-OUTS)	NO AUTOMATIC PERFORMANCE TYPE-OUTS
- RK611/RK06-RK07 UNIBUS ADDRESS	177440
- RK611/RK06-RK07 VECTOR ADDRESS	210
- RK611/RK06-RK07 PRIORITY	5

NOTE

THE PROGRAM WILL SELECT A MAXIMUM BUFFER SIZE WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAXIMUM BUFFER SIZE ASSIGNED BY THE PROGRAM WILL BE 5888 WORDS. THE DEFAULT VALUE OF THIS PARAMETER WEIGHTS THE OPERATIONS TOWARDS MAXIMIZING DATA TRANSFERS PER UNIT OF TIME. BY CHANGING THIS PARAMETER TO 1 SECTOR (256 WORDS), THE OPERATIONS WILL BE WEIGHTED TOWARDS MAXIMIZING SEEKS PER UNIT OF TIME.

350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402

3.4 SWITCHES

3.4.1 SWITCH 15

SW<15> = 1 HALT ON ERROR

IF THIS SWITCH IS SET, THE PROGRAM WILL NOT ISSUE ANOTHER COMMAND TO THE RK06 SUBSYSTEM WHEN AN ERROR OCCURS AND THE PROGRAM WILL HALT IN THE SYSMAC ERROR HANDLER. IF THE OPERATOR DESIRES TO CONTINUE FROM WHERE THE PROGRAM LEFT OFF, THE OPERATOR SIMPLY PASSES THE CONTINUE SWITCH ON THE CONSOLE PANEL AND THE PROGRAM CONTINUES WITH THE NEXT RANDOMLY GENERATED COMMAND.

3.4.2 SWITCH 14

SW<14> = 0 NORMAL OPERATION

THE OPERATION THE PROGRAM IN REGARD TO DRIVE SELECTION AND COMMAND GENERATION IS NORMAL. NORMAL OPERATION IS TO RANDOMLY SELECT A DRIVE, STORE THE PREVIOUS OPERATION EXECUTED ON THAT DRIVE, AND GENERATE A NEW RANDOM OPERATION TO BE EXECUTED ON THAT DRIVE. THE GENERATED OPERATION INCLUDES RANDOMLY GENERATED VALUES FOR THE FOLLOWING PARAMETERS:

- DRIVE COMMAND
- CYLINDER ADDRESS
- TRACK ADDRESS
- SECTOR ADDRESS
- WORD COUNT
- DATA PATTERN

THESE VALUES ARE STORED AS THE PREVIOUS OPERATION WHEN A NEW OPERATION IS GENERATED. IT IS IMPORTANT TO NOT THAT AT ANY POINT IN TIME THERE IS BOTH A CURRENT OPERATION AND A STORED PREVIOUS OPERATION ASSOCIATED WITH EACH DRIVE UNDER TEST. THE CURRENT OPERATION MAY HAVE ALREADY BEEN EXECUTED AND A NEW COMMAND HAS NOT YET BEEN GENERATED OR THE COMMAND IS WAITING TO BE EXECUTED.

403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453

SW<14> = 1

LOOP ON CURRENT OPERATIONS (TEST)

DO NOT GENERATE NEW RANDOM OPERATION BUT CONTINUE TO EXECUTE CURRENT OPERATIONS.

WHEN SWITCH 14 IS SET THE GENERATION OF A NEW OPERATION IS INHIBITED AND THE PROGRAM OPERATES AS FOLLOWS:

- 1.) A DRIVE IS RANDOMLY SELECTED.
- 2.) A SEEK IS EXECUTED ON THE DRIVE TO THE CYLINDER ADDRESS STORED IN THAT DRIVE'S PREVIOUS OPERATION.
- 3.) THE CURRENT OPERATION FOR THAT DRIVE IS EXECUTED.
- 4.) REPEAT THE ENTIRE PROCESS.

USAGE

THIS SWITCH CAN BE USED IN CONJUNCTION WITH SWITCH 15 TO ENABLE THE REPETITION OF A RANDOMLY GENERATED OPERATION. THE OPERATOR WOULD START WITH SWITCH 15 SET AND SWITCH 14 RESET. WHEN THE PROGRAM HAS INDICATED THAT AN ERROR HAS OCCURRED BY HALTING. THE CURRENT OPERATION FOR THE DRIVE IN ERROR WILL NOT HAVE BEEN MODIFIED WHEN THE PROGRAM HALTS. THE OPERATOR CAN THEN CAUSE ALL DRIVES TO LOOP ON THEIR INDIVIDUAL CURRENT OPERATIONS BY SETTING SWITCH 14, RESETTING SWITCH 15, AND HITTING CONTINUE.

TO HELP ISOLATE PROBLEMS, THE OPERATOR CAN DROP THE RK06 DRIVE(S) FROM THE TEST SEQUENCE AT ANY TIME BY USING THE DN COMMAND AS DESCRIBED IN SECTION 4.1.

3.4.3 SWITCH 13

SW<13> = 1

INHIBIT ERROR TYPE OUT

3.4.4 SWITCH 12

NO EFFECT.

455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506

3.4.5 SWITCH 11

NO EFFECT.

3.4.6 SWITCH 10

SW<10> = 1 RING TELETYPE BELL IF ERROR

3.4.7 SWITCH 9

NO EFFECT.

3.4.8 SWITCH 8

NO EFFECT.

3.4.9 SWITCH 7

NO EFFECT.

3.4.10 SWITCH 6

SW<6> = 1 INHIBIT AUTOMATIC DESELECT FOR CLEARABLE UNSAFES,
SPEED LOSS, AND AC LOW.

3.4.11 SWITCH 5

SW<5> = 1 INHIBIT AUTOMATIC DRIVE DEASSIGNMENT IF OPERATION
COUNT THRESHOLD IS EXCEEDED.

507
0008
110908
111008
111108
111208
111308
111408
111508
111608
111708
111808
111908
112008
112108
112208
112308
112408
112508
112608
112708
112808
112908
113008
113108
113208
113308
113408
113508
113608
113708
113808
113908
114008
114108
114208
114308
114408
114508
114608
114708
114808
114908
115008
115108
115208
115308
115408
115508
115608
115708
115808
115908
116008
116108
116208
116308
116408
116508
116608
116708
116808
116908
117008
117108
117208
117308
117408
117508
117608
117708
117808
117908
118008
118108
118208
118308
118408
118508
118608
118708
118808
118908
119008
119108
119208
119308
119408
119508
119608
119708
119808
119908
120008

3.4.12 SWITCH 4

SW<4> = 1 INHIBIT AUTOMATIC DRIVE DEASSIGNMENT IF ERROR THRESHOLD EXCEEDED.

3.4.13 SWITCH 3

SW<3> = 1 DISPLAY ENTIRE SECTOR READ BEFORE STARTING RETRY SEQUENCE

ALL HARDWARE DETECTED DATA ERRORS WILL BE RETRIED ON A SECTOR BASIS. IF THIS SWITCH IS SET, THE DATA SECTOR IN ERROR WILL BE DISPLAYED ENTIRELY BEFORE THE RETRY SEQUENCE BEGINS ON THE SECTOR IN ERROR.

3.4.14 SWITCH 2

NO EFFECT.

3.4.15 SWITCH 1

SW<1> = 1 INHIBIT SOFTWARE DATA COMPARISONS

THE IMPLICIT DATA COMPARISONS ASSOCIATED WITH EACH READ WILL NOT BE PERFORMED WHEN THIS SWITCH IS SET. THE ONLY DATA CHECKING PERFORMED WILL BE HARDWARE DATA CHECKING.

3.4.16 SWITCH 0

NO EFFECT.

3.5 RUN TIME

THIS PROGRAM IS DESIGNED TO RUN FOR AN EXTENDED PERIOD OF TIME. (ESTIMATED RUN TIME WITH PRESENT DEFAULT PARAMETERS IS APPROXIMATELY 250 HOURS PER DRIVE.) MEANINGFUL INFORMATION CAN STILL BE DERIVED BY

B02

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 16

SEQ 0014

563
564

RUNNING THIS PROGRAM FOR AN HOUR OR LESS SINCE STATISTICS ARE TAKEN AS
THE PROGRAM IS RUNNING.

565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
6204.0 OPERATING PROCEDURE
-----4.1 OPERATOR COMMANDS

AFTER THE PROGRAM IS LOADED AND HAS PRINTED OUT "READY TO BEGIN PERFORMANCE TESTING", THE OPERATOR CAN ONLY TYPE A CONTROL-G <G> OR A CONTROL-C <C>.

A CONTROL-G <G> IS USED FOR SOFTWARE SWITCH REGISTER MODIFICATION.

UPON RECEIVING A CONTROL-C <C> THE PROGRAM WILL RESPOND WITH "PLEASE TYPE COMMAND". THE FOLLOWING TWO CHARACTER COMMANDS FOLLOWED BY CARRIAGE RETURN WILL BE THE ONLY COMMANDS ACCEPTED: TN, PN, DN, SN, WN.

THE SECOND CHARACTER N=0,1,...,7 OR A DESIGNATES WHICH DRIVE IS REFERENCED BY THE COMMAND. IF N=A, ALL ADDRESSABLE DRIVES ARE REFERENCED. THE DESCRIPTION OF EACH OPERATOR COMMAND FOLLOWS:

- TN INITIATE TESTING OF DRIVE N AND USE PREVIOUS PARAMETERS. (DEFAULT PARAMETERS ARE USED IF PN COMMAND HAS NOT PREVIOUSLY BEEN ISSUED TO THE DRIVE(S) AT THE BEGINNING OF THEIR TEST SEQUENCE.) IF N=A, INITIATE TESTING ON ALL DRIVES.

- PN THIS COMMAND HAS TWO USES:

1.) CHANGE CURRENT DRIVE PARAMETERS AND INITIATE TESTING ON DRIVE N.

2.) CHANGE CURRENT DRIVE PARAMETERS AND CONTINUE TESTING ON DRIVE N.

IN BOTH CASES THE PROGRAM WILL THEN ASK FOR PARAMETERS IN THE ORDER AS DESCRIBED IN SECTION 5.5. A <CR> INDICATES DEFAULT THIS PARAMETER AND A CONTROL-Z <Z> INDICATES THE REST OF THE PARAMETERS FOR THIS DRIVE ARE TO BE DEFAULTED.

IF N=A, PERFORM PN COMMAND ON ALL DRIVES.

- DN DROP DRIVE FROM TESTING SEQUENCE AND PRINT PERFORMANCE SUMMARY. IF N=A, DROP ALL CURRENTLY BEING TESTED DRIVES FROM TESTING SEQUENCE

621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676

- SN DEMAND PERFORMANCE SUMMARY ON DRIVE N. IF N=A DEMAND PERFORMANCE SUMMARY ON ALL CURRENTLY BEING TESTED DRIVES.

- WN WRITE AND VERIFY THE RK06 DISK PACK ON DRIVE N WITH A RANDOM SET OF LEGAL PATTERNS. THIS IS USED PRIOR TO THE START OF THE TEST SEQUENCE TO GUARENTEE THAT THE PROPER PREDETERMINED TEST PATTERNS HAVE BEEN WRITTEN ON THE PACK. (SEE APPENDIX A FOR DETAILS) AFTER THE PACK HAS BEEN COMPLETELY WRITTEN TESTING WILL BEGIN USING THE PREVIOUS PARAMETERS AS DESCRIBED IN THE TN COMMAND.

WRITING OF THE PACKS WILL TAKE 4 TO 5 MINUTES PER PACK WITH MAXIMUM BUFFER SIZE. THE WRITING OF THE PACK MAY TAKE LONGER IF THE MAXIMUM BUFFER SIZE SPECIFIED IS SMALL. RANDOM EXERCISE CAN BE DONE ON SOME DRIVES WHILE EXERCISE ON OTHER DRIVES ARE IN PROGRESS.

NOTE: IF AN ERROR OCCURS DURING THE DRIVE ASSIGNMENT SEQUENCE OR WHILE WRITING THE RANDOM SET OF LEGAL PATTERNS, THE ERROR WILL BE REPORTED AND TESTING WILL NOT START.

EXAMPLES:

COMMAND SEQUENCE	ACTION TAKEN
TA<CR>	INITIATE TESTING ON ALL DRIVES ON SYSTEM
T1<CR>	INITIATE TESTING ON DRIVE 1
T3<CR>	INITIATE TESTING ON DRIVE 3 (TESTING STILL CONTINUES ON DRIVE 1)
SA<CR>	GATHER STATISTICS ON ALL DRIVES UNDER TEST.
DA<CR>	DROP ALL DRIVES UNDER TEST.

4.2 DRIVE PARAMETERS (PER EACH DRIVE)

A SET OF THESE PARAMETERS EXISTS FOR EACH DRIVE. A TN COMMAND WILL UTILIZE THE PREVIOUSLY ESTABLISHED PARAMETERS FOR THE DRIVE(S). A PN COMMAND WILL ALLOW THE OPERATOR TO ALTER ALL THE PARAMETERS FOR THE DRIVE(S).

A <CR> WILL DEFAULT CURRENT PARAMETERS. A CONTROL Z <↑Z> <CR> WILL DEFAULT ALL THE REST OF THE PARAMETERS. A CONTROL C <↑C> <CR> WILL RETURN TO FIRST PARAMETER ENTRY REQUEST. FOR THIS DRIVE. A RUBOUT

E02

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 19

SEQ 0017

677
678

CANCELS THE LAST CHARACTER TYPED. A CONTROL U <↑U> CANCELS LINE BEING
TYPED. ALL ENTRIES MUST BE TERMINATED WITH CARRIAGE RETURN.

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
717
718
719
720
721
722
723

PARAMETER	DEFAULT VALUE (RK06/RK07)
- MAXIMUM CYLINDER FOR START OF DATA TRANSFER 0 - 632/1456	632/1456
- MINIMUM CYLINDER FOR START OF DATA TRANSFER 0 - 632/1456	0
- MAXIMUM TRACK FOR START OF DATA TRANSFER 0 - 2	2
- MINIMUM TRACK FOR START OF DATA TRANSFER 0 - 2	0
- MAXIMUM SECTOR FOR START OF DATA TRANSFER 0 - 25	25
- MINIMUM SECTOR FOR START OF DATA TRANSFER 0 - 25	0
- RATIO OF READ/WRITE EVERY COMMAND RANDOMLY GENERATED BY THE PROGRAM WILL BE EITHER A READ OR A WRITE. FOR EACH DRIVE ONE CAN WEIGHT THE COMMANDS TOWARDS READING OR WRITING. THE POSSIBLE WEIGHTING RATIO'S ARE: READ ONLY =0 7/1 =1 3/1 =2 5/3 =3 1/1 =4 3/5 =5 1/3 =6 1/7 =7	5/3
- DO WRITE CHECK AFTER EVERY WRITE COMMAND OTHERWISE, WRITE CHECK COMMANDS WILL BE ISSUED RANDOMLY AFTER WRITE COMMANDS.	NO
- CORRECTABLE READ ERROR THRESHOLD	10
- UNCORRECTABLE READ ERROR THRESHOLD	5
- SEEK ERROR THRESHOLD	5
- INHIBIT ERROR CORRECTION	NO
- OPERATION COUNT THRESHOLD*65K	4.290*10**9
- DATA TRANSFER THRESHOLD*520K WORDS	291*10**12
- SAMPLED COMPARES	YES
- UP TO 5 EXCLUDED PACK AREAS (NO DATA TRANSFER WILL BE DONE TO THESE AREAS.)	NO EXCLUDED PACK AREAS

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
773
774
775
776
777
778
779

4.2.1 FORMATS FOR PACK AREA EXCLUSION

FORMAT -----	ACTION -----
CYL<CR>	ALL TRACK ON CYLINDER SPECIFIED WILL BE EXCLUDED.
CYL,TRK<CR>	THE ENTIRE TRACK WILL BE EXCLUDED.
CYL1,TRK1,CYL2,TRK2<CR>	ALL TRACKS FROM CYLINDER 1 TRACK 1 TO AND INCLUDING CYLINDER 2 TRACK 2 WILL BE EXCLUDED.

4.3 DIAGNOSTIC DUMP

DUE TO THE COMPLEXITY OF THIS PROGRAM A DUMP OF THE PROGRAM TABLES MUST ACCOMPANY EACH PROBLEM REPORT. THIS DUMP IS PROVIDED IN THE PROGRAM AND IS ACTIVATED BY STARTING THE PROGRAM AT LOCATION 220. THIS DUMP WILL PRINT THE PROCESSOR REGISTERS WHEN THE DUMP IS CALLED. IT WILL THEN PRINT ABOUT 12-15 TELETYPE PAGES OF SOFTWARE TABLES NECESSARY TO DETERMINE WHAT THE PROGRAM WAS DOING AT A GIVEN TIME. APPROXIMATE TIME OF DUMP IS 10-12 MINUTES.

5.0 PROGRAM DESCRIPTION

THIS PROGRAM ASSUMES THAT ONLY THE PATTERNS WRITTEN BY THIS PROGRAM WILL BE ON THE PACK. TO PRECONDITION THE PACK A WN COMMAND MUST BE DONE. PACK SERIAL, DRIVE SERIAL NUMBER, THE TIME, AND DRIVE N UNDER TEST WILL BE PRINTED AT THE END OF PACK PRECONDITIONING. THE OPERATION OF THE PROGRAM IS DEPENDENT ON THE SWITCH REGISTER, SYSTEM PARAMETERS, AND DRIVE PARAMETERS. DRIVES MAY BE DROPPED FROM TEST BY USING THE DN COMMAND. IF A DRIVE IS DROPPED FOR ANY REASON, PERFORMANCE STATISTICS WILL AUTOMATICALLY BE PRINTED.

5.1 PROGRAM RESTRICTIONS

THIS PROGRAM WILL NOT RUN IN CHAIN MODE UNDER XXDP. NO END-OF-PASS INDICATOR IS PROVIDED FOR ACT. THE APT PROTOCOL HAS NOT BEEN

CZR6PCO RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 22

H02

SEQ 0020

780

DETERMINED YET.

5

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
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830

6.0 PRINT OUTS

6.1 PERFORMANCE SUMMARY TYPEOUT

THE PERFORMANCE SUMMARY WILL CONTAIN THE FOLLOWING INFORMATION:

DRIVE NUMBER
DRIVE SERIAL NUMBER
TIME OF REPORT (IF REAL TIME CLOCK IS AVAILIABLE)
NUMBER OF ORDERS PERFORMED BY DRIVE
NUMBER OF SEEK OPERATIONS PERFORMED
TOTAL NUMBER OF WORDS WRITTEN BY DRIVE*65K
TOTAL NUMBER OF WORDS READ BY DRIVE*65K
NUMBER OF SOFT DATA ERRORS
(ECC CORRECTABLE)
(REREAD CORRECTABLE)
(OFFSET CORRECTABLE)
NUMBER OF HARD DATA ERRORS
NUMBER OF SEEK INCOMPLETES
NUMBER OF MISPOSITIONING ERRORS
TOTAL NUMBER OF ALL OTHER ERRORS.

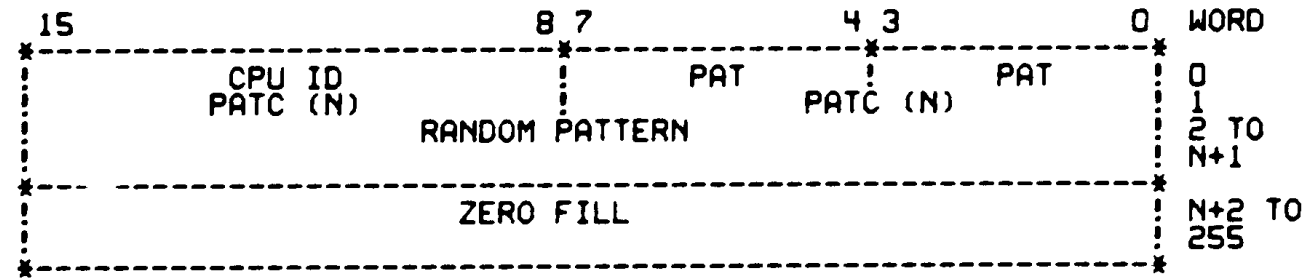
6.2 ERROR REPORTS

THE ERROR REPORTS WILL CONTAIN THE FOLLOWING INFORMATION:

DRIVE NUMBER
DRIVE SERIAL NUMBER
TIME ERROR OCCURRED (IF REAL TIME CLOCK IS AVAILIABLE)
DESCRIPTION OF ERROR
PRESENT COMMAND
PREVIOUS COMMAND
PRESENT POSITION
PREVIOUS POSITION
ALL DRIVE STATUS REGISTERS
ALL CONTROLLER REGISTERS
MEMORY ADDRESS, GOOD DATA AND BAD DATA (IF DATA ERROR)
OFFSET VALUE (IF APPLICABLE)
HEADER READ (IF MISPOSITIONING ERROR)
ECC PATTERN AND POSITION REGISTER IF DATA CHECKED

APPENDIX A

FIGURE A-1 SHOWS THE DATA EXPECTED BY THIS PROGRAM FOR EACH SECTOR OF THE RK06 PACK USED BY THIS PROGRAM.



CPU ID CPU IDENTIFIER SUPPLIED BY THE OPERATOR
 PAT PATTERN USED
 PATC PATTERN COUNT (NUMBER OF WORDS OF PATTERN WRITTEN)

FIGURE A-1
 EXPECTED DATA FORMAT

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

DATA PATTERNS USED

866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905

PATTERN 0	PATTERN 1	PATTERN 2	PATTERN 3
165555	026455	007417	052525
133333	151322	170360	125252
165555	026455	007417	052525
133333	026455	007417	052525
165555	151322	170360	125252
133333	151322	170360	125252
165555	026455	007417	052525
133333	026455	007417	052525
165555	026455	007417	052525
133333	151322	170360	125252
165555	151322	170360	125252
133333	151322	170360	125252
165555	026455	007417	052525
133333	026455	007417	052525
165555	151322	170360	125252
133333	151322	170360	125252
165555	151322	170360	125252
133333	026455	007417	052525
165555	026455	007417	052525
133333	151322	170360	125252
165555	151322	170360	125252
133333	151322	170360	125252
165555	026455	007417	052525
133333	026455	007417	052525
165555	151322	170360	125252
133333	151322	170360	125252
165555	026455	007417	052525
133333	151322	170360	125252

906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942

	PATTERN 4	PATTERN 5	PATTERN 6	PATTERN 7
	000000	000000	000001	000001
	010421	177777	000003	000002
	021042	000000	000007	000004
	031463	000000	000017	000010
	042104	177777	000037	000020
	052525	177777	000077	000040
	063146	000000	000177	000100
	073567	000000	000377	000200
	104210	000000	000777	000400
	114631	177777	001777	001000
	125252	177777	003777	002000
	135673	177777	007777	004000
	146314	000000	017777	010000
	156735	000000	037777	020000
	167356	000000	077777	040000
	177777	000000	177777	100000
	177777	177777	177776	100000
	167356	177777	177774	040000
	156735	177777	177770	020000
	146314	177777	177760	010000
	135673	000000	177740	004000
	125252	000000	177700	002000
	114631	000000	177600	001000
	104210	177777	177400	000400
	073567	177777	177000	000200
	063146	177777	176000	001000
	052525	000000	174000	000040
	042104	000000	170000	000020
	031463	177777	160000	000010
	021042	177777	140000	000004
	010421	000000	100000	000002
	000000	177777	000000	000001

943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979

PATTERN 10	PATTERN 11	PATTERN 12	PATTERN 13
177776	172666	153333	000000
177775	155555	066667	177777
177773	172666	153333	177777
177767	155555	066667	177777
177757	172666	153333	177777
177737	155555	066667	177777
177677	172666	153333	177777
177577	155555	066667	177777
177377	172666	153333	177777
176777	155555	066667	177777
175777	172666	153333	177777
173777	155555	066667	177777
167777	172666	153333	177777
157777	155555	066667	177777
137777	172666	153333	177777
077777	155555	066667	177777
077777	172666	153333	177777
137777	155555	066667	177777
157777	172666	153333	177777
167777	155555	066667	177777
173777	172666	153333	177777
175777	155555	066667	177777
176777	172666	153333	177777
177377	155555	066667	177777
177577	172666	153333	177777
177677	155555	066667	177777
177737	172666	153333	177777
177757	155555	066667	177777
177767	172666	153333	177777
177773	155555	066667	177777
177775	172666	153333	177777
177776	155555	066667	177777

	PATTERN 14	PATTERN 15	PATTERN 16	PATTERN 17
980	177777	172304	070627	133467
981	000000	172304	113431	133467
982	000000	172304	014561	133467
983	000000	172304	070627	133467
984	000000	172304	113431	133467
985	000000	172304	014561	133467
986	000000	172304	070627	133467
987	000000	172304	113431	133467
988	000000	172304	014561	133467
989	000000	172304	070627	133467
990	000000	172304	113431	133467
991	000000	172304	014561	133467
992	000000	172304	070627	133467
993	000000	172304	113431	133467
994	000000	172304	014561	133467
995	000000	172304	070627	133467
996	000000	172304	113431	133467
997	000000	172304	014561	133467
998	000000	172304	070627	133467
999	000000	172304	113431	133467
1000	000000	172304	014561	133467
1001	000000	172304	070627	133467
1002	000000	172304	113431	133467
1003	000000	172304	014561	133467
1004	000000	172304	070627	133467
1005	000000	172304	113431	133467
1006	000000	172304	014561	133467
1007	000000	172304	070627	133467
1008	000000	172304	113431	133467
1009	000000	172304	014561	133467
1010	000000	172304	070627	133467
1011	000000	172304	113431	133467
1012	000000	172304	014561	133467
1013	000000	172304	070627	133467
1014	000000	172304	113431	133467
1015	000000	172304	014561	133467
1016	000000	172304	070627	133467
1017	000000	172304	113431	133467

1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061

; *** REV 005 ***

.TITLE CZR6PCD RK611/06 PERF EXEC
*COPYRIGHT (C) 1976,1977
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY ROY SPITZER
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
*

* THE RK06-RK07 PERFORMANCE EXERCISER WILL EXERCISE 1 TO 8
* RK06-RK07 DISK DRIVES ATTACHED TO THE SAME RK06
* UNIBUS CONTROLLER IN A DEDICATED STAND ALONE MODE.

* UNDER NORMAL OPERATION SITUATIONS, THE DRIVE IS
* CHOSEN RANDOMLY FROM ANY ONE OF THE RK06 DRIVES UNDER TEST
* WHICH IS NOT CURRENTLY UNDERGOING ERROR RECOVERY.
* TO OPTIMIZE THE THROUGHOUT ON THE RK06 CONTROLLER EACH
* DATA TRANSFER COMMAND IS TRANSLATED INTO A SEEK FOLLOWED
* BY THE DATA TRANSFER COMMAND.

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
10	BELL ON ERROR
6	INHIBIT AUTOMATIC DRIVE DEASSIGNMENT IF UNSAFE, AC LOW, OR SPEED LOSS OCCURS
5	INHIBIT AUTOMATIC DRIVE DEASSIGNMENT WHEN MAXIMUM COMMAND COUNT FOR DRIVE IS EXCEEDED
4	INHIBIT AUTOMATIC DRIVE DEASSIGNMENT WHEN ERROR THRESHOLD EXCEEDED
3	DUMP ENTIRE SECTOR BEFORE BEGINNING RETRY SEQUENCE
1	INHIBIT SOFTWARE DATA COMPARISONS

```

1062      .SBTTL BASIC DEFINITIONS
1063
1064      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1065      001100      STACK= 1100
1066      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
1067      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
1068
1069      ;*MISCELLANEOUS DEFINITIONS
1070      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
1071      000012      LF= 12      ;;CODE FOR LINE FEED
1072      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
1073      000200      CRLF= 200   ;;CODE FOR CARRIAGE RETURN-LINE FEED
1074      177776      PS= 177776  ;;PROCESSOR STATUS WORD
1075      .EQUIV PS,PSW
1076      177774      STKLMT= 177774 ;;STACK LIMIT REGISTER
1077      177772      PIRG= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
1078      177570      DSWR= 177570 ;;HARDWARE SWITCH REGISTER
1079      177570      DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
1080
1081      ;*GENERAL PURPOSE REGISTER DEFINITIONS
1082      000000      R0= %0      ;;GENERAL REGISTER
1083      000001      R1= %1      ;;GENERAL REGISTER
1084      000002      R2= %2      ;;GENERAL REGISTER
1085      000003      R3= %3      ;;GENERAL REGISTER
1086      000004      R4= %4      ;;GENERAL REGISTER
1087      000005      R5= %5      ;;GENERAL REGISTER
1088      000006      R6= %6      ;;GENERAL REGISTER
1089      000007      R7= %7      ;;GENERAL REGISTER
1090      000006      SP= %6      ;;STACK POINTER
1091      000007      PC= %7      ;;PROGRAM COUNTER
1092
1093      ;*PRIORITY LEVEL DEFINITIONS
1094      000000      PR0= 0      ;;PRIORITY LEVEL 0
1095      000040      PR1= 40     ;;PRIORITY LEVEL 1
1096      000100      PR2= 100   ;;PRIORITY LEVEL 2
1097      000140      PR3= 140   ;;PRIORITY LEVEL 3
1098      000200      PR4= 200   ;;PRIORITY LEVEL 4
1099      000240      PR5= 240   ;;PRIORITY LEVEL 5
1100      000300      PR6= 300   ;;PRIORITY LEVEL 6
1101      000340      PR7= 340   ;;PRIORITY LEVEL 7
1102
1103      ;*"SWITCH REGISTER" SWITCH DEFINITIONS
1104      100000      SW15= 100000
1105      040000      SW14= 40000
1106      020000      SW13= 20000
1107      010000      SW12= 10000
1108      004000      SW11= 4000
1109      002000      SW10= 2000
1110      001000      SW09= 1000
1111      000400      SW08= 400
1112      000200      SW07= 200
1113      000100      SW06= 100
1114      000040      SW05= 40
1115      000020      SW04= 20
1116      000010      SW03= 10
1117      000004      SW02= 4

```

1118 000002
 1119 000001
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130
 1131
 1132 100000
 1133 040000
 1134 020000
 1135 010000
 1136 004000
 1137 002000
 1138 001000
 1139 000400
 1140 000200
 1141 000100
 1142 000040
 1143 000020
 1144 000010
 1145 000004
 1146 000002
 1147 000001
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160 000004
 1161 000010
 1162 000014
 1163 000014
 1164 000014
 1165 000020
 1166 000024
 1167 000030
 1168 000034
 1169 000060
 1170 000064
 1171 000240
 1172 000240
 1173 120210

SW01= 2
 SW00= 1
 .EQUIV SW09, SW9
 .EQUIV SW08, SW8
 .EQUIV SW07, SW7
 .EQUIV SW06, SW6
 .EQUIV SW05, SW5
 .EQUIV SW04, SW4
 .EQUIV SW03, SW3
 .EQUIV SW02, SW2
 .EQUIV SW01, SW1
 .EQUIV SW00, SW0

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

.*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4 ; TIME OUT AND OTHER ERRORS
 RESVEC= 10 ; RESERVED AND ILLEGAL INSTRUCTIONS
 TBITVEC= 14 ; "T" BIT
 TRTVEC= 14 ; TRACE TRAP
 BPTVEC= 14 ; BREAKPOINT TRAP (BPT)
 IOTVEC= 20 ; INPUT/OUTPUT TRAP (IOT) **SCOPE**
 PWRVEC= 24 ; POWER FAIL
 EMTVEC= 30 ; EMULATOR TRAP (EMT) **ERROR**
 TRAPVEC= 34 ; "TRAP" TRAP
 TKVEC= 60 ; TTY KEYBOARD VECTOR
 TPVEC= 64 ; TTY PRINTER VECTOR
 PIRQVEC= 240 ; PROGRAM INTERRUPT REQUEST VECTOR
 APRIOR= PR5 ; RK611 PRIORITY
 AVECT1= 120210 ; RK611 VECTOR

```

1174      177440      ABASE= 177440      ;RK611 BASE
1175      000114      MEMVEC= 114      ;VECTOR FOR MEMORY CHECK ENABLE
1176      172100      MEMBAS= 172100    ;BUS ADDRESS FOR MEMORY CHECK ENABLE
1177      000001      PAR.EN= 1      ;MEMORY ENABLE MEMORY CHECKING
1178
1179      .SBTTL  RK06-PK07 CONTROLLER REGISTER DEFINITION
1180
1181      000000      RKCS1= 0      ;CONTROL AND STATUS REGISTER 1
1182      000002      RKWC= 2      ;WORD COUNT REGISTER
1183      000004      RKBA= 4      ;BUS ADDRESS REGISTER
1184      000006      RKDA= 6      ;DESIRED TRACK SECTOR REGISTER
1185      000010      RKCS2= 10     ;CONTROL AND STATUS REGISTER 2
1186      000012      RKDS= 12     ;DRIVE STATUS REGISTER
1187      000014      RKER= 14     ;ERROR REGISTER
1188      000016      RKASOF= 16    ;ATTENTION SUMMARY AND OFFSET REGISTER
1189      000020      RKOC= 20     ;DESIRED CYLINDER REGISTER
1190      000020      RDCYL= 20    ;DESIRED CYLINDER REGISTER
1191      000024      B= 24      ;DATA BUFFER
1192      000026      R1= 26      ;MAINTENANCE REGISTER 1
1193      000034      R2= 34      ;MAINTENANCE REGISTER 2
1194      000036      R3= 36      ;MAINTENANCE REGISTER 3
1195      000030      RKPOS= 30    ;ECC POSITION INFORMATION
1196      000030      RKECPS= 30   ;ECC POSITION INFORMATION
1197      000032      RKPAT= 32    ;ECC PATTERN INFORMATION
1198      000032      RKECPT= 32   ;ECC PATTERN INFORMATION
1199
1200      .SBTTL  DRIVE COMMANDS
1201
1202      000101      SELDRV= 101   ;SELECT DRIVE
1203      000103      PACK= 103   ;PACK ACKNOWLEDGE
1204      000105      CLEAR= 105  ;DRIVE CLEAR
1205      000107      UNLOAD= 107  ;UNLOAD
1206      000111      SRTSPL= 111 ;START SPINDLE
1207      000113      RECAL= 113  ;RECALIBRATE
1208      000115      OFFSET= 115 ;OFFSET
1209      000117      SEEK= 117   ;SEEK
1210      000121      RDDATA= 121  ;READ DATA
1211      000123      WRDATA= 123 ;WRITE DATA
1212      000125      RDHEAD= 125 ;READ HEADER
1213      000127      WRHEAD= 127 ;WRITE HEADER AND DATA
1214      000131      WRTCHK= 131 ;WRITE CHECK
1215
1216      ;          THE FOLLOWING ARE NOT DRIVE COMMANDS BUT ARE USED BY THE DRIVER
1217      ;          TO SIMULATE A SPECIFIC DESIRED OPERATION
1218
1219      000140      RELEAS= 140   ;RELEASE DRIVE
1220      000141      RDSTAT= 141  ;GET ALL STATUS FROM DRIVE
1221      000164      RDALHD= 164  ;READ ALL HEADERS
1222      000176      CONCLR= 176  ;CONTROLLER CLEAR (BIT 15 OF CS1)
1223      000177      SUBCLR= 177 ;SUBSYSTEM CLEAR (BIT 5 OF CS2)
1224      000300      INTR= 300   ;GENERATE INTERRUPT TO CPU
1225
1226      ;          DRIVER ISSUED SERVICE COMMANDS
1227
1228      000001      DR.SEL= 001   ;DRIVE SELECT
1229      000005      DR.CLR= 005   ;DRIVE CLEAR

```



```

1230
1231
1232
1233          000001      GO=      BIT0          ;GO BIT
1234          000100      IE=      BIT6          ;INTERRUPT ENABLE
1235          000200      RDY=     BIT7          ;CONTROLLER READY
1236          000400      BA16=    BIT8          ;BUS ADDRESS BIT 16
1237          001000      BA17=    BIT9          ;BUS ADDRESS BIT 17
1238          002000      CDT=     BIT10         ;CONTROLLER DRIVE TYPE (0=RK06, 1=RK07)
1239          004000      CTO=     BIT11         ;CONTROLLER TIMED OUT WAITING FOR
1240                                     ;DRIVE RESPONSE
1241          010000      CFMT=    BIT12         ;CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)
1242          020000      SPAR=    BIT13         ;DRIVE BUS PARITY ERROR DETECTED BY CONTROLLER
1243          040000      DI=      BIT14         ;DRIVE INTERRUPT
1244          100000      CERR=    BIT15         ;CONTROLLER ERROR
1245          100000      CCLR=    BIT15         ;CONTROLLER CLEAR
1246
1247          ;           ; THESE BIT DEFINITIONS ARE USED FOR ADDRESS
1248          ;           ; THE HIGH BYTE OF RKCS1
1249
1250          000001      B.BA16=  BIT0          ;BUS ADDRESS BIT 16
1251          000002      B.BA17=  BIT1          ;BUS ADDRESS BIT 17
1252          000004      B.CDT=   BIT2          ;CONTROLLER DRIVE TYPE (0=RK06, 1=RK07)
1253          000020      B.CFMT=  BIT4          ;CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)
1254
1255          .SBTTL CONTROL AND STATUS REGISTER 2 BITS
1256
1257          000007      DRVMSK=  7           ;MASK FOR DRIVE SELECTION CODE
1258          000010      DESL=    BIT3          ;DESELECT OR RELEASE DRIVE IN BITS 0-2
1259          000010      RLS=    BIT3          ;DESELECT OR RELEASE DRIVE IN BITS 0-2
1260          000020      BAI=    BIT4          ;BUS ADDRESS INCREMENT INHIBIT
1261          000040      CLR=    BIT5          ;CLEAR CONTROLLER AND ALL DRIVES
1262          000040      SCLR=   BIT5          ;CLEAR CONTROLLER AND ALL DRIVES
1263          000100      IR=     BIT6          ;INPUT READY
1264          000200      OR=     BIT7          ;OUTPUT READY
1265          000400      UFE=    BIT8          ;UNIT FIELD ERROR
1266          001000      MDS=    BIT9          ;MULTIPLE DRIVE SELECT
1267          002000      PGE=    BIT10         ;PROGRAMMING ERROR
1268          004000      NEM=    BIT11         ;NON-EXISTENT MEMORY
1269          010000      NED=    BIT12         ;NON-EXISTENT DRIVE
1270          020000      UPE=    BIT13         ;UNIBUS PARITY ERROR
1271          040000      WCE=    BIT14         ;WRITE CHECK ERROR
1272          100000      DLT=    BIT15         ;DATA LATE ERROR
1273
1274          .SBTTL ERROR REGISTER BIT DEFINITION
1275
1276          000001      ILC=     BIT0          ;ILLEGAL FUNCTION CODE
1277          ;*ILF=    BIT0          ;ILLEGAL FUNCTION CODE
1278          000002      SKI=    BIT1          ;SEEK INCOMPLETE
1279          000004      ILF=    BIT2          ;ILLEGAL DRIVE FUNCTION
1280          000004      NXF=    BIT2          ;ILLEGAL DRIVE FUNCTION
1281          000010      DRPAR=   BIT3          ;DRIVE DETECTED DRIVE BUS PARITY ERROR
1282          000020      FMTE=   BIT4          ;FORMAT ERROR
1283          000040      DTYE=   BIT5          ;DRIVE TYPE ERROR
1284          000100      ECH=    BIT6          ;ECC HARD
1285          000200      BSE=    BIT7          ;BAD SECTOR ERROR

```

```

1286      000400      HCRC= BIT8      ;HEADER CRC ERROR
1287      000400      HVRC= BIT8      ;HEADER VRC ERROR
1288      001000      COE= BIT9      ;CYLINDER ADDRESS OVERFLOW ERROR
1289      002000      IDAE= BIT10     ;INVALID DISK ADDRESS ERROR
1290      004000      WLE= BIT11     ;WRITE LOCK ERROR
1291      010000      DTE= BIT12     ;DRIVE TIMING ERROR
1292      020000      OPI= BIT13     ;OPERATION (SEARCH) INCOMPLETE
1293      040000      UNS= BIT14     ;DRIVE UNSAFE
1294      100000      DCK= BIT15     ;DATA CHECK
1295
1296      .SBTTL STATUS REGISTER BIT DEFINITION
1297
1298      000001      DRA= BIT0      ;DRIVE AVAILABLE (CONTROLLER IS SET IF
1299      ; THIS BIT IS RESET)
1300      000004      OFST= BIT2      ;DRIVE OFFSET
1301      000010      ACLO= BIT3      ;AC LOW
1302      000020      SPDLSS= BIT4     ;SPEED LOSS
1303      000020      DCLO= BIT4      ;DC LOW
1304      000040      DROT= BIT5      ;DRIVE OFF TRACK
1305      000100      VV= BIT6      ;VOLUME VALID
1306      000200      DRY= BIT7      ;DRIVE READY
1307      000200      DRDY= BIT7      ;DRIVE READY
1308      000400      DDT= BIT8      ;DRIVE TYPE (0=RK06, 1=RK07)
1309      004000      WRL= BIT11     ;WRITE LOCK
1310      020000      PIP= BIT13     ;POSITIONING IN PROGRESS
1311      040000      DSC= BIT14     ;DRIVE STATUS CHANGE
1312      100000      SVAL= BIT15     ;STATUS VALID
1313
1314      .SBTTL MAINTENANCE REGISTER 1 BIT DEFINITION
1315
1316      000017      MESMSK= 17      ;MESSAGE MASK
1317
1318      000020      PAT= BIT4      ;FORCE EVEN PARITY ON DRIVE BUS MESSAGE LINES
1319      000040      DMD= BIT5      ;DIAGNOSTIC MODE
1320      000100      MSP= BIT6      ;MAINTENANCE SECTOR PULSE
1321      000200      MIND= BIT7      ;MAINTENANCE INDEX
1322      000400      MCLK= BIT8      ;MAINTENANCE CLOCK
1323      001000      MERD= BIT9      ;MAINTENANCE ENCODED READ DATA
1324      002000      MEWD= BIT10     ;MAINTENANCE ENCODED WRITE DATA
1325      004000      PCA= BIT11     ;PRECOMPENSATION ADVANCE
1326      010000      PCD= BIT12     ;PRECOMPENSATION DELAY
1327      020000      ECCW= BIT13     ;ECC WORD IS BEING READ OR WRITTEN
1328      040000      WRTGAT= BIT14   ;WRITE GATE
1329      100000      RDGATE= BIT15   ;READ GATE
1330
1331      .SBTTL DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE A
1332
1333      000040      S.DRA= BIT5      ;DRIVE AVAILIABLE
1334      000100      S.VV= BIT6      ;VOLUME VALID
1335      000200      S.DRY= BIT7      ;DRIVE READY
1336      000400      S.TYPE= BIT8     ;DRIVE TYPE
1337      001000      S.FORM= BIT9     ;DRIVE FORMAT
1338      002000      S.OFF= BIT10     ;OFFSET
1339      004000      S.WRL= BIT11     ;WRITE LOCK
1340      010000      S.SPIN= BIT12    ;SPINDLE ON
1341      020000      S.PIP= BIT13     ;POSITIONING IN PROGRESS

```

```

1342          040GJ0          S.DSC= BIT14          ;DRIVE STATUS CHANGE
1343
1344          .SBTTL  DEFINITION OF DPIVE STATUS BYTE 00 MESSAGE B
1345
1346          000040          S.ICYL= BIT5          ;ILLEGAL CYLINDER ADDRESS
1347          000100          S.ACLO= BIT6          ;AC LOW
1348          000200          S.FLT= BIT7          ;DRIVE FAULT
1349          000400          S.ILF= BIT8          ;ILLEGAL FUNCTION
1350          001000          S.PAR= BIT9          ;DRIVE DETECTED DRIVE BUS PARITY ERROR
1351          002000          S.SKI= BIT10         ;SEEK INCOMPLETE
1352          004000          S.WLE= BIT11         ;WRITE LOCK ERROR
1353          010000          S.SPLS= BIT12        ;SPEED LOSS
1354          010000          S.DCLO= BIT12        ;DC LOW
1355          020000          S.DROT= BIT13        ;DRIVE OFF TRACK
1356          040000          S.UNS= BIT14        ;DRIVE UNSAFE
1357
1358          .SBTTL  DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE A
1359
1360          000020          S.XDOK= BIT4          ;TRANSDUCER OK
1361          000040          S.HDHM= BIT5          ;HEADS HOME
1362          000100          S.BRHM= BIT6          ;BRUSHES HOME
1363          000200          S.DOOR= BIT7          ;DOOR INTERLOCKED
1364          000400          S.CART= BIT8          ;CARTRAGE INTERLOCK
1365          001000          S.SPOK= BIT9          ;SPEED OK
1366          002000          S.FWD= BIT10         ;FORWARD
1367          004000          S.REV= BIT11         ;REVERSE
1368          010000          S.LOAD= BIT12        ;HEADS LOADING
1369          020000          S.RTZ= BIT13        ;RETURN TO ZERO
1370          040000          S.UNLD= BIT14        ;HEADS UNLOADING
1371
1372          .SBTTL  DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE B
1373
1374          000020          S.SECT= BIT4          ;SECTOR ERROR
1375          000040          S.WCLK= BIT5          ;WRITE CLOCK AND NO WRITE GATE
1376          000100          S.WGAT= BIT6          ;WRITE GATE AND NO TRANSISTIONS
1377          000200          S.HDFL= BIT7          ;HEAD FAULT
1378          000400          S.MHD= BIT8          ;MULTIPLE HEAD SELECT
1379          001000          S.XERR= BIT9          ;INDEX ERROR
1380          002000          S.DIB= BIT10         ;DIBIT ERROR
1381          004000          S.PLO= BIT11         ;PLO ERROR
1382          010000          S.NMOV= BIT12        ;SEEK AND NO MOTION
1383          020000          S.LIMD= BIT13        ;LIMIT DETECT ON SEEK
1384          040000          S.BRKE= BIT14        ;SERVO-BRAKE
1385
1386          .SBTTL  COMMON MASKS
1387
1388          000007          M.DRV= 7          ;DRIVE CODE
1389          100000          M.PAR= BIT15         ;PARITY
1390          000003          M.ID= 3          ;BYTE ID
1391          017760          M.CDIF= 17760        ;CYLINDER DIFFERENCE/OFFSET
1392          017760          M.CADD= 17760        ;CYLINDER ADDRESS
1393          077770          M.SER= 77770        ;DRIVE SERIAL NUMBER
1394          000760          M.SECT= 760        ;SECTOR COUNT
1395          007000          M.HEAD= 7000       ;HEAD DECODE
1396
1397          .SBTTL  DEFAULT DRIVE VALUES

```

1398				
1399	000000	D.MNCL= 0		; MINIMUM CYLINDER
1400	000000	D.MNTR= 0		; MINIMUM TRACK
1401	000002	D.MXTR= 2		; MAXIMUM TRACK
1402	000000	D.MNSC= 0		; MINIMUM SECTOR
1403	000025	D.MXSC= 21.		; MAXIMUM SECTOR
1404	000003	D.RATE= 3		; READ/WRITE RATIO (5/3)
1405	000000	D.AWCK= 0		; NO AUTOMATIC WRITE-CHECK AFTER EVERY WRITE
1406	000012	D.CERT= 10.		; CORRECTABLE READ ERROR THRESHOLD
1407	000005	D.UERT= 5		; UNCORRECTABLE READ ERROR THRESHOLD
1408	000005	D.SKET= 5		; SEEK ERROR THRESHOLD
1409	077777	D.CTRH= 77777		; COMMAND COUNT THRESHOLD
1410	177770	D.CTRL= 177770		; (MAXIMUM OF COMMANDS ISSUED TO DRIVE)
1411	077777	D.WTHI= 77777		; WORDS TRANSFERRED THRESHOLD
1412	177770	D.WTLO= 177770		; (MAXIMUM NUMBER OF WORDS READ OR WRITTEN
1413				; ON PACK)
1414	000020	T.NER= 20		; THRESHOLD OF OTHER DRIVE ERRORS
1415				
1416		.SBTTL OFFSET VALUES		
1417				
1418	000060	OFFP12= 060		; OFFSET +1200 MICRO-INCHES
1419	000040	OFFP8= 040		; OFFSET +800 MICRO-INCHES
1420	000020	OFFP4= 020		; OFFSET +400 MICRO-INCHES
1421	000220	OFFM4= 220		; OFFSET -400 MICRO-INCHES
1422	000240	OFFM8= 240		; OFFSET -800 MICRO-INCHES
1423	000260	OFFM12= 260		; OFFSET -1200 MICRO-INCHES
1424				
1425		.SBTTL DEFAULT CONTROLLER THRESHOLD		
1426				
1427	001000	T.DLT= 1000		; THRESHOLD FOR DATA LATE
1428	000012	T.CONT= 10.		; THRESHOLD FOR OTHER CONTROLLER ERRORS

1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484

.SBTTL PARAMETER BLOCK ALLOCATION

```

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

```

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

.SBTTL PARAMETERS PASSED TO THE DRIVER

THE FOLLOWING DEFINITIONS ARE USED TO PASS PARAMETERS TO THE RK06/07 DRIVER

000000
000001
000002
000004
000005
000006
000007
000007
000010
000012
000014

```

P.DRVN= 0 ;DRIVE NUMBER
P.CMND= 1 ;COMMAND
P.CYLN= 2 ;CYLINDER ADDRESS
P.SECT= 4 ;SECTOR
P.TRCK= 5 ;TRACK
P.OFST= 6 ;OFFSET
P.CS1H= 7 ;RKCS1 BITS 8-15
P.BA16= 7 ;BUS ADDRESS (BITS 16 AND 17)
P.BA10= 10 ;BUS ADDRESS (BITS 0-15)
P.WC= 12 ;WORD COUNT (2'S COMPLEMENT)
P.PRST= 14 ;PROGRAM DRIVE STATUS INFORMATION

```

.SBTTL PROGRAM DEVICE STATUS REGISTER DEFINITION

000001
000002
000004
000010
000020
000040

```

DRVUSE= BIT0 ;DRIVE IN USE
DRVPOS= BIT1 ;DRIVE POSITIONING
DRVPOD= BIT2 ;DRIVE POSITIONED FOR DATA TRANSFER
UEXATT= BIT3 ;UNEXPECTED ATTENTION
DRVHRD= BIT4 ;DRIVE HAS HARD ERROR
DRVDSCL= BIT5 ;DRIVE STATUS CHANGE DID NOT CLEAR

```

```

1485          000100          CMDTO= BIT6          ;NO TERMINATION TO COMMAND FOR AT
1486          ;              ;LEAST 1 SECOND
1487          000200          W.WCK= BIT7          ;WRITE FOR WRITE CHECK
1488          000400          NOCHK= BIT8          ;NO CHECK, DO NOT SET INTERRUPT ENABLE
1489          001000          PBSVAL= BIT9          ;PARAMETER STATUS WORDS VALID
1490          ;              ;(SET WHEN ERROR TERMINATION OR
1491          ;              ;READ STATUS COMMAND)
1492          002000          DRPDRV= BIT10         ;DROP DRIVE FROM TEST SEQUENCE
1493          004000          NODSC= BIT11         ;ATTENTION SET BUT DCS AND FAULT RESET
1494          010000          DRVSZD= BIT12         ;DRIVE SEIZED BY OTHER PORT
1495          020000          E.UNLD= BIT13         ;DRIVE UNLOADED DUE TO ERROR
1496          040000          Q.INIT= BIT14         ;PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
1497          100000          DTBAII= BIT15         ;INHIBIT BUS ADDRESS INCREMENT
1498
1499          .SBTTL  PARAMETERS PASSED FROM DRIVER TO PROGRAM
1500
1501          ;              THE FOLLOWING DEFINITIONS ARE USED FOR REGISTER RETURNS
1502          ;              FROM THE DRIVER TO THE CALLING PROGRAM
1503
1504          000016          P.CS1= 16             ;COMMAND AND STATUS REGISTER 1
1505          000020          P.CS2= 20             ;COMMAND AND STATUS REGISTER 2
1506          000022          P.WCR= 22             ;WORD COUNT REGISTER
1507          000024          P.BAR= 24             ;BUS ADDRESS REGISTER
1508          000026          P.DTS= 26             ;DESIRED TRACK SECTOR REGISTER
1509          000030          P.DCYL= 30            ;DESIRED CYLINDER REGISTER
1510          000032          P.ASOF= 32            ;ATTENTION SUMMARY/OFFSET REGISTER
1511          000034          P.ER= 34             ;ERROR REGISTER
1512          000036          P.OS= 36             ;STATUS REGISTER
1513          000040          P.A00= 40             ;MESSAGE A STATUS BYTE 00
1514          000042          P.B00= 42             ;MESSAGE B STATUS BYTE 00
1515          000044          P.A01= 44             ;MESSAGE A STATUS BYTE 01
1516          000046          P.B01= 46             ;MESSAGE B STATUS BYTE 01
1517          000050          P.A10= 50             ;MESSAGE A STATUS BYTE 10
1518          000052          P.B10= 52             ;MESSAGE B STATUS BYTE 10
1519          000054          P.A11= 54             ;MESSAGE A STATUS BYTE 11
1520          000056          P.B11= 56             ;MESSAGE B STATUS BYTE 11
1521          000060          P.EPOS= 60            ;ECC POSITION INFORMATION
1522          000062          P.EPAT= 62            ;ECC PATTERN INFORMATION
1523
1524          .SBTTL  DRIVE SPECIFIC PARAMETERS
1525
1526          000064          P.QLNK= 64             ;QUEUE LINK

```

```

1527 ; THE FOLLOWING DEFINITIONS ARE USED FOR DRIVE PARAMETER ACCESS
1528
1529 000066 P.MNCL= 66 ;MINIMUM CYLINDER
1530 000070 P.MXCL= 70 ;MAXIMUM CYLINDER
1531 000072 P.MNTR= 72 ;MINIMUM TRACK
1532 000073 P.MXTR= 73 ;MAXIMUM TRACK
1533 000074 P.MNSC= 74 ;MINIMUM SECTOR
1534 000075 P.MXSC= 75 ;MAXIMUM SECTOR
1535 000076 P.RATE= 76 ;READ/WRITE RATIO
1536 000077 P.AWCK= 77 ;AUTOMATIC WRITE CHECK
1537 000100 P.CERT= 100 ;CORRECTABLE READ ERROR THRESHOLD
1538 000102 P.UERT= 102 ;UNCORRECTABLE READ ERROR THRESHOLD
1539 000104 P.SKET= 104 ;SEEK ERROR THRESHOLD
1540 000106 P.MXCD= 106 ;MAXIMUM COMMANDS ISSUED TO DRIVE
1541 000112 P.MXWT= 112 ;MAXIMUM NUMBER WORDS TRANSFERRED
1542 000116 P.SMPL= 116 ;SAMPLE COMPARE FLAG
1543 000117 P.ECMP= 117 ;ECC COMPARE FLAG
1544 000120 P.IERC= 120 ;INHIBIT ERROR CORRECTION
1545 000121 P.DPAT= 121 ;DATA PATTERN USED
1546
1547 ; THE FOLLOWING DEFINITIONS ARE USED IN STORING
1548 ; CURRENT RANDOMLY GENERATED COMMAND
1549
1550 000122 P.RDPT= 122 ;CURRENT RANDOMLY GENERATED DATA PATTERN
1551 000123 P.RCMD= 123 ;CURRENT RANDOMLY GENERATED COMMAND
1552 000124 P.RCYL= 124 ;CURRENT RANDOMLY GENERATED CYLINDER
1553 000126 P.RSEC= 126 ;CURRENT RANDOMLY GENERATED SECTOR
1554 000127 P.RTRK= 127 ;CURRENT RANDOMLY GENERATED TRACK
1555 000130 P.RWC= 130 ;CURRENT RANDOMLY GENERATED WORD COUNT
1556 000132 P.RBAL= 132 ;CURRENT RANDOMLY GENERATER BUS ADDRESS
1557 000134
1558
1559 ; THE FOLLOWING DEFINITIONS ARE USED IN STORING
1560 ; LAST ISSUED
1561
1562 000135 P.LCMD= 135 ;LAST COMMAND
1563 000136 P.LCYL= 136 ;LAST CYLINDER
1564 000140 P.LSEC= 140 ;LAST SECTOR
1565 000141 P.LTRK= 141 ;LAST TRACK
1566 000142 P.LWC= 142 ;LAST WORD COUNT
1567 000144 P.LDPT= 144 ;LAST DATA PATTERN
1568 000145 P.BUFF= 145 ;BUFFER ALLOCATED

```

1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614

000146
000147

000150

000152
000156
000164
000172
000174
000176
000200
000202
000204
000206

P.RECT= 146
P.READ= 147

P.DSTT= 150

P.NODR= 152
P.NWRT= 156
P.NRD= 164
P.SRRD= 172
P.SOFT= 174
P.SECC= 176
P.HARD= 200
P.NSKI= 202
P.NOPI= 204
P.NER= 206

THE FOLLOWING IS USED FOR THE RETRY SEQUENCE

:RETRY COUNT
:RETRY INDICATION FOR DATA ERRORS
:0 NOT IN RETRY SEQUENCE
:1-20 REREAD AT CENTER LINE
:21 OFFSET +400 MICRO-INCHES
:22-23 REREAD OFFSET +400 MICRO-INCHES
:24 OFFSET -400 MICRO-INCHES
:25-26 REREAD OFFSET -400 MICRO-INCHES
:27 OFFSET +800 MICRO-INCHES
:30-31 REREAD OFFSET +800 MICRO-INCHES
:32 OFFSET -800 MICRO-INCHES
:33-34 REREAD OFFSET -800 MICRO-INCHES
:35 OFFSET +1200 MICRO-INCHES
:36-37 REREAD OFFSET +1200 MICRO-INCHES
:40 OFFSET -1200 MICRO-INCHES
:41-42 REREAD OFFSET -1200 MICRO-INCHES

:DRIVE STATUS INFORMATION FOR ERROR PROCESSING
:BIT 0 INITIAL DATA ERROR
:BIT 1 DATA ERROR BEING PROCESSED
:BIT 2 NON-DATA ERROR BEING PROCESSED
:BIT 3 SECTOR IN ERROR PRINTED
:BIT 4 READ HEADER ISSUED
:BIT 6 WAITING FOR READ STATUS FOR
: POSITIONING TIME OUT
:BIT 7 FIRST ERROR
:BIT 11 RECALIBRATE HAS BEEN ISSUED
:BIT 12 SEEK HAS BEEN ISSUED
:BIT 13 OFFSET HAS BEEN ISSUED
:BIT 14 BAD SECTOR PROCESSING FOR WRITE CHECK
:BIT 15 ERROR ENQUEUED

THE FOLLOWING DEFINITIONS ARE USED IN STATISTICAL GATHERING

:NUMBER OF ORDERS (2 WORDS)
:NUMBER OF WORDS WRITTEN (3 WORDS)
:NUMBER OF WORDS READ (3 WORDS)
:NUMBER SOFT ERRORS (REREAD CORRECTABLE)
:NUMBER SOFT ERRORS (OFFSET CORRECTABLE)
:NUMBER SOFT ERRORS (ECC CORRECTABLE)
:NUMBER OF HARD DATA ERRORS
:NUMBER OF SEEK INCOMPLETE
:NUMBER OF OPERATION INCOMPLETES
:NUMBER OF ALL OTHER ERRORS


```

1615 ; THE FOLLOWING IS USED FOR THE DRIVE ASSIGNMENT SEQUENCE
1616 ;
1617 000210 P.ASSN= 210 ;ASSIGNMENT COMMAND SEQUENCE
1618 ;0 INITIAL STATE
1619 ;BIT 0 DRIVE SERIAL NUMBER
1620 ;BIT 1 START SPINDLE
1621 ;BIT 2 PACK ACKNOWLEDGE
1622 ;BIT 3 READ PACK SERIAL NUMBER
1623 ;BIT 4 WRITE PACK COMMAND ISSUED
1624 ;BIT 5 END OF PACK WRITE
1625 ;BIT 6 RECAL
1626 ;BIT 7 RECAL AND RETRY READ PACK
1627 ; SERIAL NUMBER
1628 ;
1629 000211 P.SEEK= 211 ;SEEK TO PREVIOUS CYLINDER FLAG
1630 ;
1631 ; THE FOLLOWING LOCATION IS USED TO INDICATE ERRORS
1632 ;
1633 000212 P.ERR= 212 ;BIT 2 DATA COMPARISON ERROR
1634 ;BIT 3 DRIVE ERROR NOT REPORTED
1635 ;BIT 4 DRIVE SEIZED TIME OUT
1636 ;BIT 5 DRIVE POSITIONING TIME OUT
1637 ;BIT 6 ERROR WHILE ENQUEUED
1638 ;BIT 7 ATTENTION WHEN DRIVE NOT IN USE
1639 ;BIT 8 TIME OUT WHILE WAITING FOR HEADS
1640 ; TO LOAD AFTER ERROR
1641 ;BIT 10 DRIVE NOT READY AFTER START SPINDLE
1642 ;BIT 11 PACK ACKNOWLEDGE DID NOT SET VOLUME
1643 ; VALID
1644 ;
1645 000214 P.ERHD= 214 ;HEADER ADDRESS OF FIRST DATA MISCOMPARE (3 BYTES)
1646 000217 P.ERAD= 217 ;OFFSET COUNT FROM HEADER ADDRESS
1647 000220 P.CMLG= 220 ;COMPARISON LENGTH
1648 000222 P.BFCP= 222 ;BUFFER COMPARE LENGTH
1649 ;
1650 ; THE FOLLOWING LOCATIONS ARE USED FOR DRIVE SERIAL NUMBER,
1651 ; PACK SERIAL NUMBER, AND TRACK EXCLUSIONS
1652 ;
1653 000224 P.SERL= 224 ;DRIVE SERIAL NUMBER
1654 000226 P.PKSR= 226 ;CARTRIDGE SERIAL NUMBER
1655 000232 P.EXAR= 232 ;EXCLUDED AREA INFORMATION

```

```

1656
1657
1658      000000
1659
1660
1661
1662      000174
1663 000174 000000
1664 000176 000000
1665
1666 000200 000137 006670
1667 000204 000137 006652
1668
1669 000214 000137 006642
1670 000220 000137 065062
1671 000224 000700
1672 000226 004600
1673
1674 000240 000137 006662
1675      001000
1676
1677
1678
1679
1680
1681      001000
1682      000024
1683 000024 000200
1684      000044
1685 000044 001000
1686      001000
1687
1688
1689
1690
1691 001000
1692 001000 000000
1693 001002 001170
1694 001004 003000
1695 001006 003000
1696 001010 003000
1697 001012 000052

```

```

.SBTTL TRAP CATCHER
      =0
; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      =174
DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
      =214
JMP @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
JMP RESTART ;; BRANCH TO RESTART
      =240
JMP PARSRT  ;; GET SYSTEM PARAMETERS AND START TESTING
JMP ..DUMP  ;; *** MEMORY DUMP (DEBUG ONLY)
..LOW: 700    ;; START OF DUMP
..HIGH: PATTAB-2 ;; LAST ADDRESS OF DUMP
      =240
JMP START1
      =1000
.SBTTL APT PARAMETER BLOCK
; *****
; SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
; *****
      =24
.SX=      ;; SAVE CURRENT LOCATION
      =24
      200  ;; SET POWER FAIL TO POINT TO START OF PROGRAM
      =44
      $APTHDR ;; FOR APT START UP
      =44
      $APTHDR ;; POINT TO APT INDIRECT ADDRESS PNTR.
      =.SX
      $APTHDR ;; POINT TO APT HEADER BLOCK
      =.SX
      $APTHDR ;; RESET LOCATION COUNTER
; *****
; SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-POP11 DIAGNOSTIC
; INTERFACE SPEC.
$APTHDR:
$HIBTS: .WORD 0          ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL      ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 3000      ;; RUN TIME OF LONGEST TEST
$PASTM: .WORD 3000      ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 3000      ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD SETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

```

COMMON TAGS

.SBTTL COMMON TAGS

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

. =1100

\$CMTAG:

;; START OF COMMON TAGS

.WORD 0

;; CONTAINS THE TEST NUMBER

.BYTE 0

;; CONTAINS ERROR FLAG

.BYTE 0

;; CONTAINS SUBTEST ITERATION COUNT

.WORD 0

;; CONTAINS SCOPE LOOP ADDRESS

.WORD 0

;; CONTAINS SCOPE RETURN FOR ERRORS

.WORD 0

;; CONTAINS TOTAL ERRORS DETECTED

.WORD 0

;; CONTAINS ITEM CONTROL BYTE

.BYTE 0

;; CONTAINS MAX. ERRORS PER TEST

.BYTE 1

;; CONTAINS PC OF LAST ERROR INSTRUCTION

.WORD 0

;; CONTAINS ADDRESS OF 'GOOD' DATA

.WORD 0

;; CONTAINS ADDRESS OF 'BAD' DATA

.WORD 0

;; CONTAINS 'GOOD' DATA

.WORD 0

;; CONTAINS 'BAD' DATA

.WORD 0

;; RESERVED--NOT TO BE USED

.WORD 0

;; AUTOMATIC MODE INDICATOR

.BYTE 0

;; INTERRUPT MODE INDICATOR

.BYTE 0

.WORD 0

;; ADDRESS OF SWITCH REGISTER

.WORD DSWR

;; ADDRESS OF DISPLAY REGISTER

.WORD DDISP

.WORD 177560

;; TTY KBD STATUS

.WORD 177562

;; TTY KBD BUFFER

.WORD 177564

;; TTY PRINTER STATUS REG. ADDRESS

.WORD 177566

;; TTY PRINTER BUFFER REG. ADDRESS

.BYTE 0

;; CONTAINS NULL CHARACTER FOR FILLS

.BYTE 2

;; CONTAINS # OF FILLER CHARACTERS REQUIRED

.BYTE 12

;; INSERT FILL CHARS. AFTER A "LINE FEED"

.BYTE 0

;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)

.ASCIZ <207><377><377>

;; CODE FOR BELL

.ASCIZ /?/

;; QUESTION MARK

.ASCIZ <15>

;; CARRIAGE RETURN

.ASCIZ <12>

;; LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN

.WORD AMSGTY

;; APT MAILBOX

.WORD AFATAL

;; MESSAGE TYPE CODE

.WORD ATESTN

;; FATAL ERROR NUMBER

.WORD APASS

;; TEST NUMBER

.WORD ADEVCT

;; PASS COUNT

.WORD AUNIT

;; DEVICE COUNT

.WORD AMSGAD

;; I/O UNIT NUMBER

.WORD AMSGLG

;; MESSAGE ADDRESS

.ETABLE:

;; MESSAGE LENGTH

;; APT ENVIRONMENT TABLE

1698
1699
1700
1701
1702
1703
1704 001100
1705 001100 000000
1706 001102 000
1707 001103 000
1708 001104 000000
1709 001106 000000
1710 001110 000000
1711 001112 000000
1712 001114 000
1713 001115 001
1714 001116 000000
1715 001120 000000
1716 001122 000000
1717 001124 000000
1718 001126 000000
1719 001130 000000
1720 001132 000000
1721 001134 000
1722 001135 000
1723 001136 000000
1724 001140 177570
1725 001142 177570
1726 001144 177560
1727 001146 177562
1728 001150 177564
1729 001152 177566
1730 001154 000
1731 001155 002
1732 001156 012
1733 001157 000
1734 001160 177607 000377
1735 001164 077
1736 001165 015
1737 001166 000012
1738
1739
1740
1741
1742
1743
1744 001170
1745 001170 000000
1746 001172 000000
1747 001174 000000
1748 001176 000000
1749 001200 000000
1750 001202 000000
1751 001204 000000
1752 001206 000000
1753 001210

1754	001210	000	\$ENV:	.BYTE	AENV	:: ENVIRONMENT BYTE
1755	001211	000	\$ENVM:	.BYTE	AENVM	:: ENVIRONMENT MODE BITS
1756	001212	000000	\$SWREG:	.WORD	ASWREG	:: APT SWITCH REGISTER
1757	001214	000000	\$USWR:	.WORD	AUSWR	:: USER SWITCHES
1758	001216	000000	\$CPUOP:	.WORD	ACPUOP	:: CPU TYPE, OPTIONS
1759			*			BITS 15-11=CPU TYPE
1760			*			11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
1761			*			11/70=06, PDQ=07, Q=10
1762			*			BIT 10=REAL TIME CLOCK
1763			*			BIT 9=FLOATING POINT PROCESSOR
1764			*			BIT 8=MEMORY MANAGEMENT
1765	001220	000	\$MAMS1:	.BYTE	AMAMS1	:: HIGH ADDRESS M.S. BYTE
1766	001221	000	\$MTYP1:	.BYTE	AMTYP1	:: MEM. TYPE, BLK#1
1767			*			MEM. TYPE BYTE -- (HIGH BYTE)
1768			*			900 MSEC CORE=001
1769			*			300 MSEC BIPOLAR=002
1770			*			500 MSEC MOS=003
1771	001222	000000	\$MADR1:	.WORD	AMADR1	:: HIGH ADDRESS, BLK#1
1772			*			MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
1773	001224	000	\$MAMS2:	.BYTE	AMAMS2	:: HIGH ADDRESS M.S. BYTE
1774	001225	000	\$MTYP2:	.BYTE	AMTYP2	:: MEM. TYPE, BLK#2
1775	001226	000000	\$MADR2:	.WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
1776	001230	000	\$MAMS3:	.BYTE	AMAMS3	:: HIGH ADDRESS M.S. BYTE
1777	001231	000	\$MTYP3:	.BYTE	AMTYP3	:: MEM. TYPE, BLK#3
1778	001232	000000	\$MADR3:	.WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
1779	001234	000	\$MAMS4:	.BYTE	AMAMS4	:: HIGH ADDRESS M.S. BYTE
1780	001235	000	\$MTYP4:	.BYTE	AMTYP4	:: MEM. TYPE, BLK#4
1781	001236	000000	\$MADR4:	.WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
1782	001240	120210	\$VECT1:	.WORD	AVECT1	:: INTERRUPT VECTOR#1 BUS PRIORITY#1
1783	001242	000000	\$VECT2:	.WORD	AVECT2	:: INTERRUPT VECTOR#2 BUS PRIORITY#2
1784	001244	177440	\$BASE:	.WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
1785	001246	000000	\$DEVN:	.WORD	ADEVN	:: DEVICE MAP
1786	001250	000000	\$CDW1:	.WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
1787	001252	000000	\$CDW2:	.WORD	ACDW2	:: CONTROLLER DESCRIPTION WORD#2
1788	001254	000000	\$DDW0:	.WORD	ADDW0	:: DEVICE DESCRIPTOR WORD#0
1789	001256	000000	\$DDW1:	.WORD	ADDW1	:: DEVICE DESCRIPTOR WORD#1
1790	001260	000000	\$DDW2:	.WORD	ADDW2	:: DEVICE DESCRIPTOR WORD#2
1791	001262	000000	\$DDW3:	.WORD	ADDW3	:: DEVICE DESCRIPTOR WORD#3
1792	001264	000000	\$DDW4:	.WORD	ADDW4	:: DEVICE DESCRIPTOR WORD#4
1793	001266	000000	\$DDW5:	.WORD	ADDW5	:: DEVICE DESCRIPTOR WORD#5
1794	001270	030000	\$DDW6:	.WORD	ADDW6	:: DEVICE DESCRIPTOR WORD#6
1795	001272	000000	\$DDW7:	.WORD	ADDW7	:: DEVICE DESCRIPTOR WORD#7
1796	001274	000000	\$DDW8:	.WORD	ADDW8	:: DEVICE DESCRIPTOR WORD#8
1797	001276	000000	\$DDW9:	.WORD	ADDW9	:: DEVICE DESCRIPTOR WORD#9
1798	001300	000000	\$DDW10:	.WORD	ADDW10	:: DEVICE DESCRIPTOR WORD#10
1799	001302	000000	\$DDW11:	.WORD	ADDW11	:: DEVICE DESCRIPTOR WORD#11
1800	001304	000000	\$DDW12:	.WORD	ADDW12	:: DEVICE DESCRIPTOR WORD#12
1801	001306	000000	\$DDW13:	.WORD	ADDW13	:: DEVICE DESCRIPTOR WORD#13
1802	001310	000000	\$DDW14:	.WORD	ADDW14	:: DEVICE DESCRIPTOR WORD#14
1803	001312	000000	\$DDW15:	.WORD	ADDW15	:: DEVICE DESCRIPTOR WORD#15
1804						
1805						
1806	001314		\$ETEND:			
1807						

```

1808      .SBTTL  QUEUE HEAD AND TAIL ALLOCATIONS
1809
1810 001314 000000 000000  AVAILQ: .WORD 0,0      ;HEAD AND TAIL OF AVAILABLE QUEUE
1811 001320 000000 000000  BWAITQ: .WORD 0,0     ;HEAD AND TAIL OF BUFFER WAIT QUEUE
1812 001324 000000 000000  CINITQ: .WORD 0,0     ;HEAD AND TAIL OF COMMAND INITIATION QUEUE
1813 001330 000000 000000  ERRPRQ: .WORD 0,0     ;HEAD AND TAIL OF ERROR PROCESSING QUEUE
1814
1815      .SBTTL  SYSTEM PARAMETERS
1816
1817 001334 172540  $LKCSR: .WORD 172540   ;ADDRESS OF KW11-P STATUS REGISTER
1818 001336 172542  $LKCSB: .WORD 172542   ;ADDRESS OF KW11-P COUNTER BUFFER
1819 001340 000104  $LPVEC: .WORD 104      ;KW11-P VECTOR ADDRESS
1820 001342 177546  $LKS:   .WORD 177546   ;ADDRESS OF KW11-L STATUS REGISTER
1821 001344 000100  $LLVEC: .WORD 100      ;KW11-L VECTOR ADDRESS
1822 001346 177514  $LSCS:  .WORD 177514   ;ADDRESS OF PRINTER STATUS WORD
1823 001350 177516  $LSDB:  .WORD 177516   ;ADDRESS OF PRINTER DATA BUFFER
1824 001352 074     HZ:      .BYTE 60.     ;74 (60 DECIMAL) IF SYSTEM IS 60 HZ.
1825                                     ;62 (50 DECIMAL) IF SYSTEM IS 50 HZ.
1826 001353 000     CLKFRQ: .BYTE 0      ;COUNT FOR SECOND
1827 001354 013400  MAXBUF: .WORD 256.*23. ;MAXIMUM BUFFER FOR READ OR WRITE
1828 001356 177777  PASCNT: .WORD 177777   ;PASS COUNT
1829 001360 000     PERINV: .BYTE 0      ;INTERVAL BETWEEN PERFORMANCE SUMMARIES
1830 001361 000     FLAG:   .BYTE 0      ;RESTART AND PARAMETER FLAG
1831 001362 000003  SOFCMP: .WORD 3        ;NUMBER OF SOFTWARE COMPARES
1832 001364 000     CPUID:  .BYTE 0      ;CPU DESIGNATION FOR PACK WRITING WITH DUAL ACCESS
1833 001365 000     OVLYLD: .BYTE 0      ;OVERLAY LOADER
1834 001366 000000  HOUR:   .WORD 0        ;TIMER HOUR
1835 001370 000000  MINUTE: .WORD 0        ;TIMER MINUTE
1836 001372 000000  SECOND: .WORD 0        ;TIMER SECOND
1837 001374 000     CLKFLG: .BYTE 0      ;KW11-P OR KW11-L PRESENT
1838 001375 000     PERIOD: .BYTE 0      ;PERIOD SINCE LAST STATISTIC PRINT OUT
1839 001376 000000  TIMHR:  .WORD 0        ;HOUR COUNT FOR PRINT OUT
1840 001400 000000  TIMMIN: .WORD 0        ;MINUTE COUNT FOR PRINT OUT
1841 001402 000000  TIMSEC: .WORD 0        ;SECOND COUNT FOR PRINT OUT
1842 001404 000000  SAVSWR: .WORD 0        ;SAVED SWITCH REGISTER FROM POWER FAIL
1843 001406 000000  RELFLG: .WORD 0        ;FLAG FOR RELEASE DURING WRITE PACK
1844
1845      .SBTTL  TEMPORARY CONTROLLER REGISTER STORAGE
1846
1847 001410 000000  T.CS1:  .WORD 0        ;TEMPORARY STORAGE FOR COMMAND AND STATUS
1848                                     ;REGISTER 1
1849 001412 000000  T.CS2:  .WORD 0        ;TEMPORARY STORAGE FOR COMMAND AND STATUS
1850                                     ;REGISTER 2
1851 001414 000000  T.WCR:  .WORD 0        ;TEMPORARY STORAGE FOR WORD COUNT REGISTER
1852 001416 000000  T.BA:   .WORD 0        ;TEMPORARY STORAGE FOR BUS ADDRESS REGISTER
1853 001420 000000  T.DA:   .WORD 0        ;TEMPORARY STORAGE FOR DISK TRACK AND SECTOR
1854 001422 000000  T.DC:   .WORD 0        ;TEMPORARY STORAGE FOR DRIVE CYLINDER
1855 001424 000000  T.ASOF: .WORD 0        ;TEMPORARY STORAGE FOR ATTENTION SUMMARY
1856                                     ;AND OFFSET
1857 001426 000000  T.ER:   .WORD 0        ;TEMPORARY STORAGE FOR ERROR REGISTER
1858 001430 000000  T.DS:   .WORD 0        ;TEMPORARY STORAGE FOR DRIVE STATUS REGISTER
1859 001432 000000  T.MR1:  .WORD 0        ;TEMPORARY STORAGE FOR MAINTENANCE REGISTER 1
1860 001434 000000  T.MR2:  .WORD 0        ;TEMPORARY STORAGE FOR MAINTENANCE REGISTER 2
1861 001436 000000  T.MR3:  .WORD 0        ;TEMPORARY STORAGE FOR MAINTENANCE REGISTER 3
1862 001440 000000  T.POS:  .WORD 0        ;TEMPORARY STORAGE FOR ECC POSITION
1863 001442 000000  T.PAT:  .WORD 0        ;TEMPORARY STORAGE FOR ECC PATTERN

```

```

1864 001444 000000      T.DB: .WORD 0          ; TEMPORARY STORAGE FOR DATA BUFFER REGISTER
1865
1866                      .SBTTL DRIVER PARAMERTERS
1867
1868 001446 177440      RKBAS: .WORD 177440    ; ADDRESS OF RK611 UNIBUS ADDRESS BLOCK
1869 001450 000210      RKVEC: .WORD 210      ; ADDRESS OF R611 VECTOR
1870 001452 000240      RKPRI: .WORD PR5      ; RK611 INTERRUPT PRIORITY
1871 001454 024222      A.NORM: NORMAL       ; ADDRESS OF NORMAL RETURN FROM DRIVER
1872 001456 025312      A.ABNL: ABNORM       ; ADDRESS OF ABNORMAL RETURN FROM DRIVER
1873 001460 026346      A.CONT: CONTRL      ; ADDRESS OF CONTROLLER ERROR RETURN
1874 001462 000000      E.CONT: .WORD 0      ; CONTROLLER ERROR STATUS
1875
1876                      ; THIS LOCATION IS CLEARED WHEN EVERY COMMAND
1877                      ; IS INITIATED. IF A CONTROLLER ERROR
1878                      ; OCCURS THE FOLLOWING BIT ASSIGNMENT IS
1879                      ; USED:
1880
1881                      E.CCLR= BIT0    ; CLEAR CONTROLLER DID NOT CLEAR ERROR
1882                      E.NOAT= BIT1    ; NO ATTENTION IN ATTENTION SUMMARY REG
1883                      E.UATT= BIT2    ; UNSOLICATED ATTENTION (SEQUENTIAL ONLY)
1884                      E.UDAT= BIT3    ; UNEXPECTED DATA TYPE ERROR
1885                      E.CLAT= BIT4    ; ATTENTION DID NOT RESET WITH CLEAR
1886                      E.SCLR= BITS   ; SUBSYSTEM CLEAR DID NOT CLEAR DRIVE
1887                      ; ATTENTION
1888                      E.ILLD= BIT6    ; ILLEGAL DRIVER COMMAND
1889                      E.DLT= BIT8     ; DATA LATE WHEN UNLOADING HEADER
1890                      E.CERR= BIT9    ; CONTROLLER ERROR DURING DRIVER SERVICING
1891                      E.DPAR= BIT10   ; DRIVE DETECTED PARITY ERROR
1892                      E.CMTO= BIT14   ; CONTROLLER COMMAND TIME OUT (QUEUED ONLY)
1893                      E.MDS= BIT15   ; MULTIPLE DRIVE SELECT
1894 001464 000000      O.WAIT: .WORD 0      ; PARAMETER BLOCK OF THE DRIVE
1895                      ; WAITING FOR COMMAND COMPLETION
1896 001466 000400      W.MTIM: .WORD 400    ; LOOP COUNTER FOR MILLISECOND SCAN OF DRIVE
1897 001470 000400      W.MILI: .WORD 400    ; 16 MILLISECOND TIME FOR PROGRAM
1898
1899                      ; CPU VALUE
1900                      ; ---
1901                      ;
1902                      ; 11/05 100
1903                      ; 11/10
1904                      ; 11/20
1905                      ; 11/34
1906                      ; 11/40
1907                      ; 11/45 400
1908                      ; 11/50
1909                      ; 11/70
1910 001472 000300      W.SEC: .WORD 300     ; SECOND COUNT COUNT FOR ALL COMMANDS
1911                      ; EXCEPT START SPINDLE
1912 001474 003000      W.BSEC: .WORD 3000   ; 8 SECOND FOR DRIVE CYCLE DOWN
1913 001476 030000      W.MIN: .WORD 30000   ; MINUTE TIME FOR START SPINDLE
1914 001500 000000      OPTFLG: .WORD 0      ; FLAG = 1 WHILE IN CMND OPTIMIZER
1915 001502 000000      HDR.AD: .WORD 0      ; ADDRESS USED FOR READ ALL HEADERS
1916 001504 000000      HDR.CT: .WORD 0      ; NUMBER OF HEADERS LEFT TO READ FOR READ
1917                      ; ALL HEADERS
1918 001506 000          I.ISRL: .BYTE 0    ; INTERRUPT OR RELEASED COMMAND ISSUED
1919 001507 002          H.HEAD: .BYTE 2,4,10 ; HEAD DECODES

```

1920 001512 000
1921 001513 377
1922
1923
1924
1925 001514 001
1926 001515 002
1927 001516 004
1928 001517 010
1929 001520 020
1930 001521 040
1931 001522 100
1932 001523 200
1933
1934
1935
1936 001524 001736
1937 001526 002214
1938 001530 002472
1939 001532 002750
1940 001534 003226
1941 001536 003504
1942 001540 003762
1943 001542 004240
1944
1945
1946
1947 001544 000000
1948 001546 000000
1949 001550 000000
1950 001552 000000
1951 001554 000000
1952 001556 000000
1953 001560 000000
1954 001562 000000

W.TIME: .BYTE 0 ;DRIVES BEING WATCH-DOG TIMED
O.OVER: .BYTE -1 ;OVERLAPPED OPERATIONS

.SBTTL TABLE OF INTERRUPT MASKS

INTMSK: .BYTE 1 ; INTERRUPT FOR DRIVE 0
.BYTE 2 ; INTERRUPT FOR DRIVE 1
.BYTE 4 ; INTERRUPT FOR DRIVE 2
.BYTE 10 ; INTERRUPT FOR DRIVE 3
.BYTE 20 ; INTERRUPT FOR DRIVE 4
.BYTE 40 ; INTERRUPT FOR DRIVE 5
.BYTE 100 ; INTERRUPT FOR DRIVE 6
.BYTE 200 ; INTERRUPT FOR DRIVE 7

.SBTTL PARAMETER BLOCK TABLE

PBLKT: PARM0 ; ADDRESS OF DRIVE 0'S PARAMETER BLOCK
PARM1 ; ADDRESS OF DRIVE 1'S PARAMETER BLOCK
PARM2 ; ADDRESS OF DRIVE 2'S PARAMETER BLOCK
PARM3 ; ADDRESS OF DRIVE 3'S PARAMETER BLOCK
PARM4 ; ADDRESS OF DRIVE 4'S PARAMETER BLOCK
PARM5 ; ADDRESS OF DRIVE 5'S PARAMETER BLOCK
PARM6 ; ADDRESS OF DRIVE 6'S PARAMETER BLOCK
PARM7 ; ADDRESS OF DRIVE 7'S PARAMETER BLOCK

.SBTTL WATCH-DOG TIMER COUNTS

W.DRV: .WORD 0 ; WATCH-DOG FOR DRIVE 0
.WORD 0 ; WATCH-DOG FOR DRIVE 1
.WORD 0 ; WATCH-DOG FOR DRIVE 2
.WORD 0 ; WATCH-DOG FOR DRIVE 3
.WORD 0 ; WATCH-DOG FOR DRIVE 4
.WORD 0 ; WATCH-DOG FOR DRIVE 5
.WORD 0 ; WATCH-DOG FOR DRIVE 6
.WORD 0 ; WATCH-DOG FOR DRIVE 7

.SBTTL PERFORMANCE EXERCISER STATUS INFORMATION

1955									
1956									
1957	001564	000000							
1958									
1959									
1960									
1961									
1962									
1963									
1964	001566	000000							
1965	001570	000000							
1966	001572	000000	000000						
1967	001576	000000	000000						
1968	001602	000000	000000						
1969	001606	000000	000000						
1970	001612	000000	000000						
1971	001616	000000	000000						
1972	001622	000000	000000						
1973	001626	000000							
1974	001630	000000							
1975	001632	000							
1976	001633	000							
1977	001634	000000							
1978	001636	000000							
1979	001640	000							
1980	001641	000							
1981	001642	000000							
1982	001644	000							
1983	001645	000							
1984	001646	000000							
1985	001650	000000							
1986	001652	000000							
1987	001654	000							
1988	001655	000							
1989	001656	000000							
1990	001660	000000							
1991	001662	000000							
1992	001664	000000							
1993	001666	000000							
1994									
1995									
1996									
1997	001670	000001							
1998									
1999	001672	000000	000000						
2000	001676	000000	000000						
2001	001702	000000	000000						
2002	001706	000000	000000						
2003	001712	000000	000000						
2004	001716	000000	000000						
2005	001722	000000	000000						
2006	001726	000000	000000						
2007	001732	000000	000000						

```

ERCONT: .WORD 0 ; CONTROLLER ERROR (RECOVERABLE)
; BIT 0 CONTROLLER ERROR NOT FLAGGED
; BIT 1 WORD COUNT NOT EQUAL 0
; BIT 2 BUS ADDRESS INCORRECT
; BIT 3 CYLINDER TRACK SECTOR INCORRECT
; BIT 15 CONTROLLER ERROR DETECTED

ERRPRO: .WORD 0 ; ADDRESS OF PARAMETER BLOCK PROCESSING ERROR
TMPCNT: .WORD 0 ; TRANSFER COUNT CALCULATIONS
TMPTRK: .WORD 0,0 ; RANDOM TRACK CALCULATIONS
TMPSEC: .WORD 0,0 ; RANDOM SECTOR CALCULATIONS
PARCYL: .WORD 0,0 ; PARAMETER CYLINDER MODIFICATION
PARTRK: .WORD 0,0 ; PARAMETER TRACK MODIFICATION
PARSEC: .WORD 0,0 ; PARAMETER SECTOR MODIFICATION
EXAREA: .WORD 0,0 ; EXCLUDED AREA CALCULATIONS
ECCPAT: .WORD 0,0 ; ECC PATTERN FOR ECC CORRECTION
DLTCNT: .WORD 0 ; DATA LATE COUNT
CNTCNT: .WORD 0 ; OTHER CONTROLLER ERRORS COUNT
DRV CNT: .BYTE 0 ; DRIVE COUNT FOR OPERATOR COMMAND
STATIS: .BYTE 0 ; REPORT INTERVAL STATISTICS
RTYBA: .WORD 0 ; RETRY BUS ADDRESS
RTYWC: .WORD 0 ; RETRY WORD COUNT
RTYSEC: .BYTE 0 ; SECTOR ADDRESS RETRY
RTYTRK: .BYTE 0 ; TRACK ADDRESS RETRY
RTYCYL: .WORD 0 ; CYLINDER ADDRESS RETRY
RTYSCN: .BYTE 0 ; NUMBER OF SECTORS TO BE TRANSFERRED IN RETRY
RTYCMD: .BYTE 0 ; COMMAND BEING RETRIED
HEAD1: .WORD 0 ; TEMPORARY STORAGE FOR FIRST WORD OF HEADER
HEAD2: .WORD 0 ; TEMPORARY STORAGE FOR SECOND WORD OF HEADER
HEAD3: .WORD 0 ; TEMPORARY STORAGE FOR THIRD WORD OF HEADER
BSSECT: .BYTE 0 ; SECTOR IN BAD SECTOR SERVICING
BSTRCK: .BYTE 0 ; TRACK IN BAD SECTOR SERVICING
BSCYLN: .WORD 0 ; CYLINDER IN BAD SECTOR SERVICING
BSWC: .WORD 0 ; WORD COUNT IN BAD SECTOR SERVICING
BSBA: .WORD 0 ; BUS ADDRESS IN BAD SECTOR SERVICING
BSWORD: .WORD 0 ; WORDS TRANSFERRED IN BAD SECTOR SERVICING
POSCMD: .WORD 0 ; POSITIONING TIME OUT COMMAND

```

.SBTTL FREE SPACE BLOCK

1997	001670	000001							
1998									
1999	001672	000000	000000						
2000	001676	000000	000000						
2001	001702	000000	000000						
2002	001706	000000	000000						
2003	001712	000000	000000						
2004	001716	000000	000000						
2005	001722	000000	000000						
2006	001726	000000	000000						
2007	001732	000000	000000						

```

FBLKC: .WORD 1 ; FREE BLOCK COUNT
FBLKT: .WORD 0,0 ; FREE BLOCK 0
; FREE BLOCK 1
; FREE BLOCK 2
; FREE BLOCK 3
; FREE BLOCK 4
; FREE BLOCK 5
; FREE BLOCK 6
; FREE BLOCK 7
; FREE BLOCK 8

```



```

2008
2009          .SBTTL  PARAMETER BLOCK FOR DRIVE 0
2010
2011 001736      000          PARM0: .BYTE 0          :DRIVE 0
2012 001737      000          .BYTE 00          :COMMAND
2013 001740      000000      .WORD 00          :CYLINDER ADDRESS
2014 001742      000          .BYTE 00          :SECTOR ADDRESS
2015 001743      000          .BYTE 00          :TRACK ADDRESS
2016 001744      000          .BYTE 00          :OFFSET VALUE
2017 001745      000          .BYTE 00          :BUS ADDRESS (BITS 16 AND 17)
2018 001746      000000      .WORD 00          :BUS ADDRESS (BITS 0 - 15)
2019 001750      000000      .WORD 00          :WORD COUNT (2'S COMPLEMENT)
2020 001752      000000      .WORD 00          :PROGRAM DRIVE STATUS INFORMATION
2021 001754      000000      .WORD 00          :COMMAND AND STATUS REGISTER 1
2022 001756      000000      .WORD 00          :COMMAND AND STATUS REGISTER 2
2023 001760      000000      .WORD 00          :WORD COUNT REGISTER
2024 001762      000000      .WORD 00          :BUS ADDRESS REGISTER
2025 001764      000000      .WORD 00          :DESIRED TRACK AND SECTOR REGISTER
2026 001766      000000      .WORD 00          :DESIRED CYLINDER REGISTER
2027 001770      000000      .WORD 00          :ATTENTION SUMMARY/OFFSET REGISTER
2028 001772      000000      .WORD 00          :ERROR REGISTER
2029 001774      000000      .WORD 00          :STATUS REGISTER
2030 001776      000000      .WORD 00          :MESSAGE LINE A STATUS BYTE 00
2031 002000      000000      .WORD 00          :MESSAGE LINE B STATUS BYTE 00
2032 002002      000000      .WORD 00          :MESSAGE LINE A STATUS BYTE 01
2033 002004      000000      .WORD 00          :MESSAGE LINE B STATUS BYTE 01
2034 002006      000000      .WORD 00          :MESSAGE LINE A STATUS BYTE 10
2035 002010      000000      .WORD 00          :MESSAGE LINE B STATUS BYTE 10
2036 002012      000000      .WORD 00          :MESSAGE LINE A STATUS BYTE 11
2037 002014      000000      .WORD 00          :MESSAGE LINE B STATUS BYTE 11
2038 002016      000000      .WORD 00          :ECC POSITION INFORMATION
2039 002020      000000      .WORD 00          :ECC PATTERN INFORMATION
2040 002022      000000      .WORD 00          :QUEUE LINK
2041 002024      000000      .WORD 00          :MINIMUM CYLINDER
2042 002026      000632      .WORD 410.        :MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
2043 002030      000          .BYTE 00          :MINIMUM TRACK
2044 002031      002          .BYTE 02          :MAXIMUM TRACK
2045 002032      000          .BYTE 00          :MINIMUM SECTOR
2046 002033      025          .BYTE 21.        :MAXIMUM SECTOR
2047 002034      003          .BYTE 03          :READ/WRITE RATIO
2048 002035      000          .BYTE 00          :AUTOMATIC WRITE CHECK
2049 002036      000012      .WORD 10.        :CORRECTABLE READ ERROR THRESHOLD
2050 002040      000005      .WORD 5          :UNCORRECTABLE READ ERROR THRESHOLD
2051 002042      000005      .WORD 5          :SEEK ERROR THRESHOLD
2052 002044      177770      177777      .WORD 177770,177777 :MAXIMUM NUMBER OF COMMANDS TO DRIVE
2053 002050      177770      177777      .WORD 177770,177777 :MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
2054 002054      000          .BYTE 00          :SAMPLE COMPARE FLAG
2055 002055      000          .BYTE 00          :ECC COMPARE FLAG
2056 002056      000          .BYTE 00          :INHIBIT ERROR CORRECTION
2057 002057      000          .BYTE 00          :DATA PATTERN USED
2058 002060      000          .BYTE 00          :CURRENT GENERATED DATA PATTERN
2059 002061      000          .BYTE 00          :CURRENT RANDOMLY GENERATED COMMAND
2060 002062      000000      .WORD 00          :CURRENT RANDOMLY GENERATED CYLINDER
2061 002064      000          .BYTE 00          :CURRENT RANDOMLY GENERATED SECTOR
2062 002065      000          .BYTE 00          :CURRENT RANDOMLY GENERATED TRACK
2063 002066      000000      .WORD 0          :CURRENT RANDOMLY GENERATED WORD COUNT

```

2064	002070	000000			.WORD	0	; CURRENT RANDOMLY GENERATED BUS ADDRESS
2065	002072	000			.BYTE	0,0	
2066	002073	000			.BYTE	0,0	; LAST COMMAND
2067	002074	000000			.WORD	0,0	; LAST CYLINDER
2068	002076	000			.BYTE	0,0	; LAST SECTOR
2069	002077	000			.BYTE	0,0	; LAST TRACK
2070	002100	000000			.WORD	0,0	; LAST WORD COUNT
2071	002102	000			.BYTE	0,0	; LAST DATA PATTERN USED
2072	002103	000			.BYTE	0,0	; BUFFER ALLOCATED
2073	002104	000			.BYTE	0,0	; RETRY COUNT
2074	002105	000			.BYTE	0,0	; REREAD STATUS
2075	002106	000000			.WORD	0,0	; DRIVE STATUS FLAGS
2076	002110	000000	000000		.WORD	0,0	; NUMBER OF ORDERS
2077	002114	000000	000000	000000	.WORD	0,0,0	; NUMBER OF WORDS WRITTEN
2078	002122	000000	000000	000000	.WORD	0,0,0	; NUMBER OF WORDS READ
2079	002130	000000			.WORD	0,0	; NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2080	002132	000000			.WORD	0,0	; NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2081	002134	000000			.WORD	0,0	; NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2082	002136	000000			.WORD	0,0	; NUMBER OF HARD DATA ERRORS
2083	002140	000000			.WORD	0,0	; NUMBER OF SEEK INCOMPLETES
2084	002142	000000			.WORD	0,0	; NUMBER OF MISPOSITIONING ERRORS
2085	002144	000000			.WORD	0,0	; NUMBER OF ALL OTHER ERRORS
2086	002146	000			.BYTE	0,0	; ASSIGNMENT COMMAND SEQUENCE
2087	002147	000			.BYTE	0,0	; SEEK TO PREVIOUS COMMAND FLAG
2088	002150	000000			.WORD	0,0	; ERROR STATUS INFORMATION
2089	002152	000000			.WORD	0,0	; HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2090	002154	000			.BYTE	0,0	
2091	002155	000			.BYTE	0,0	; OFFSET ADDRESS FROM HEADER ADDRESS
2092	002156	000000			.WORD	0,0	; COMPARISON LENGTH
2093	002160	000000			.WORD	0,0	; BUFFER COMPARISON LENGTH
2094	002162	007777			.WORD	007777	; DRIVE SERIAL NUMBER
2095	002164	000000	000000		.WORD	0,0	; CARRIAGE SERIAL NUMBER
2096	002170	000000	000000		.WORD	0,0	; FIRST EXCLUSION AREA
2097	002174	000000	000000		.WORD	0,0	; SECOND EXCLUSION AREA
2098	002200	000000	000000		.WORD	0,0	; THIRD EXCLUSION AREA
2099	002204	000000	000000		.WORD	0,0	; FOURTH EXCLUSION AREA
2100	002210	000000	000000		.WORD	0,0	; FIFTH EXCLUSION AREA

```

2101
2102
2103
2104 002214 001
2105 002215 000
2106 002216 000000
2107 002220 000
2108 002221 000
2109 002222 000
2110 002223 000
2111 002224 000000
2112 002226 000000
2113 002230 000000
2114 002232 000000
2115 002234 000000
2116 002236 000000
2117 002240 000000
2118 002242 000000
2119 002244 000000
2120 002246 000000
2121 002250 000000
2122 002252 000000
2123 002254 000000
2124 002256 000000
2125 002260 000000
2126 002262 000000
2127 002264 000000
2128 002266 000000
2129 002270 000000
2130 002272 000000
2131 002274 000000
2132 002276 000000
2133 002300 000000
2134 002302 000000
2135 002304 000632
2136 002306 000
2137 002307 002
2138 002310 000
2139 002311 025
2140 002312 003
2141 002313 000
2142 002314 000012
2143 002316 000005
2144 002320 000005
2145 002322 177770 177777
2146 002326 177770 177777
2147 002332 000
2148 002333 000
2149 002334 000
2150 002335 000
2151 002336 000
2152 002337 000
2153 002340 000000
2154 002342 000
2155 002343 000
2156 002344 000000

```

.SBTTL PARAMETER BLOCK FOR DRIVE 1

```

PARM1: .BYTE 1 ;DRIVE 1
        .BYTE 0 ;COMMAND
        .WORD 0 ;CYLINDER ADDRESS
        .BYTE 0 ;SECTOR ADDRESS
        .BYTE 0 ;TRACK ADDRESS
        .BYTE 0 ;OFFSET VALUE
        .BYTE 0 ;BUS ADDRESS (BITS 16 AND 17)
        .WORD 0 ;BUS ADDRESS (BITS 0 - 15)
        .WORD 0 ;WORD COUNT (2'S COMPLEMENT)
        .WORD 0 ;PROGRAM DRIVE STATUS INFORMATION
        .WORD 0 ;COMMAND AND STATUS REGISTER 1
        .WORD 0 ;COMMAND AND STATUS REGISTER 2
        .WORD 0 ;WORD COUNT REGISTER
        .WORD 0 ;BUS ADDRESS REGISTER
        .WORD 0 ;DESIRED TRACK AND SECTOR REGISTER
        .WORD 0 ;DESIRED CYLINDER REGISTER
        .WORD 0 ;ATTENTION SUMMARY/OFFSET REGISTER
        .WORD 0 ;ERROR REGISTER
        .WORD 0 ;STATUS REGISTER
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 00
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 00
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 01
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 01
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 10
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 10
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 11
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 11
        .WORD 0 ;ECC POSITION INFORMATION
        .WORD 0 ;ECC PATTERN INFORMATION
        .WORD 0 ;QUEUE LINK
        .WORD 0 ;MINIMUM CYLINDER
        .WORD 410. ;MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
        .BYTE 0 ;MINIMUM TRACK
        .BYTE 2 ;MAXIMUM TRACK
        .BYTE 0 ;MINIMUM SECTOR
        .BYTE 21. ;MAXIMUM SECTOR
        .BYTE 3 ;READ/WRITE RATIO
        .BYTE 0 ;AUTOMATIC WRITE CHECK
        .WORD 10. ;CORRECTABLE READ ERROR THRESHOLD
        .WORD 5 ;UNCORRECTABLE READ ERROR THRESHOLD
        .WORD 5 ;SEEK ERROR THRESHOLD
        .WORD 177770,177777 ;MAXIMUM NUMBER OF COMMANDS TO DRIVE
        .WORD 177770,177777 ;MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
        .BYTE 0 ;SAMPLE COMPARE FLAG
        .BYTE 0 ;ECC COMPARE FLAG
        .BYTE 0 ;INHIBIT ERROR CORRECTION
        .BYTE 0 ;DATA PATTERN USED
        .BYTE 0 ;CURRENT GENERATED DATA PATTERN
        .BYTE 0 ;CURRENT RANDOMLY GENERATED COMMAND
        .WORD 0 ;CURRENT RANDOMLY GENERATED CYLINDER
        .BYTE 0 ;CURRENT RANDOMLY GENERATED SECTOR
        .BYTE 0 ;CURRENT RANDOMLY GENERATED TRACK
        .WORD 0 ;CURRENT RANDOMLY GENERATED WORD COUNT

```

2157	002346	000000			.WORD	0	; CURRENT RANDOMLY GENERATED BUS ADDRESS
2158	002350	000			.BYTE	0	
2159	002351	000			.BYTE	0	; LAST COMMAND
2160	002352	000000			.WORD	0	; LAST CYLINDER
2161	002354	000			.BYTE	0	; LAST SECTOR
2162	002355	000			.BYTE	0	; LAST TRACK
2163	002356	000000			.WORD	0	; LAST WORD COUNT
2164	002360	000			.BYTE	0	; LAST DATA PATTERN USED
2165	002361	000			.BYTE	0	; BUFFER ALLOCATED
2166	002362	000			.BYTE	0	; RETRY COUNT
2167	002363	000			.BYTE	0	; REREAD STATUS
2168	002364	000000			.WORD	0	; DRIVE STATUS FLAGS
2169	002366	000000	000000		.WORD	0,0	; NUMBER OF ORDERS
2170	002372	000000	000000	000000	.WORD	0,0,0	; NUMBER OF WORDS WRITTEN
2171	002400	000000	000000	000000	.WORD	0,0,0	; NUMBER OF WORDS READ
2172	002406	000000			.WORD	0	; NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2173	002410	000000			.WORD	0	; NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2174	002412	000000			.WORD	0	; NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2175	002414	000000			.WORD	0	; NUMBER OF HARD DATA ERRORS
2176	002416	000000			.WORD	0	; NUMBER OF SEEK INCOMPLETES
2177	002420	000000			.WORD	0	; NUMBER OF MISPOSITIONING ERRORS
2178	002422	000000			.WORD	0	; NUMBER OF ALL OTHER ERRORS
2179	002424	000			.BYTE	0	; ASSIGNMENT COMMAND SEQUENCE
2180	002425	000			.BYTE	0	; SEEK TO PREVIOUS COMMAND FLAG
2181	002426	000000			.WORD	0	; ERROR STATUS INFORMATION
2182	002430	000000			.WORD	0	; HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2183	002432	000			.BYTE	0	
2184	002433	000			.BYTE	0	; OFFSET ADDRESS FROM HEADER ADDRESS
2185	002434	000000			.WORD	0	; COMPARISON LENGTH
2186	002436	000000			.WORD	0	; BUFFER COMPARISON LENGTH
2187	002440	007777			.WORD	007777	; DRIVE SERIAL NUMBER
2188	002442	000000	000000		.WORD	0,0	; CARTRIDGE SERIAL NUMBER
2189	002446	000000	000000		.WORD	0,0	; FIRST EXCLUSION AREA
2190	002452	000000	000000		.WORD	0,0	; SECOND EXCLUSION AREA
2191	002456	000000	000000		.WORD	0,0	; THIRD EXCLUSION AREA
2192	002462	000000	000000		.WORD	0,0	; FOURTH EXCLUSION AREA
2193	002466	000000	000000		.WORD	0,0	; FIFTH EXCLUSION AREA

2194
2195
2196
2197 002472 002
2198 002473 000
2199 002474 000000
2200 002476 000
2201 002477 000
2202 002500 000
2203 002501 000
2204 002502 000(00
2205 002504 000(00
2206 002506 000(00
2207 002510 000(00
2208 002512 000000
2209 002514 000000
2210 002516 000000
2211 002520 000000
2212 002522 000000
2213 002524 000000
2214 002526 000000
2215 002530 000000
2216 002532 000000
2217 002534 000000
2218 002536 000000
2219 002540 000000
2220 002542 000000
2221 002544 000000
2222 002546 000000
2223 002550 000000
2224 002552 000000
2225 002554 000000
2226 002556 000000
2227 002560 000000
2228 002562 000632
2229 002564 000
2230 002565 002
2231 002566 000
2232 002567 025
2233 002570 003
2234 002571 000
2235 002572 000012
2236 002574 000005
2237 002576 000005
2238 002600 177770 177777
2239 002604 177770 177777
2240 002610 000
2241 002611 000
2242 002612 000
2243 002613 000
2244 002614 000
2245 002615 000
2246 002616 000000
2247 002620 000
2248 002621 000
2249 002622 000000

.SBTTL PARAMETER BLOCK FOR DRIVE 2

PARM2: .BYTE 2 : DRIVE 2
.BYTE 000 : COMMAND
.WORD 000 : CYLINDER ADDRESS
.BYTE 000 : SECTOR ADDRESS
.BYTE 000 : TRACK ADDRESS
.BYTE 000 : OFFSET VALUE
.BYTE 000 : BUS ADDRESS (BITS 16 AND 17)
.WORD 000 : BUS ADDRESS (BITS 0 - 15)
.WORD 000 : WORD COUNT (2'S COMPLEMENT)
.WORD 000 : PROGRAM DRIVE STATUS INFORMATION
.WORD 000 : COMMAND AND STATUS REGISTER 1
.WORD 000 : COMMAND AND STATUS REGISTER 2
.WORD 000 : WORD COUNT REGISTER
.WORD 000 : BUS ADDRESS REGISTER
.WORD 000 : DESIRED TRACK AND SECTOR REGISTER
.WORD 000 : DESIRED CYLINDER REGISTER
.WORD 000 : ATTENTION SUMMARY/OFFSET REGISTER
.WORD 000 : ERROR REGISTER
.WORD 000 : STATUS REGISTER
.WORD 000 : MESSAGE LINE A STATUS BYTE 00
.WORD 000 : MESSAGE LINE B STATUS BYTE 00
.WORD 000 : MESSAGE LINE A STATUS BYTE 01
.WORD 000 : MESSAGE LINE B STATUS BYTE 01
.WORD 000 : MESSAGE LINE A STATUS BYTE 10
.WORD 000 : MESSAGE LINE B STATUS BYTE 10
.WORD 000 : MESSAGE LINE A STATUS BYTE 11
.WORD 000 : MESSAGE LINE B STATUS BYTE 11
.WORD 000 : ECC POSITION INFORMATION
.WORD 000 : ECC PATTERN INFORMATION
.WORD 000 : QUEUE LINK
.WORD 000 : MINIMUM CYLINDER
.WORD 410. : MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
.BYTE 000 : MINIMUM TRACK
.BYTE 020 : MAXIMUM TRACK
.BYTE 021. : MINIMUM SECTOR
.BYTE 021. : MAXIMUM SECTOR
.BYTE 000 : READ/WRITE RATIO
.WORD 000 : AUTOMATIC WRITE CHECK
.WORD 10. : CORRECTABLE READ ERROR THRESHOLD
.WORD 5 : UNCORRECTABLE READ ERROR THRESHOLD
.WORD 5 : SEEK ERROR THRESHOLD
.WORD 177770,177777 : MAXIMUM NUMBER OF COMMANDS TO DRIVE
.WORD 177770,177777 : MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
.BYTE 0 : SAMPLE COMPARE FLAG
.BYTE 000 : ECC COMPARE FLAG
.BYTE 000 : INHIBIT ERROR CORRECTION
.BYTE 000 : DATA PATTERN USED
.WORD 000 : CURRENT GENERATED DATA PATTERN
.WORD 000 : CURRENT RANDOMLY GENERATED COMMAND
.WORD 000 : CURRENT RANDOMLY GENERATED CYLINDER
.BYTE 000 : CURRENT RANDOMLY GENERATED SECTOR
.BYTE 000 : CURRENT RANDOMLY GENERATED TRACK
.WORD 0 : CURRENT RANDOMLY GENERATED WORD COUNT

2250	002624	000000			.WORD	0			;CURRENT RANDOMLY GENERATED BUS ADDRESS
2251	002626	000			.BYTE	00			
2252	002627	000			.BYTE	00			;LAST COMMAND
2253	002630	000000			.WORD	00			;LAST CYLINDER
2254	002632	000			.BYTE	00			;LAST SECTOR
2255	002633	000			.BYTE	00			;LAST TRACK
2256	002634	000000			.WORD	00			;LAST WORD COUNT
2257	002636	000			.BYTE	00			;LAST DATA PATTERN USED
2258	002637	000			.BYTE	00			;BUFFER ALLOCATED
2259	002640	000			.BYTE	00			;RETRY COUNT
2260	002641	000			.BYTE	00			;REPEAT STATUS
2261	002642	000000			.WORD	00			;DRIVE STATUS FLAGS
2262	002644	000000	000000		.WORD	0,0			;NUMBER OF ORDERS
2263	002650	000000	000000	000000	.WORD	0,0,0			;NUMBER OF WORDS WRITTEN
2264	002656	000000	000000	000000	.WORD	0,0,0			;NUMBER OF WORDS READ
2265	002664	000000			.WORD	00			;NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2266	002666	000000			.WORD	00			;NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2267	002670	000000			.WORD	00			;NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2268	002672	000000			.WORD	00			;NUMBER OF HARD DATA ERRORS
2269	002674	000000			.WORD	00			;NUMBER OF SEEK INCOMPLETES
2270	002676	000000			.WORD	00			;NUMBER OF MISPOSITIONING ERRORS
2271	002700	000000			.WORD	00			;NUMBER OF ALL OTHER ERRORS
2272	002702	000			.BYTE	00			;ASSIGNMENT COMMAND SEQUENCE
2273	002703	000			.BYTE	00			;SEEK TO PREVIOUS COMMAND FLAG
2274	002704	000000			.WORD	00			;ERROR STATUS INFORMATION
2275	002706	000000			.WORD	00			;HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2276	002710	000			.BYTE	00			
2277	002711	000			.BYTE	00			;OFFSET ADDRESS FROM HEADER ADDRESS
2278	002712	000000			.WORD	00			;COMPARISON LENGTH
2279	002714	000000			.WORD	00			;BUFFER COMPARISON LENGTH
2280	002716	007777			.WORD	007777			;DRIVE SERIAL NUMBER
2281	002720	000000	000000		.WORD	0,0			;CARIAGE SERIAL NUMBER
2282	002724	000000	000000		.WORD	0,0			;FIRST EXCLUSION AREA
2283	002730	000000	000000		.WORD	0,0			;SECOND EXCLUSION AREA
2284	002734	000000	000000		.WORD	0,0			;THIRD EXCLUSION AREA
2285	002740	000000	000000		.WORD	0,0			;FOURTH EXCLUSION AREA
2286	002744	000000	000000		.WORD	0,0			;FIFTH EXCLUSION AREA

```

2287
2288
2289
2290 002750 003
2291 002751 000
2292 002752 000000
2293 002754 000
2294 002755 000
2295 002756 000
2296 002757 000
2297 002760 000000
2298 002762 000000
2299 002764 000000
2300 002766 000000
2301 002770 000000
2302 002772 000000
2303 002774 000000
2304 002776 000000
2305 003000 000000
2306 003002 000000
2307 003004 000000
2308 003006 000000
2309 003010 000000
2310 003012 000000
2311 003014 000000
2312 003016 000000
2313 003020 000000
2314 003022 000000
2315 003024 000000
2316 003026 000000
2317 003030 000000
2318 003032 000000
2319 003034 000000
2320 003036 000000
2321 003040 000632
2322 003042 000
2323 003043 002
2324 003044 000
2325 003045 025
2326 003046 003
2327 003047 000
2328 003050 000012
2329 003052 000005
2330 003054 000005
2331 003056 177770 177777
2332 003062 177770 177777
2333 003066 000
2334 003067 000
2335 003070 000
2336 003071 000
2337 003072 000
2338 003073 000
2339 003074 000000
2340 003076 000
2341 003077 000
2342 003100 000000

```

.SBTTL PARAMETER BLOCK FOR DRIVE 3

```

PARM3: .BYTE 3 ;DRIVE 3
        .BYTE 00 ;COMMAND
        .WORD 00 ;CYLINDER ADDRESS
        .BYTE 00 ;SECTOR ADDRESS
        .BYTE 00 ;TRACK ADDRESS
        .BYTE 00 ;OFFSET VALUE
        .BYTE 00 ;BUS ADDRESS (BITS 16 AND 17)
        .WORD 00 ;BUS ADDRESS (BITS 0 - 15)
        .WORD 00 ;WORD COUNT (2'S COMPLEMENT)
        .WORD 00 ;PROGRAM DRIVE STATUS INFORMATION
        .WORD 00 ;COMMAND AND STATUS REGISTER 1
        .WORD 00 ;COMMAND AND STATUS REGISTER 2
        .WORD 00 ;WORD COUNT REGISTER
        .WORD 00 ;BUS ADDRESS REGISTER
        .WORD 00 ;DESIRED TRACK AND SECTOR REGISTER
        .WORD 00 ;DESIRED CYLINDER REGISTER
        .WORD 00 ;ATTENTION SUMMARY/OFFSET REGISTER
        .WORD 00 ;ERROR REGISTER
        .WORD 00 ;STATUS REGISTER
        .WORD 00 ;MESSAGE LINE A STATUS BYTE 00
        .WORD 00 ;MESSAGE LINE B STATUS BYTE 00
        .WORD 00 ;MESSAGE LINE A STATUS BYTE 01
        .WORD 00 ;MESSAGE LINE B STATUS BYTE 01
        .WORD 00 ;MESSAGE LINE A STATUS BYTE 10
        .WORD 00 ;MESSAGE LINE B STATUS BYTE 10
        .WORD 00 ;MESSAGE LINE A STATUS BYTE 11
        .WORD 00 ;MESSAGE LINE B STATUS BYTE 11
        .WORD 00 ;ECC POSITION INFORMATION
        .WORD 00 ;ECC PATTERN INFORMATION
        .WORD 00 ;QUEUE LINK
        .WORD 00 ;MINIMUM CYLINDER
        .WORD 410. ;MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
        .BYTE 00 ;MINIMUM TRACK
        .BYTE 020 ;MAXIMUM TRACK
        .BYTE 20 ;MINIMUM SECTOR
        .BYTE 21. ;MAXIMUM SECTOR
        .BYTE 3 ;READ/WRITE RATIO
        .BYTE 0 ;AUTOMATIC WRITE CHECK
        .WORD 10. ;CORRECTABLE READ ERROR THRESHOLD
        .WORD 5 ;UNCORRECTABLE READ ERROR THRESHOLD
        .WORD 5 ;SEEK ERROR THRESHOLD
        .WORD 177770,177777 ;MAXIMUM NUMBER OF COMMANDS TO DRIVE
        .WORD 177770,177777 ;MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
        .BYTE 00 ;SAMPLE COMPARE FLAG
        .BYTE 00 ;ECC COMPARE FLAG
        .BYTE 00 ;INHIBIT ERROR CORRECTION
        .BYTE 00 ;DATA PATTERN USED
        .BYTE 00 ;CURRENT GENERATED DATA PATTERN
        .WORD 00 ;CURRENT RANDOMLY GENERATED COMMAND
        .WORD 00 ;CURRENT RANDOMLY GENERATED CYLINDER
        .BYTE 00 ;CURRENT RANDOMLY GENERATED SECTOR
        .BYTE 00 ;CURRENT RANDOMLY GENERATED TRACK
        .WORD 0 ;CURRENT RANDOMLY GENERATED WORD COUNT

```

2343	003102	000000			.WORD	0	; CURRENT RANDOMLY GENERATED BUS ADDRESS
2344	003104	000			.BYTE	0,0	; LAST COMMAND
2345	003105	000			.BYTE	0,0,0	; LAST CYLINDER
2346	003106	000000			.WORD	0,0,0,0	; LAST SECTOR
2347	003110	000			.BYTE	0,0,0,0	; LAST TRACK
2348	003111	000			.BYTE	0,0,0,0	; LAST WORD COUNT
2349	003112	000000			.WORD	0,0,0,0	; LAST DATA PATTERN USED
2350	003114	000			.BYTE	0,0,0,0	; BUFFER ALLOCATED
2351	003115	000			.BYTE	0,0,0,0	; RETRY COUNT
2352	003116	000			.BYTE	0,0,0,0	; REREAD STATUS
2353	003117	000			.BYTE	0,0,0,0	; DRIVE STATUS FLAGS
2354	003120	000000			.WORD	0,0,0,0	; NUMBER OF ORDERS
2355	003122	000000	000000		.WORD	0,0,0,0	; NUMBER OF WORDS WRITTEN
2356	003126	000000	000000	000000	.WORD	0,0,0,0	; NUMBER OF WORDS READ
2357	003134	000000	000000	000000	.WORD	0,0,0,0	; NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2358	003142	000000			.WORD	0,0,0,0	; NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2359	003144	000000			.WORD	0,0,0,0	; NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2360	003146	000000			.WORD	0,0,0,0	; NUMBER OF HARD DATA ERRORS
2361	003150	000000			.WORD	0,0,0,0	; NUMBER OF SEEK INCOMPLETES
2362	003152	000000			.WORD	0,0,0,0	; NUMBER OF MISPOSITIONING ERRORS
2363	003154	000000			.WORD	0,0,0,0	; NUMBER OF ALL OTHER ERRORS
2364	003156	000000			.WORD	0,0,0,0	; ASSIGNMENT COMMAND SEQUENCE
2365	003160	000			.BYTE	0,0,0,0	; SEEK TO PREVIOUS COMMAND FLAG
2366	003161	000			.BYTE	0,0,0,0	; ERROR STATUS INFORMATION
2367	003162	000000			.WORD	0,0,0,0	; HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2368	003164	000000			.WORD	0,0,0,0	; OFFSET ADDRESS FROM HEADER ADDRESS
2369	003166	000			.BYTE	0,0,0,0	; COMPARISON LENGTH
2370	003167	000			.BYTE	0,0,0,0	; BUFFER COMPARISON LENGTH
2371	003170	000000			.WORD	0,0,0,0	; DRIVE SERIAL NUMBER
2372	003172	000000			.WORD	0,0,0,0	; CARIAGE SERIAL NUMBER
2373	003174	007777			.WORD	007777	; FIRST EXCLUSION AREA
2374	003176	000000	000000		.WORD	0,0,0,0	; SECOND EXCLUSION AREA
2375	003202	000000	000000		.WORD	0,0,0,0	; THIRD EXCLUSION AREA
2376	003206	000000	000000		.WORD	0,0,0,0	; FOURTH EXCLUSION AREA
2377	003212	000000	000000		.WORD	0,0,0,0	; FIFTH EXCLUSION AREA
2378	003216	000000	000000		.WORD	0,0,0,0	
2379	003222	000000	000000		.WORD	0,0,0,0	

2380			.SBTTL	PARAMETER BLOCK FOR DRIVE 4	
2381			PARM4:	.BYTE	4
2382				.WORD	00
2383	003228	004		.BYTE	00
2384	003229	000		.WORD	00
2385	003230	000000		.BYTE	00
2386	003232	000		.BYTE	00
2387	003233	000		.WORD	00
2388	003234	000		.WORD	00
2389	003235	000		.WORD	00
2390	003236	000000		.WORD	00
2391	003240	000000		.WORD	00
2392	003242	000000		.WORD	00
2393	003244	000000		.WORD	00
2394	003246	000000		.WORD	00
2395	003250	000000		.WORD	00
2396	003252	000000		.WORD	00
2397	003254	000000		.WORD	00
2398	003256	000000		.WORD	00
2399	003260	000000		.WORD	00
2400	003262	000000		.WORD	00
2401	003264	000000		.WORD	00
2402	003266	000000		.WORD	00
2403	003270	000000		.WORD	00
2404	003272	000000		.WORD	00
2405	003274	000000		.WORD	00
2406	003276	000000		.WORD	00
2407	003300	000000		.WORD	00
2408	003302	000000		.WORD	00
2409	003304	000000		.WORD	00
2410	003306	000000		.WORD	00
2411	003310	000000		.WORD	00
2412	003312	000000		.WORD	00
2413	003314	000000		.WORD	00
2414	003316	000632		.WORD	410.
2415	003320	000		.BYTE	0
2416	003321	002		.BYTE	0
2417	003322	000		.BYTE	0
2418	003323	025		.BYTE	21.
2419	003324	003		.BYTE	3
2420	003325	000		.BYTE	0
2421	003326	000012		.WORD	10.
2422	003330	000005		.WORD	5
2423	003332	000005		.WORD	5
2424	003334	177770	177777	.WORD	177770,177777
2425	003340	177770	177777	.WORD	177770,177777
2426	003344	000		.BYTE	0
2427	003345	000		.BYTE	0
2428	003346	000		.BYTE	0
2429	003347	000		.BYTE	0
2430	003350	000		.BYTE	0
2431	003351	000		.BYTE	0
2432	003352	000000		.WORD	0
2433	003354	000		.BYTE	0
2434	003355	000		.BYTE	0
2435	003356	000000		.WORD	0

DRIVE 4
COMMAND
CYLINDER ADDRESS
SECTOR ADDRESS
TRACK ADDRESS
OFFSET VALUE
BUS ADDRESS (BITS 16 AND 17)
BUS ADDRESS (BITS 0 - 15)
WORD COUNT (2'S COMPLEMENT)
PROGRAM DRIVE STATUS INFORMATION
COMMAND AND STATUS REGISTER 1
COMMAND AND STATUS REGISTER 2
WORD COUNT REGISTER
BUS ADDRESS REGISTER
DESIRED TRACK AND SECTOR REGISTER
DESIRED CYLINDER REGISTER
ATTENTION SUMMARY/OFFSET REGISTER
ERROR REGISTER
STATUS REGISTER
MESSAGE LINE A STATUS BYTE 00
MESSAGE LINE B STATUS BYTE 00
MESSAGE LINE A STATUS BYTE 01
MESSAGE LINE B STATUS BYTE 01
MESSAGE LINE A STATUS BYTE 10
MESSAGE LINE B STATUS BYTE 10
MESSAGE LINE A STATUS BYTE 11
MESSAGE LINE B STATUS BYTE 11
ECC POSITION INFORMATION
ECC PATTERN INFORMATION
QUEUE LINK
MINIMUM CYLINDER
MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
MINIMUM TRACK
MAXIMUM TRACK
MINIMUM SECTOR
MAXIMUM SECTOR
READ/WRITE RATIO
AUTOMATIC WRITE CHECK
CORRECTABLE READ ERROR THRESHOLD
UNCORRECTABLE READ ERROR THRESHOLD
SEEK ERROR THRESHOLD
MAXIMUM NUMBER OF COMMANDS TO DRIVE
MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
SAMPLE COMPARE FLAG
ECC COMPARE FLAG
INHIBIT ERROR CORRECTION
DATA PATTERN USED
CURRENT GENERATED DATA PATTERN
CURRENT RANDOMLY GENERATED COMMAND
CURRENT RANDOMLY GENERATED CYLINDER
CURRENT RANDOMLY GENERATED SECTOR
CURRENT RANDOMLY GENERATED TRACK
CURRENT RANDOMLY GENERATED WORD COUNT

2436	003360	000000			.WORD	0	; CURRENT RANDOMLY GENERATED BUS ADDRESS
2437	003362	000			.BYTE	0,0	
2438	003363	000			.BYTE	0	; LAST COMMAND
2439	003364	000000			.WORD	0	; LAST CYLINDER
2440	003366	000			.BYTE	0,0	; LAST SECTOR
2441	003367	000			.BYTE	0	; LAST TRACK
2442	003370	000000			.WORD	0	; LAST WORD COUNT
2443	003372	000			.BYTE	0	; LAST DATA PATTERN USED
2444	003373	000			.BYTE	0	; BUFFER ALLOCATED
2445	003374	000			.BYTE	0	; RETRY COUNT
2446	003375	000			.BYTE	0	; REREAD STATUS
2447	003376	000000			.WORD	0	; DRIVE STATUS FLAGS
2448	003400	000000	000000		.WORD	0,0	; NUMBER OF ORDERS
2449	003404	000000	000000	000000	.WORD	0,0,0	; NUMBER OF WORDS WRITTEN
2450	003412	000000	000000	000000	.WORD	0,0,0	; NUMBER OF WORDS READ
2451	003420	000000			.WORD	0	; NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2452	003422	000000			.WORD	0	; NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2453	003424	000000			.WORD	0	; NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2454	003426	000000			.WORD	0	; NUMBER OF HARD DATA ERRORS
2455	003430	000000			.WORD	0	; NUMBER OF SEEK INCOMPLETES
2456	003432	000000			.WORD	0	; NUMBER OF MISPOSITIONING ERRORS
2457	003434	000000			.WORD	0	; NUMBER OF ALL OTHER ERRORS
2458	003436	000			.BYTE	0	; ASSIGNMENT COMMAND SEQUENCE
2459	003437	000			.BYTE	0	; SEEK TO PREVIOUS COMMAND FLAG
2460	003440	000000			.WORD	0	; ERROR STATUS INFORMATION
2461	003442	000000			.WORD	0	; HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2462	003444	000			.BYTE	0	
2463	003445	000			.BYTE	0	; OFFSET ADDRESS FROM HEADER ADDRESS
2464	003446	000000			.WORD	0	; COMPARISON LENGTH
2465	003450	000000			.WORD	0	; BUFFER COMPARISON LENGTH
2466	003452	007777			.WORD	007777	; DRIVE SERIAL NUMBER
2467	003454	000000	000000		.WORD	0,0	; CARIAGE SERIAL NUMBER
2468	003460	000000	000000		.WORD	0,0	; FIRST EXCLUSION AREA
2469	003464	000000	000000		.WORD	0,0	; SECOND EXCLUSION AREA
2470	003470	000000	000000		.WORD	0,0	; THIRD EXCLUSION AREA
2471	003474	000000	000000		.WORD	0,0	; FOURTH EXCLUSION AREA
2472	003500	000000	000000		.WORD	0,0	; FIFTH EXCLUSION AREA

2473
2474
2475
2476 003504 005
2477 003505 000
2478 003506 000000
2479 003510 000
2480 003511 000
2481 003512 000
2482 003513 000
2483 003514 000000
2484 003516 000000
2485 003520 000000
2486 003522 000000
2487 003524 000000
2488 003526 000000
2489 003530 000000
2490 003532 000000
2491 003534 000000
2492 003536 000000
2493 003540 000000
2494 003542 000000
2495 003544 000000
2496 003546 000000
2497 003550 000000
2498 003552 000000
2499 003554 000000
2500 003556 000000
2501 003560 000000
2502 003562 000000
2503 003564 000000
2504 003566 000000
2505 003570 000000
2506 003572 000000
2507 003574 000632
2508 003576 000
2509 003577 002
2510 003600 000
2511 003601 025
2512 003602 003
2513 003603 000
2514 003604 000012
2515 003606 000005
2516 003610 000005
2517 003612 177770 177777
2518 003616 177770 177777
2519 003622 000
2520 003623 000
2521 003624 000
2522 003625 000
2523 003626 000
2524 003627 000
2525 003630 000000
2526 003632 000
2527 003633 000
2528 003634 000000

.SBTTL PARAMETER BLOCK FOR DRIVE 5

PARMS: .BYTE 5 : DRIVE 5
 .BYTE 0 : COMMAND
 .WORD 00 : CYLINDER ADDRESS
 .BYTE 00 : SECTOR ADDRESS
 .BYTE 00 : TRACK ADDRESS
 .BYTE 00 : OFFSET VALUE
 .BYTE 00 : BUS ADDRESS (BITS 16 AND 17)
 .WORD 00 : BUS ADDRESS (BITS 0 - 15)
 .WORD 00 : WORD COUNT (2'S COMPLEMENT)
 .WORD 00 : PROGRAM DRIVE STATUS INFORMATION
 .WORD 00 : COMMAND AND STATUS REGISTER 1
 .WORD 00 : COMMAND AND STATUS REGISTER 2
 .WORD 00 : WORD COUNT REGISTER
 .WORD 00 : BUS ADDRESS REGISTER
 .WORD 00 : DESIRED TRACK AND SECTOR REGISTER
 .WORD 00 : DESIRED CYLINDER REGISTER
 .WORD 00 : ATTENTION SUMMARY/OFFSET REGISTER
 .WORD 00 : ERROR REGISTER
 .WORD 00 : STATUS REGISTER
 .WORD 00 : MESSAGE LINE A STATUS BYTE 00
 .WORD 00 : MESSAGE LINE B STATUS BYTE 00
 .WORD 00 : MESSAGE LINE A STATUS BYTE 01
 .WORD 00 : MESSAGE LINE B STATUS BYTE 01
 .WORD 00 : MESSAGE LINE A STATUS BYTE 10
 .WORD 00 : MESSAGE LINE B STATUS BYTE 10
 .WORD 00 : MESSAGE LINE A STATUS BYTE 11
 .WORD 00 : MESSAGE LINE B STATUS BYTE 11
 .WORD 00 : ECC POSITION INFORMATION
 .WORD 00 : ECC PATTERN INFORMATION
 .WORD 00 : QUEUE LINK
 .WORD 410. : MINIMUM CYLINDER
 .WORD 410. : MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
 .BYTE 00 : MINIMUM TRACK
 .BYTE 20 : MAXIMUM TRACK
 .BYTE 21. : MINIMUM SECTOR
 .BYTE 33. : MAXIMUM SECTOR
 .BYTE 10. : READ/WRITE RATIO
 .WORD 5 : AUTOMATIC WRITE CHECK
 .WORD 5 : CORRECTABLE READ ERROR THRESHOLD
 .WORD 5 : UNCORRECTABLE READ ERROR THRESHOLD
 .WORD 5 : SEEK ERROR THRESHOLD
 .WORD 177770,177777 : MAXIMUM NUMBER OF COMMANDS TO DRIVE
 .WORD 177770,177777 : MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
 .BYTE 00 : SAMPLE COMPARE FLAG
 .BYTE 00 : ECC COMPARE FLAG
 .BYTE 00 : INHIBIT ERROR CORRECTION
 .BYTE 00 : DATA PATTERN USED
 .WORD 00 : CURRENT GENERATED DATA PATTERN
 .WORD 00 : CURRENT RANDOMLY GENERATED COMMAND
 .WORD 00 : CURRENT RANDOMLY GENERATED CYLINDER
 .WORD 00 : CURRENT RANDOMLY GENERATED SECTOR
 .WORD 00 : CURRENT RANDOMLY GENERATED TRACK
 .WORD 00 : CURRENT RANDOMLY GENERATED WORD COUNT

2529	003636	000000			.WORD	0		;CURRENT RANDOMLY GENERATED BUS ADDRESS
2530	003640	000			.BYTE	0,0		
2531	003641	000			.BYTE	0,0		;LAST COMMAND
2532	003642	000000			.WORD	0,0		;LAST CYLINDER
2533	003644	000			.BYTE	0,0		;LAST SECTOR
2534	003645	000			.BYTE	0,0		;LAST TRACK
2535	003646	000000			.WORD	0,0		;LAST WORD COUNT
2536	003650	000			.BYTE	0,0		;LAST DATA PATTERN USED
2537	003651	000			.BYTE	0,0		;BUFFER ALLOCATED
2538	003652	000			.BYTE	0,0		;RETRY COUNT
2539	003653	000			.BYTE	0,0		;REREAD STATUS
2540	003654	000000			.WORD	0,0		;DRIVE STATUS FLAGS
2541	003656	000000	000000		.WORD	0,0,0		;NUMBER OF ORDERS
2542	003662	000000	000000	000000	.WORD	0,0,0,0		;NUMBER OF WORDS WRITTEN
2543	003670	000000	000000	000000	.WORD	0,0,0,0		;NUMBER OF WORDS READ
2544	003676	000000			.WORD	0,0		;NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2545	003700	000000			.WORD	0,0		;NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2546	003702	000000			.WORD	0,0		;NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2547	003704	000000			.WORD	0,0		;NUMBER OF HARD DATA ERRORS
2548	003706	000000			.WORD	0,0		;NUMBER OF SEEK INCOMPLETES
2549	003710	000000			.WORD	0,0		;NUMBER OF MISPOSITIONING ERRORS
2550	003712	000000			.WORD	0,0		;NUMBER OF ALL OTHER ERRORS
2551	003714	000			.BYTE	0,0		;ASSIGNMENT COMMAND SEQUENCE
2552	003715	000			.BYTE	0,0		;SEEK TO PREVIOUS COMMAND FLAG
2553	003716	000000			.WORD	0,0		;ERROR STATUS INFORMATION
2554	003720	000000			.WORD	0,0		;HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2555	003722	000			.BYTE	0,0		
2556	003723	000			.BYTE	0,0		;OFFSET ADDRESS FROM HEADER ADDRESS
2557	003724	000000			.WORD	0,0		;COMPARISON LENGTH
2558	003726	000000			.WORD	0,0		;BUFFER COMPARISON LENGTH
2559	003730	007777			.WORD	007777		;DRIVE SERIAL NUMBER
2560	003732	000000	000000		.WORD	0,0		;CARIAGE SERIAL NUMBER
2561	003736	000000	000000		.WORD	0,0		;FIRST EXCLUSION AREA
2562	003742	000000	000000		.WORD	0,0		;SECOND EXCLUSION AREA
2563	003746	000000	000000		.WORD	0,0		;THIRD EXCLUSION AREA
2564	003752	000000	000000		.WORD	0,0		;FOURTH EXCLUSION AREA
2565	003756	000000	000000		.WORD	0,0		;FIFTH EXCLUSION AREA

```

2566
2567
2568
2569 003762 006
2570 003763 000
2571 003764 000000
2572 003766 000
2573 003767 000
2574 003770 000
2575 003771 000
2576 003772 000000
2577 003774 000000
2578 003776 000000
2579 004000 000000
2580 004002 000000
2581 004004 000000
2582 004006 000000
2583 004010 000000
2584 004012 000000
2585 004014 000000
2586 004016 000000
2587 004020 000000
2588 004022 000000
2589 004024 000000
2590 004026 000000
2591 004030 000000
2592 004032 000000
2593 004034 000000
2594 004036 000000
2595 004040 000000
2596 004042 000000
2597 004044 000000
2598 004046 000000
2599 004050 000000
2600 004052 000632
2601 004054 000
2602 004055 002
2603 004056 000
2604 004057 025
2605 004060 003
2606 004061 000
2607 004062 000012
2608 004064 000005
2609 004066 000005
2610 004070 177770 177777
2611 004074 177770 177777
2612 004100 000
2613 004101 000
2614 004102 000
2615 004103 000
2616 004104 000
2617 004105 000
2618 004106 000000
2619 004110 000
2620 004111 000
2621 004112 000000

```

.SBTTL PARAMETER BLOCK FOR DRIVE 6

```

PARM6: .BYTE 6 : DRIVE 6
        .BYTE 00 : COMMAND
        .WORD 00 : CYLINDER ADDRESS
        .BYTE 00 : SECTOR ADDRESS
        .BYTE 00 : TRACK ADDRESS
        .BYTE 00 : OFFSET VALUE
        .WORD 00 : BUS ADDRESS (BITS 16 AND 17)
        .WORD 00 : BUS ADDRESS (BITS 0 - 15)
        .WORD 00 : WORD COUNT (2'S COMPLEMENT)
        .WORD 00 : PROGRAM DRIVE STATUS INFORMATION
        .WORD 00 : COMMAND AND STATUS REGISTER 1
        .WORD 00 : COMMAND AND STATUS REGISTER 2
        .WORD 00 : WORD COUNT REGISTER
        .WORD 00 : BUS ADDRESS REGISTER
        .WORD 00 : DESIRED TRACK AND SECTOR REGISTER
        .WORD 00 : DESIRED CYLINDER REGISTER
        .WORD 00 : ATTENTION SUMMARY/OFFSET REGISTER
        .WORD 00 : ERROR REGISTER
        .WORD 00 : STATUS REGISTER
        .WORD 00 : MESSAGE LINE A STATUS BYTE 00
        .WORD 00 : MESSAGE LINE B STATUS BYTE 00
        .WORD 00 : MESSAGE LINE A STATUS BYTE 01
        .WORD 00 : MESSAGE LINE B STATUS BYTE 01
        .WORD 00 : MESSAGE LINE A STATUS BYTE 10
        .WORD 00 : MESSAGE LINE B STATUS BYTE 10
        .WORD 00 : MESSAGE LINE A STATUS BYTE 11
        .WORD 00 : MESSAGE LINE B STATUS BYTE 11
        .WORD 00 : ECC POSITION INFORMATION
        .WORD 00 : ECC PATTERN INFORMATION
        .WORD 00 : QUEUE LINK
        .WORD 00 : MINIMUM CYLINDER
        .WORD 410. : MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
        .BYTE 00 : MINIMUM TRACK
        .BYTE 20 : MAXIMUM TRACK
        .BYTE 00 : MINIMUM SECTOR
        .BYTE 21. : MAXIMUM SECTOR
        .BYTE 30 : READ/WRITE RATIO
        .BYTE 00 : AUTOMATIC WRITE CHECK
        .WORD 10. : CORRECTABLE READ ERROR THRESHOLD
        .WORD 50 : UNCORRECTABLE READ ERROR THRESHOLD
        .WORD 50 : SEEK ERROR THRESHOLD
        .WORD 177770,177777 : MAXIMUM NUMBER OF COMMANDS TO DRIVE
        .WORD 177770,177777 : MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
        .BYTE 00 : SAMPLE COMPARE FLAG
        .BYTE 00 : ECC COMPARE FLAG
        .BYTE 00 : INHIBIT ERROR CORRECTION
        .BYTE 00 : DATA PATTERN USED
        .BYTE 00 : CURRENT GENERATED DATA PATTERN
        .WORD 00 : CURRENT RANDOMLY GENERATED COMMAND
        .WORD 00 : CURRENT RANDOMLY GENERATED CYLINDER
        .WORD 00 : CURRENT RANDOMLY GENERATED SECTOR
        .BYTE 00 : CURRENT RANDOMLY GENERATED TRACK
        .WORD 00 : CURRENT RANDOMLY GENERATED WORD COUNT

```

2622	004114	000000			.WORD	0				; CURRENT RANDOMLY GENERATED BUS ADDRESS
2623	004116	000			.BYTE	0				
2624	004117	000			.BYTE	0				; LAST COMMAND
2625	004120	000000			.WORD	0				; LAST CYLINDER
2626	004122	000			.BYTE	0				; LAST SECTOR
2627	004123	000			.BYTE	0				; LAST TRACK
2628	004124	000000			.WORD	0				; LAST WORD COUNT
2629	004126	000			.BYTE	0				; LAST DATA PATTERN USED
2630	004127	000			.BYTE	0				; BUFFER ALLOCATED
2631	004130	000			.BYTE	0				; RETRY COUNT
2632	004131	000			.BYTE	0				; RERead STATUS
2633	004132	000000			.WORD	0				; DRIVE STATUS FLAGS
2634	004134	000000	000000		.WORD	0,0				; NUMBER OF ORDERS
2635	004140	000000	000000	000000	.WORD	0,0,0				; NUMBER OF WORDS WRITTEN
2636	004146	000000	000000	000000	.WORD	0,0,0				; NUMBER OF WORDS READ
2637	004154	000000			.WORD	0				; NUMBER OF SOFT ERRORS (RERead CORRECTABLE)
2638	004156	000000			.WORD	0				; NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2639	004160	000000			.WORD	0				; NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2640	004162	000000			.WORD	0				; NUMBER OF HARD DATA ERRORS
2641	004164	000000			.WORD	0				; NUMBER OF SEEK INCOMPLETES
2642	004166	000000			.WORD	0				; NUMBER OF MISPOSITIONING ERRORS
2643	004170	000000			.WORD	0				; NUMBER OF ALL OTHER ERRORS
2644	004172	000			.BYTE	0				; ASSIGNMENT COMMAND SEQUENCE
2645	004173	000			.BYTE	0				; SEEK TO PREVIOUS COMMAND FLAG
2646	004174	000000			.WORD	0				; ERROR STATUS INFORMATION
2647	004176	000000			.WORD	0				; HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2648	004200	000			.BYTE	0				
2649	004201	000			.BYTE	0				; OFFSET ADDRESS FROM HEADER ADDRESS
2650	004202	000000			.WORD	0				; COMPARISON LENGTH
2651	004204	000000			.WORD	0				; BUFFER COMPARISON LENGTH
2652	004206	007777			.WORD	007777				; DRIVE SERIAL NUMBER
2653	004210	000000	000000		.WORD	0,0				; CARIAGE SERIAL NUMBER
2654	004214	000000	000000		.WORD	0,0				; FIRST EXCLUSION AREA
2655	004220	000000	000000		.WORD	0,0				; SECOND EXCLUSION AREA
2656	004224	000000	000000		.WORD	0,0				; THIRD EXCLUSION AREA
2657	004230	000000	000000		.WORD	0,0				; FOURTH EXCLUSION AREA
2658	004234	000000	000000		.WORD	0,0				; FIFTH EXCLUSION AREA

```

2659
2660 .SBTTL PARAMETER BLOCK FOR DRIVE 7
2661
2662 004240 007 PARM7: .BYTE 7 ;DRIVE ;
2663 004241 000 .BYTE 0 ;COMMAND
2664 004242 000000 .WORD 0 ;CYLINDER ADDRESS
2665 004244 000 .BYTE 0 ;SECTOR ADDRESS
2666 004245 000 .BYTE 0 ;TRACK ADDRESS
2667 004246 000 .BYTE 0 ;OFFSET VALUE
2668 004247 000 .BYTE 0 ;BUS ADDRESS (BITS 16 AND 17)
2669 004250 000000 .WORD 0 ;BUS ADDRESS (BITS 0 - 15)
2700 004252 000000 .WORD 0 ;WORD COUNT (2'S COMPLEMENT)
2671 004254 000000 .WORD 0 ;PROGRAM DRIVE STATUS INFORMATION
2672 004256 000000 .WORD 0 ;COMMAND AND STATUS REGISTER 1
2673 004260 000000 .WORD 0 ;COMMAND AND STATUS REGISTER 2
2674 004262 000000 .WORD 0 ;WORD COUNT REGISTER
2675 004264 000000 .WORD 0 ;BUS ADDRESS REGISTER
2676 004266 000000 .WORD 0 ;DESIRED TRACK AND SECTOR REGISTER
2677 004270 000000 .WORD 0 ;DESIRED CYLINDER REGISTER
2678 004272 000000 .WORD 0 ;ATTENTION SUMMARY/OFFSET REGISTER
2679 004274 000000 .WORD 0 ;ERROR REGISTER
2680 004276 000000 .WORD 0 ;STATUS REGISTER
2681 004300 000000 .WORD 0 ;MESSAGE LINE A STATUS BYTE 00
2682 004302 000000 .WORD 0 ;MESSAGE LINE B STATUS BYTE 00
2683 004304 000000 .WORD 0 ;MESSAGE LINE A STATUS BYTE 01
2684 004306 000000 .WORD 0 ;MESSAGE LINE B STATUS BYTE 01
2685 004310 000000 .WORD 0 ;MESSAGE LINE A STATUS BYTE 10
2686 004312 000000 .WORD 0 ;MESSAGE LINE B STATUS BYTE 10
2687 004314 000000 .WORD 0 ;MESSAGE LINE A STATUS BYTE 11
2688 004316 000000 .WORD 0 ;MESSAGE LINE B STATUS BYTE 11
2689 004320 000000 .WORD 0 ;ECC POSITION INFORMATION
2690 004322 000000 .WORD 0 ;ECC PATTERN INFORMATION
2691 004324 000000 .WORD 0 ;QUEUE LINK
2692 004326 000000 .WORD 0 ;MINIMUM CYLINDER
2693 004330 000632 .WORD 410. ;MAXIMUM CYLINDER (DEFAULT TO RK06 MAX CYL)
2694 004332 000 .BYTE 0 ;MINIMUM TRACK
2695 004333 002 .BYTE 20 ;MAXIMUM TRACK
2696 004334 000 .BYTE 0 ;MINIMUM SECTOR
2697 004335 025 .BYTE 21. ;MAXIMUM SECTOR
2698 004336 003 .BYTE 3 ;READ/WRITE RATIO
2699 004337 000 .BYTE 0 ;AUTOMATIC WRITE CHECK
2700 004340 000012 .WORD 10. ;CORRECTABLE READ ERROR THRESHOLD
2701 004342 000005 .WORD 5 ;UNCORRECTABLE READ ERROR THRESHOLD
2702 004344 000005 .WORD 5 ;SEEK ERROR THRESHOLD
2703 004346 177770 177777 .WORD 177770,177777 ;MAXIMUM NUMBER OF COMMANDS TO DRIVE
2704 004352 177770 177777 .WORD 177770,177777 ;MAXIMUM NUMBER OF WORDS TRANSFERRED TO DRIVE
2705 004356 000 .BYTE 0 ;SAMPLE COMPARE FLAG
2706 004357 000 .BYTE 0 ;ECC COMPARE FLAG
2707 004360 000 .BYTE 0 ;INHIBIT ERROR CORRECTION
2708 004361 000 .BYTE 0 ;DATA PATTERN USED
2709 004362 000 .BYTE 0 ;CURRENT GENERATED DATA PATTERN
2710 004363 000 .BYTE 0 ;CURRENT RANDOMLY GENERATED COMMAND
2711 004364 000000 .WORD 0 ;CURRENT RANDOMLY GENERATED CYLINDER
2712 004366 000 .BYTE 0 ;CURRENT RANDOMLY GENERATED SECTOR
2713 004367 000 .BYTE 0 ;CURRENT RANDOMLY GENERATED TRACK
2714 004370 000000 .WORD 0 ;CURRENT RANDOMLY GENERATED WORD COUNT

```

2715	004372	000000			.WORD	0				;CURRENT RANDOMLY GENERATED BUS ADDRESS
2716	004374	000			.BYTE	00				
2717	004375	000			.BYTE	00				;LAST COMMAND
2718	004376	000000			.WORD	00				;LAST CYLINDER
2719	004400	000			.BYTE	00				;LAST SECTOR
2720	004401	000			.BYTE	00				;LAST TRACK
2721	004402	000000			.WORD	00				;LAST WORD COUNT
2722	004404	000			.BYTE	00				;LAST DATA PATTERN USED
2723	004405	000			.BYTE	00				;BUFFER ALLOCATED
2724	004406	000			.BYTE	00				;RETRY COUNT
2725	004407	000			.BYTE	00				;REREAD STATUS
2726	004410	000000			.WORD	00				;DRIVE STATUS FLAGS
2727	004412	000000	000000		.WORD	0,0				;NUMBER OF ORDERS
2728	004416	000000	000000	000000	.WORD	0,0,0				;NUMBER OF WORDS WRITTEN
2729	004424	000000	000000	000000	.WORD	0,0,0				;NUMBER OF WORDS READ
2730	004432	000000			.WORD	00				;NUMBER OF SOFT ERRORS (REREAD CORRECTABLE)
2731	004434	000000			.WORD	00				;NUMBER OF SOFT ERRORS (OFFSET CORRECTABLE)
2732	004436	000000			.WORD	00				;NUMBER OF SOFT ERRORS (ECC CORRECTABLE)
2733	004440	000000			.WORD	00				;NUMBER OF HARD DATA ERRORS
2734	004442	000000			.WORD	00				;NUMBER OF SEEK INCOMPLETES
2735	004444	000000			.WORD	00				;NUMBER OF MISPOSITIONING ERRORS
2736	004446	000000			.WORD	00				;NUMBER OF ALL OTHER ERRORS
2737	004450	000			.BYTE	00				;ASSIGNMENT COMMAND SEQUENCE
2738	004451	000			.BYTE	00				;SEEK TO PREVIOUS COMMAND FLAG
2739	004452	000000			.WORD	00				;ERROR STATUS INFORMATION
2740	004454	000000			.WORD	00				;HEADER ADDRESS OF FIRST MISCOMPARE ERROR
2741	004456	000			.BYTE	00				
2742	004457	000			.BYTE	00				;OFFSET ADDRESS FROM HEADER ADDRESS
2743	004460	000000			.WORD	00				;COMPARISON LENGTH
2744	004462	000000			.WORD	00				;BUFFER COMPARISON LENGTH
2745	004464	007777			.WORD	007777				;DRIVE SERIAL NUMBER
2746	004466	000000	000000		.WORD	0,0				;CARIAGE SERIAL NUMBER
2747	004472	000000	000000		.WORD	0,0				;FIRST EXCLUSION AREA
2748	004476	000000	000000		.WORD	0,0				;SECOND EXCLUSION AREA
2749	004502	000000	000000		.WORD	0,0				;THIRD EXCLUSION AREA
2750	004506	000000	000000		.WORD	0,0				;FOURTH EXCLUSION AREA
2751	004512	000000	000000		.WORD	0,0				;FIFTH EXCLUSION AREA


```

2752
2753
2754
2755
2756 004516      000
2757 004517      000
2758 004520 000000
2759 004522      000
2760 004523      000
2761 004524      000
2762 004525      000
2763 004526 000000
2764 004530 000000
2765 004532 000000
2766 004534 000000
2767 004536 000000
2768 004540 000000
2769 004542 000000
2770 004544 000000
2771 004546 000000
2772 004550 000000
2773 004552 000000
2774 004554 000000
2775 004556 000000
2776 004560 000000
2777 004562 000000
2778 004564 000000
2779 004566 000000
2780 004570 000000
2781 004572 000000
2782 004574 000000
2783 004576 000000
2784 004600 000000

```

.SBTTL --- RETRY PARAMETER BLOCK ---

.SBTTL PARAMETER BLOCK 10 FOR DRIVE

```

PARM10: .BYTE 0 ;DRIVE NUMBER
        .BYTE 0 ;COMMAND
        .WORD 0 ;CYLINDER ADDRESS
        .BYTE 0 ;SECTOR ADDRESS
        .BYTE 0 ;TRACK ADDRESS
        .BYTE 0 ;OFFSET VALUE
        .BYTE 0 ;BUS ADDRESS (BITS 16 AND 17)
        .WORD 0 ;BUS ADDRESS (BITS 0 - 15)
        .WORD 0 ;WORD COUNT (2'S COMPLEMENT)
        .WORD 0 ;PROGRAM DRIVE STATUS INFORMATION
        .WORD 0 ;COMMAND AND STATUS REGISTER 1
        .WORD 0 ;COMMAND AND STATUS REGISTER 2
        .WORD 0 ;WORD COUNT REGISTER
        .WORD 0 ;BUS ADDRESS REGISTER
        .WORD 0 ;DESIRED TRACK AND SECTOR REGISTER
        .WORD 0 ;DESIRED CYLINDER REGISTER
        .WORD 0 ;ATTENTION SUMMARY/OFFSET REGISTER
        .WORD 0 ;ERROR REGISTER
        .WORD 0 ;STATUS REGISTER
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 00
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 00
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 01
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 01
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 10
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 10
        .WORD 0 ;MESSAGE LINE A STATUS BYTE 11
        .WORD 0 ;MESSAGE LINE B STATUS BYTE 11
        .WORD 0 ;ECC POSITION INFORMATION
        .WORD 0 ;ECC PATTERN INFORMATION

```

.SBTTL DATA PATTERNS AND DATA PATTERN TABLE

PATTAB: PAT0
PAT1
PAT2
PAT3
PAT4
PAT5
PAT6
PAT7
PAT10
PAT11
PAT12
PAT13
PAT14
PAT15
PAT16
PAT17

PATO: .WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333
.WORD 165555
.WORD 133333

PAT1: .WORD 026455
.WORD 151322
.WORD 026455
.WORD 026455

2785		
2786		
2787	004602	004642
2788	004604	004742
2789	004606	005042
2790	004610	005142
2791	004612	005242
2792	004614	005342
2793	004616	005442
2794	004620	005542
2795	004622	005642
2796	004624	005742
2797	004626	006042
2798	004630	006142
2799	004632	006242
2800	004634	006342
2801	004636	006442
2802	004640	006542
2803		
2804	004642	165555
2805	004644	133333
2806	004646	165555
2807	004650	133333
2808	004652	165555
2809	004654	133333
2810	004656	165555
2811	004660	133333
2812	004662	165555
2813	004664	133333
2814	004666	165555
2815	004670	133333
2816	004672	165555
2817	004674	133333
2818	004676	165555
2819	004700	133333
2820	004702	165555
2821	004704	133333
2822	004706	165555
2823	004710	133333
2824	004712	165555
2825	004714	133333
2826	004716	165555
2827	004720	133333
2828	004722	165555
2829	004724	133333
2830	004726	165555
2831	004730	133333
2832	004732	165555
2833	004734	133333
2834	004736	165555
2835	004740	133333
2836		
2837	004742	026455
2838	004744	151322
2839	004746	026455
2840	004750	026455

2841	004752	151322	.WORD	151322
2842	004754	151322	.WORD	151322
2843	004756	026455	.WORD	026455
2844	004760	026455	.WORD	026455
2845	004762	026455	.WORD	026455
2846	004764	151322	.WORD	151322
2847	004766	151322	.WORD	151322
2848	004770	151322	.WORD	151322
2849	004772	026455	.WORD	026455
2850	004774	026455	.WORD	026455
2851	004776	026455	.WORD	026455
2852	005000	026455	.WORD	026455
2853	005002	151322	.WORD	151322
2854	005004	151322	.WORD	151322
2855	005006	151322	.WORD	151322
2856	005010	151322	.WORD	151322
2857	005012	026455	.WORD	026455
2858	005014	026455	.WORD	026455
2859	005016	026455	.WORD	026455
2860	005020	151322	.WORD	151322
2861	005022	151322	.WORD	151322
2862	005024	151322	.WORD	151322
2863	005026	026455	.WORD	026455
2864	005030	026455	.WORD	026455
2865	005032	151322	.WORD	151322
2866	005034	151322	.WORD	151322
2867	005036	026455	.WORD	026455
2868	005040	151322	.WORD	151322
2869				
2870	005042	007417	PAT2: .WORD	007417
2871	005044	170360	.WORD	170360
2872	005046	007417	.WORD	007417
2873	005050	007417	.WORD	007417
2874	005052	170360	.WORD	170360
2875	005054	170360	.WORD	170360
2876	005056	007417	.WORD	007417
2877	005060	007417	.WORD	007417
2878	005062	007417	.WORD	007417
2879	005064	170360	.WORD	170360
2880	005066	170360	.WORD	170360
2881	005070	170360	.WORD	170360
2882	005072	007417	.WORD	007417
2883	005074	007417	.WORD	007417
2884	005076	007417	.WORD	007417
2885	005100	007417	.WORD	007417
2886	005102	170360	.WORD	170360
2887	005104	170360	.WORD	170360
2888	005106	170360	.WORD	170360
2889	005110	170360	.WORD	170360
2890	005112	007417	.WORD	007417
2891	005114	007417	.WORD	007417
2892	005116	007417	.WORD	007417
2893	005120	170360	.WORD	170360
2894	005122	170360	.WORD	170360
2895	005124	170360	.WORD	170360
2896	005126	007417	.WORD	007417

2897	005130	007417	.WORD	007417
2898	005132	170360	.WORD	170360
2899	005134	170360	.WORD	170360
2900	005136	007417	.WORD	007417
2901	005140	170360	.WORD	170360
2902				
2903	005142	052525	PAT3: .WORD	052525
2904	005144	125252	.WORD	125252
2905	005146	052525	.WORD	052525
2906	005150	052525	.WORD	052525
2907	005152	125252	.WORD	125252
2908	005154	125252	.WORD	125252
2909	005156	052525	.WORD	052525
2910	005160	052525	.WORD	052525
2911	005162	052525	.WORD	052525
2912	005164	125252	.WORD	125252
2913	005166	125252	.WORD	125252
2914	005170	125252	.WORD	125252
2915	005172	052525	.WORD	052525
2916	005174	052525	.WORD	052525
2917	005176	052525	.WORD	052525
2918	005200	052525	.WORD	052525
2919	005202	125252	.WORD	125252
2920	005204	125252	.WORD	125252
2921	005206	125252	.WORD	125252
2922	005210	125252	.WORD	125252
2923	005212	052525	.WORD	052525
2924	005214	052525	.WORD	052525
2925	005216	052525	.WORD	052525
2926	005220	125252	.WORD	125252
2927	005222	125252	.WORD	125252
2928	005224	125252	.WORD	125252
2929	005226	052525	.WORD	052525
2930	005230	052525	.WORD	052525
2931	005232	125252	.WORD	125252
2932	005234	125252	.WORD	125252
2933	005236	052525	.WORD	052525
2934	005240	125252	.WORD	125252
2935				
2936	005242	000000	PAT4: .WORD	000000
2937	005244	010421	.WORD	010421
2938	005246	021042	.WORD	021042
2939	005250	031463	.WORD	031463
2940	005252	042104	.WORD	042104
2941	005254	052525	.WORD	052525
2942	005256	063146	.WORD	063146
2943	005260	073567	.WORD	073567
2944	005262	104210	.WORD	104210
2945	005264	114631	.WORD	114631
2946	005266	125252	.WORD	125252
2947	005270	135673	.WORD	135673
2948	005272	146314	.WORD	146314
2949	005274	156735	.WORD	156735
2950	005276	167356	.WORD	167356
2951	005300	177777	.WORD	177777
2952	005302	177777	.WORD	177777

2953	005304	167356	.WORD	167356
2954	005306	156735	.WORD	156735
2955	005310	146314	.WORD	146314
2956	005312	135673	.WORD	135673
2957	005314	125252	.WORD	125252
2958	005316	114631	.WORD	114631
2959	005320	104210	.WORD	104210
2960	005322	073567	.WORD	073567
2961	005324	063146	.WORD	063146
2962	005326	052525	.WORD	052525
2963	005330	042104	.WORD	042104
2964	005332	031463	.WORD	031463
2965	005334	021042	.WORD	021042
2966	005336	010421	.WORD	010421
2967	005340	000000	.WORD	000000
2968				
2969	005342	000000	PATS: .WORD	000000
2970	005344	177777	.WORD	177777
2971	005346	000000	.WORD	000000
2972	005350	000000	.WORD	000000
2973	005352	177777	.WORD	177777
2974	005354	177777	.WORD	177777
2975	005356	000000	.WORD	000000
2976	005360	000000	.WORD	000000
2977	005362	000000	.WORD	000000
2978	005364	177777	.WORD	177777
2979	005366	177777	.WORD	177777
2980	005370	177777	.WORD	177777
2981	005372	000000	.WORD	000000
2982	005374	000000	.WORD	000000
2983	005376	000000	.WORD	000000
2984	005400	000000	.WORD	000000
2985	005402	177777	.WORD	177777
2986	005404	177777	.WORD	177777
2987	005406	177777	.WORD	177777
2988	005410	177777	.WORD	177777
2989	005412	000000	.WORD	000000
2990	005414	000000	.WORD	000000
2991	005416	000000	.WORD	000000
2992	005420	177777	.WORD	177777
2993	005422	177777	.WORD	177777
2994	005424	177777	.WORD	177777
2995	005426	000000	.WORD	000000
2996	005430	000000	.WORD	000000
2997	005432	177777	.WORD	177777
2998	005434	177777	.WORD	177777
2999	005436	000000	.WORD	000000
3000	005440	177777	.WORD	177777
3001				
3002	005442	000001	PAT6: .WORD	000001
3003	005444	000003	.WORD	000003
3004	005446	000007	.WORD	000007
3005	005450	000017	.WORD	000017
3006	005452	000037	.WORD	000037
3007	005454	000077	.WORD	000077
3008	005456	000177	.WORD	000177

3009	005460	000377	.WORD	000377
3010	005462	000777	.WORD	000777
3011	005464	001777	.WORD	001777
3012	005466	003777	.WORD	003777
3013	005470	007777	.WORD	007777
3014	005472	017777	.WORD	017777
3015	005474	037777	.WORD	037777
3016	005476	077777	.WORD	077777
3017	005500	177777	.WORD	177777
3018	005502	177776	.WORD	177776
3019	005504	177774	.WORD	177774
3020	005506	177770	.WORD	177770
3021	005510	177760	.WORD	177760
3022	005512	177740	.WORD	177740
3023	005514	177700	.WORD	177700
3024	005516	177600	.WORD	177600
3025	005520	177400	.WORD	177400
3026	005522	177000	.WORD	177000
3027	005524	176000	.WORD	176000
3028	005526	174000	.WORD	174000
3029	005530	170000	.WORD	170000
3030	005532	160000	.WORD	160000
3031	005534	140000	.WORD	140000
3032	005536	100000	.WORD	100000
3033	005540	000000	.WORD	000000
3034				
3035	005542	000001	PAT7: .WORD	000001
3036	005544	000002	.WORD	000002
3037	005546	000004	.WORD	000004
3038	005550	000010	.WORD	000010
3039	005552	000020	.WORD	000020
3040	005554	000040	.WORD	000040
3041	005556	000100	.WORD	000100
3042	005560	000200	.WORD	000200
3043	005562	000400	.WORD	000400
3044	005564	001000	.WORD	001000
3045	005566	002000	.WORD	002000
3046	005570	004000	.WORD	004000
3047	005572	010000	.WORD	010000
3048	005574	020000	.WORD	020000
3049	005576	040000	.WORD	040000
3050	005600	100000	.WORD	100000
3051	005602	100000	.WORD	100000
3052	005604	040000	.WORD	040000
3053	005606	020000	.WORD	020000
3054	005610	010000	.WORD	010000
3055	005612	004000	.WORD	004000
3056	005614	002000	.WORD	002000
3057	005616	001000	.WORD	001000
3058	005620	000400	.WORD	000400
3059	005622	000200	.WORD	000200
3060	005624	000100	.WORD	000100
3061	005626	000040	.WORD	000040
3062	005630	000020	.WORD	000020
3063	005632	000010	.WORD	000010
3064	005634	000004	.WORD	000004

3065	005636	000002	.WORD	000002
3066	005640	000001	.WORD	000001
3067				
3068	005642	177776	PAT10: .WORD	177776
3069	005644	177775	.WORD	177775
3070	005646	177773	.WORD	177773
3071	005650	177767	.WORD	177767
3072	005652	177757	.WORD	177757
3073	005654	177737	.WORD	177737
3074	005656	177677	.WORD	177677
3075	005660	177577	.WORD	177577
3076	005662	177377	.WORD	177377
3077	005664	176777	.WORD	176777
3078	005666	175777	.WORD	175777
3079	005670	173777	.WORD	173777
3080	005672	167777	.WORD	167777
3081	005674	157777	.WORD	157777
3082	005676	137777	.WORD	137777
3083	005700	077777	.WORD	077777
3084	005702	077777	.WORD	077777
3085	005704	137777	.WORD	137777
3086	005706	157777	.WORD	157777
3087	005710	167777	.WORD	167777
3088	005712	173777	.WORD	173777
3089	005714	175777	.WORD	175777
3090	005716	176777	.WORD	176777
3091	005720	177377	.WORD	177377
3092	005722	177577	.WORD	177577
3093	005724	177677	.WORD	177677
3094	005726	177737	.WORD	177737
3095	005730	177757	.WORD	177757
3096	005732	177767	.WORD	177767
3097	005734	177773	.WORD	177773
3098	005736	177775	.WORD	177775
3099	005740	177776	.WORD	177776
3100				
3101	005742	172666	PAT11: .WORD	172666
3102	005744	155555	.WORD	155555
3103	005746	172666	.WORD	172666
3104	005750	155555	.WORD	155555
3105	005752	172666	.WORD	172666
3106	005754	155555	.WORD	155555
3107	005756	172666	.WORD	172666
3108	005760	155555	.WORD	155555
3109	005762	172666	.WORD	172666
3110	005764	155555	.WORD	155555
3111	005766	172666	.WORD	172666
3112	005770	155555	.WORD	155555
3113	005772	172666	.WORD	172666
3114	005774	155555	.WORD	155555
3115	005776	172666	.WORD	172666
3116	006000	155555	.WORD	155555
3117	006002	172666	.WORD	172666
3118	006004	155555	.WORD	155555
3119	006006	172666	.WORD	172666
3120	006010	155555	.WORD	155555

3121	006012	172666	.WORD	172666
3122	006014	155555	.WORD	155555
3123	006016	172666	.WORD	172666
3124	006020	155555	.WORD	155555
3125	006022	172666	.WORD	172666
3126	006024	155555	.WORD	155555
3127	006026	172666	.WORD	172666
3128	006030	155555	.WORD	155555
3129	006032	172666	.WORD	172666
3130	006034	155555	.WORD	155555
3131	006036	172666	.WORD	172666
3132	006040	155555	.WORD	155555
3133				
3134	006042	153333	PAT12: .WORD	153333
3135	006044	066667	.WORD	066667
3136	006046	153333	.WORD	153333
3137	006050	066667	.WORD	066667
3138	006052	153333	.WORD	153333
3139	006054	066667	.WORD	066667
3140	006056	153333	.WORD	153333
3141	006060	066667	.WORD	066667
3142	006062	153333	.WORD	153333
3143	006064	066667	.WORD	066667
3144	006066	153333	.WORD	153333
3145	006070	066667	.WORD	066667
3146	006072	153333	.WORD	153333
3147	006074	066667	.WORD	066667
3148	006076	153333	.WORD	153333
3149	006100	066667	.WORD	066667
3150	006102	153333	.WORD	153333
3151	006104	066667	.WORD	066667
3152	006106	153333	.WORD	153333
3153	006110	066667	.WORD	066667
3154	006112	153333	.WORD	153333
3155	006114	066667	.WORD	066667
3156	006116	153333	.WORD	153333
3157	006120	066667	.WORD	066667
3158	006122	153333	.WORD	153333
3159	006124	066667	.WORD	066667
3160	006126	153333	.WORD	153333
3161	006130	066667	.WORD	066667
3162	006132	153333	.WORD	153333
3163	006134	066667	.WORD	066667
3164	006136	153333	.WORD	153333
3165	006140	066667	.WORD	066667
3166				
3167	006142	000000	PAT13: .WORD	000000
3168	006144	177777	.WORD	177777
3169	006146	177777	.WORD	177777
3170	006150	177777	.WORD	177777
3171	006152	177777	.WORD	177777
3172	006154	177777	.WORD	177777
3173	006156	177777	.WORD	177777
3174	006160	177777	.WORD	177777
3175	006162	177777	.WORD	177777
3176	006164	177777	.WORD	177777

3177	006166	177777	.WORD	177777
3178	006170	177777	.WORD	177777
3179	006172	177777	.WORD	177777
3180	006174	177777	.WORD	177777
3181	006176	177777	.WORD	177777
3182	006200	177777	.WORD	177777
3183	006202	177777	.WORD	177777
3184	006204	177777	.WORD	177777
3185	006206	177777	.WORD	177777
3186	006210	177777	.WORD	177777
3187	006212	177777	.WORD	177777
3188	006214	177777	.WORD	177777
3189	006216	177777	.WORD	177777
3190	006220	177777	.WORD	177777
3191	006222	177777	.WORD	177777
3192	006224	177777	.WORD	177777
3193	006226	177777	.WORD	177777
3194	006230	177777	.WORD	177777
3195	006232	177777	.WORD	177777
3196	006234	177777	.WORD	177777
3197	006236	177777	.WORD	177777
3198	006240	177777	.WORD	177777
3199				
3200	006242	177777		
3201	006244	000000	PAT14: .WORD	177777
3202	006246	000000	.WORD	000000
3203	006250	000000	.WORD	000000
3204	006252	000000	.WORD	000000
3205	006254	000000	.WORD	000000
3206	006256	000000	.WORD	000000
3207	006260	000000	.WORD	000000
3208	006262	000000	.WORD	000000
3209	006264	000000	.WORD	000000
3210	006266	000000	.WORD	000000
3211	006270	000000	.WORD	000000
3212	006272	000000	.WORD	000000
3213	006274	000000	.WORD	000000
3214	006276	000000	.WORD	000000
3215	006300	000000	.WORD	000000
3216	006302	000000	.WORD	000000
3217	006304	000000	.WORD	000000
3218	006306	000000	.WORD	000000
3219	006310	000000	.WORD	000000
3220	006312	000000	.WORD	000000
3221	006314	000000	.WORD	000000
3222	006316	000000	.WORD	000000
3223	006320	000000	.WORD	000000
3224	006322	000000	.WORD	000000
3225	006324	000000	.WORD	000000
3226	006326	000000	.WORD	000000
3227	006330	000000	.WORD	000000
3228	006332	000000	.WORD	000000
3229	006334	000000	.WORD	000000
3230	006336	000000	.WORD	000000
3231	006340	000000	.WORD	000000
3232				

3233	006342	172304	PAT15: .WORD	172304
3234	006344	172304	.WORD	172304
3235	006346	172304	.WORD	172304
3236	006350	172304	.WORD	172304
3237	006352	172304	.WORD	172304
3238	006354	172304	.WORD	172304
3239	006356	172304	.WORD	172304
3240	006360	172304	.WORD	172304
3241	006362	172304	.WORD	172304
3242	006364	172304	.WORD	172304
3243	006366	172304	.WORD	172304
3244	006370	172304	.WORD	172304
3245	006372	172304	.WORD	172304
3246	006374	172304	.WORD	172304
3247	006376	172304	.WORD	172304
3248	006400	172304	.WORD	172304
3249	006402	172304	.WORD	172304
3250	006404	172304	.WORD	172304
3251	006406	172304	.WORD	172304
3252	006410	172304	.WORD	172304
3253	006412	172304	.WORD	172304
3254	006414	172304	.WORD	172304
3255	006416	172304	.WORD	172304
3256	006420	172304	.WORD	172304
3257	006422	172304	.WORD	172304
3258	006424	172304	.WORD	172304
3259	006426	172304	.WORD	172304
3260	006430	172304	.WORD	172304
3261	006432	172304	.WORD	172304
3262	006434	172304	.WORD	172304
3263	006436	172304	.WORD	172304
3264	006440	172304	.WORD	172304
3265				
3266	006442	070627	PAT16: .WORD	070627
3267	006444	113431	.WORD	113431
3268	006446	014561	.WORD	014561
3269	006450	070627	.WORD	070627
3270	006452	113431	.WORD	113431
3271	006454	014561	.WORD	014561
3272	006456	070627	.WORD	070627
3273	006460	113431	.WORD	113431
3274	006462	014561	.WORD	014561
3275	006464	070627	.WORD	070627
3276	006466	113431	.WORD	113431
3277	006470	014561	.WORD	014561
3278	006472	070627	.WORD	070627
3279	006474	113431	.WORD	113431
3280	006476	014561	.WORD	014561
3281	006500	070627	.WORD	070627
3282	006502	113431	.WORD	113431
3283	006504	014561	.WORD	014561
3284	006506	070627	.WORD	070627
3285	006510	113431	.WORD	113431
3286	006512	014561	.WORD	014561
3287	006514	070627	.WORD	070627
3288	006516	113431	.WORD	113431

3289	006520	014561	.WORD	014561
3290	006522	070627	.WORD	070627
3291	006524	113431	.WORD	113431
3292	006526	014561	.WORD	014561
3293	006530	070627	.WORD	070627
3294	006532	113431	.WORD	113431
3295	006534	014561	.WORD	014561
3296	006536	070627	.WORD	070627
3297	006540	113431	.WORD	113431
3298				
3299	006542	133467	PAT17: .WORD	133467
3300	006544	133467	.WORD	133467
3301	006546	133467	.WORD	133467
3302	006550	133467	.WORD	133467
3303	006552	133467	.WORD	133467
3304	006554	133467	.WORD	133467
3305	006556	133467	.WORD	133467
3306	006560	133467	.WORD	133467
3307	006562	133467	.WORD	133467
3308	006564	133467	.WORD	133467
3309	006566	133467	.WORD	133467
3310	006570	133467	.WORD	133467
3311	006572	133467	.WORD	133467
3312	006574	133467	.WORD	133467
3313	006576	133467	.WORD	133467
3314	006600	133467	.WORD	133467
3315	006602	133467	.WORD	133467
3316	006604	133467	.WORD	133467
3317	006606	133467	.WORD	133467
3318	006610	133467	.WORD	133467
3319	006612	133467	.WORD	133467
3320	006614	133467	.WORD	133467
3321	006616	133467	.WORD	133467
3322	006620	133467	.WORD	133467
3323	006622	133467	.WORD	133467
3324	006624	133467	.WORD	133467
3325	006626	133467	.WORD	133467
3326	006630	133467	.WORD	133467
3327	006632	133467	.WORD	133467
3328	006634	133467	.WORD	133467
3329	006636	133467	.WORD	133467
3330	006640	133467	.WORD	133467

```

3331      .SBTTL PROGRAM SETUP
3332
3333 006642 112737 000001 001361 PARSRT: MOVB #1,FLAG ;SET FLAG FOR PARAMETER ENTRY
3334 006650 000412                BR      PRGSRT ;START PROGRAM
3335
3336 006652 112737 177777 001361 RESTR7: MOVB #-1,FLAG ;SET FLAG FOR RETRY
3337 006660 000406                BR      PRGSRT ;START PROGRAM
3338
3339 006662 105037 001361          START1: CLRB FLAG ;SET FOR ENTRY INPUT START
3340 006666 000403                BR      PRGSRT ;START PROGRAM
3341
3342 006670 112737 000002 001361 START:  MOVB #2,FLAG ;SET FOR AUTO SIZE PGM SIZE
3343
3344 006676 012737 000340 177776 PRGSRT: MOV #PR7,PS ;LOCK OUT ALL INTERRUPTS
3345      .SBTTL INITIALIZE THE COMMON TAGS
3346      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
3347 006704 012706 001100          MOV #SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
3348 006710 005026                CLR (R6)+ ;CLEAR MEMORY LOCATION
3349 006712 022706 001140          CMP #SWR,R6 ;;DONE?
3350 006716 001374                BNE -6 ;LOOP BACK IF NO
3351 006720 012706 001100          MOV #STACK,SP ;SETUP THE STACK POINTER
3352      ;;INITIALIZE A FEW VECTORS
3353 006724 012737 055664 000034 MOV #STRAP,@TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
3354 006732 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
3355 006740 012737 054552 000024 MOV #SPWRON,@PWRVEC ;POWER FAILURE VECTOR
3356 006746 012737 000340 000026 MOV #340,@PWRVEC+2 ;LEVEL 7
3357      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3358      ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
3359 006754 013746 000004          MOV @ERRVEC-(SP) ;SAVE ERROR VECTOR
3360 006760 012737 007014 000004 MOV #64$,@ERRVEC ;SET UP ERROR VECTOR
3361 006766 012737 177570 001140 MOV #DSWR,SWR ;SETUP FOR A HARDWARE SWICH REGISTER
3362 006774 012737 177570 001142 MOV #DDISP,DISPLAY ;AND A HARDWARE DISPLAY REGISTER
3363 007002 022777 177777 172130 CMP #-1,@SWR ;TRY TO REFERENCE HARDWARE SWR
3364 007010 001012                BNE 66$ ;BRANCH IF NO TIMEOUT TRAP OCCURRED
3365 ;AND THE HARDWARE SWR IS NOT = -1
3366 007012 000403                BR 65$ ;BRANCH IF NO TIMEOUT
3367 007014 012716 007022          64$: MOV #65$,(SP) ;SET UP FOR TRAP RETURN
3368 007020 000002                RTI
3369 007022 012737 000176 001140 65$: MOV #SWREG,SWR ;POINT TO SOFTWARE SWR
3370 007030 012737 000174 001142 MOV #DISPREG,DISPLAY
3371 007036 012637 000004          66$: MOV (SP)+,@ERRVEC ;RESTORE ERROR VECTOR
3372
3373 007042 005037 001176          CLR $PASS ;CLEAR PASS COUNT
3374 007046 132737 000200 001211 BITB #APTSIZE,$ENVM ;TEST USER SIZE UNDER APT
3375 007054 001403                BEQ 67$ ;YES,USE NON-APT SWITCH
3376 007056 012737 001212 001140 MOV #SSWREG,SWR ;NO,USE APT SWITCH REGISTER
3377 007064          67$:
3378      .SBTTL TYPE PROGRAM NAME
3379      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
3380 007064 005227 177777          INC #-1 ;FIRST TIME?
3381 007070 001040                BNE 68$ ;BRANCH IF NO
3382 007072 104401 007100          TYPE 69$ ;TYPE ASCIZ STRING
3383 007076 000435                BR 68$ ;GET OVER THE ASCIZ
3384 ;69$: .ASCIZ <CRLF>"RK611/RK06-RK07 PERFORMANCE EXERCISER: MAINDEC DZR6P-C"<CRLF>
3385 ;68$:
3386 007172 000005          PWSRRT: RESET ;RESET SYSTEM

```

3387	007174	012737	007246	000004		MOV	#20\$,ERRVEC	;SET VECTOR FOR MEMORY PARITY CHECK
3388	007202	012737	000340	000006		MOV	#PR7,ERRVEC+2	
3389	007210	012703	172100			MOV	#MEMBAS,R3	;LOAD REGISTER TO DTERMIN IF
3390								;MEMORY CHECK ENABLE ABAILIABLE
3391	007214	012704	000014			MOV	#12,R4	;LOAD COUNT
3392	007220	012723	000001		16\$:	MOV	#PAR,EN,(R3)+	;ENABLE MEMORY CHECK VECTORY
3393	007224	012737	052472	000114		MOV	#MEMERR,MEMVEC	;LOAD MEMORY CHECK VECTOR
3394	007232	012737	000340	000116		MOV	#PR7,MEMVEC+2	
3395	007240	005304				DEC	R4	;CHECK IF FINISHED
3396	007242	001366				BNE	16\$;NO, SET UP NEXT MEMORY PARITY MODULE
3397	007244	000401				BR	22\$;RESTORE TRAP VECTOR
3398								
3399	007246	022626			20\$:	CMP	(SP)+,(SP)+	;ADJUST STACK
3400	007250	012737	000006	000004	22\$:	MOV	#ERRVEC+2,ERRVEC	;RESTORE TRAP CATCHER
3401	007256	005037	000006			CLR	ERRVEC+2	
3402	007262	004737	054454		START\$:	JSR	PC,\$SIZE	;GET MAXIMUM MEMORY ADDRESS
3403	007266	012737	000001	001670		MOV	#1,FBLKC	;I ITIALIZE FREE BLOCK COUNT
3404	007274	012703	001672			MOV	#FBLKT,R3	;STORE FREE BLOCK TABLE ADDRESS
3405	007300	012704	000022			MOV	#18,R4	;LOAD COUNT FOR FREE BLOCK CLEAR
3406	007304	005023			2\$:	CLR	(R3)+	;CLEAR FREE BLOCK TABLE
3407	007306	005304				DEC	R4	;DECREMENT COUNT
3408	007310	001375				BNE	2\$;CHECK IF FINISHED
3409	007312	012737	066212	001672		MOV	#BUFADD,FBLKT	;LOAD FIRST ENTRY IN FREE BLOCK TABLE
3410	007320	105037	001644			CLRB	RTYSCN	;CLEAR RETRY SECTOR COUNT
3411	007324	005037	001566			CLR	ERRPRO	;CLEAR ADDRESS OF PARAMETER BLOCK
3412								;PROCESSING ERROR
3413	007330	005037	001462			CLR	E.CONT	;CLEAR CONTROLLER ERROR FLAGS
3414	007334	005037	001564			CLR	ERCONT	;CLEAR RECOVERABLE CONTROLLER ERROR FLAGS
3415	007340	005037	001626			CLR	DLTCNT	;CLEAR DATA LATE COUNT
3416	007344	005037	001630			CLR	CNTCNT	;CLEAR OTHER ERROR COUNT
3417	007350	105037	001633			CLRB	STATIS	;CLEAR INTERVAL STATISTICS FLAG
3418	007354	112737	177777	001513		MOVB	#-1,0.OVER	;SET ALL DRIVES FOR IMPLIED SEEKS
3419	007362	012703	001314			MOV	#AVAILQ,R3	;STORE QUEUE BASE ADDRESS
3420	007366	012704	000010			MOV	#8,R4	;LOAD QUEUE COUNT
3421	007372	005023			3\$:	CLR	(R3)+	;CLEAR QUEUE HEADS AND TAILS
3422	007374	005304				DEC	R4	;DECREMENT COUNT
3423	007376	001375				BNE	3\$;CHECK IF FINISHED
3424	007400	012704	001544			MOV	#W.DRV,R4	;LOAD ADDRESS OF TIME OUT COUNTS
3425	007404	012703	000010			MOV	#8,R3	;LOAD COUNT
3426	007410	005024			4\$:	CLR	(R4)+	;CLEAR DRIVE TIME OUT COUNTS
3427	007412	005303				DEC	R3	;DECREMENT COUNT
3428	007414	001375				BNE	4\$;CHECK IF FINISHED
3429	007416	004737	043372			JSR	PC,CHKCLK	;CHECK IF CLOCK IS PRESENT
3430	007422	105037	001512			CLRB	W.TIME	;CLEAR DRIVES BEING TIMED
3431	007426	013737	001470	001466		MOV	W.MILI,W.MTIM	;INITIALIZE MILI-SECOND TIMER
3432	007434	005037	001464			CLR	0.WAIT	;CLEAR PARAMETER BLOCK FOR DRIVE
3433								;WAITING COMMAND COMPLETION
3434	007440	105037	001506			CLRB	I.ISRL	;RESET RELEASE OR CONTROLLER CLEAR ISSUED
3435	007444	105737	001361			TSTB	FLAG	;CHECK IF RESTART
3436	007450	100072				BPL	RESTO	;NO, CONTINUE
3437								
3438	007452	013737	054550	001674		MOV	\$LSTAD,FBLKT+2	;LOAD MAX. ADD IF FREE BLOCK TABLE
3439	007460	105737	001365			TSTB	OVLYLD	;CHECK IF OVERLAY LOADER
3440	007464	001012				BNE	20\$;YES, LOAD RK06 VECTOR ADDRESS
3441	007466	105737	000041			TSTB	41	;SEE WHO LOADED THE PROBLEM
3442	007472	001004				BNE	10\$;BRANCH IF XXDP

```

3443 007474 162737 000300 001674      SUB      #96.*2.,FBLKT+2 ;SUBTRACT ABS LOADER SIZE
3444 007502 000403                BP      20$          ;LOAD VECTOR ADDRESS
3445
3446 007504 162737 004004 001674 10$:      SUB      #1026.*2.,FBLKT+2 ;SUBTRACT XXDP LOADER
3447 007512 013704 001450 20$:      MOV      RKVEC,R4      ;STORE VECTOR ADDRESS
3448 007516 012724 044266      MOV      #I.INTR,(R4)+ ;LOAD RK06 VECTOR ADDRESS
3449 007522 013714 001452      MOV      RKPRI,(R4)    ; AND INTERRUPT PRIORITY
3450 007526 012703 001524      MOV      #PBLKT,R3    ;LOAD PARAMETER BLOCK TABLE ADDRESS
3451 007532 012704 000011      MOV      #9.,R4       ;LOAD COUNT
3452
3453 007536 005304                21$:      DEC      R4           ;DECREMENT COUNT AND CHECK IF DONE
3454 007540 001434                BEQ      30$          ;YES, SET UP TTY INTERRUPT
3455 007542 012305                MOV      (R3)+,R5     ;GET PARAMETER BLOCK ADDRESS
3456 007544 105065 000145      CLRB    P.BUFF(R5)    ;CLEAR BUFFER ALLOCATED
3457 007550 005065 000212      CLR     P.ERR(R5)     ;CLEAR ERROR FLAGS AND RETRY COUNTS
3458 007554 005065 000150      CLR     P.DSTT(R5)
3459 007560 005065 000146      CLR     P.RECT(R5)
3460
3461
3462      ;
3463      ;
3464      ;
3465      ;
3466      ;
3467      ;
3468 007564 042765 070006 000014      BIC     #DRVPOS!DRVPDT!DRVSZD!E.UNLD!Q.INIT,P.PRST(R5)
3469
3470 007572 032765 000001 000014      BIT     #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
3471 0076J0 001756                BEQ     21$          ;NO, GET NEXT PARAMETER BLOCK
3472 007602 105765 000210      TSTB   P.ASSN(R5)    ;CHECK IF DRIVE IS BEING ASSIGNED
3473 007606 001003                BNE     22$          ;YES, RESTART ASSIGNMENT SEQUENCE
3474 007610 004737 013434      JSR     PC.TEST1     ;ISSUE A START SPINDLE IF NECESSARY
3475 007614 000750                BR      21$          ;GET NEXT PARAMETER BLOCK
3476
3477 007616 142765 177757 000210 22$:      BICB   #1C<BIT4>,P.ASSN(R5) ;KEEP WRITE PACK BIT
3478 007624 004737 013360      JSR     PC.TEST      ;INITIATE DRIVE ASSIGNMENT SEQUENCE
3479 007630 000742                BR      21$          ;GET NEXT PARAMETER BLOCK
3480
3481 007632 000137 011F 2      30$:      JMP     REST2        ;SET TTY INTERRUPT
3482
3483 007636 012737 176543 055126 RESTD:  MOV     #176543,$HINUM ;INITIALIZE RANDOM NUMBER GENERATOR
3484 007644 012737 23456 055130      MOV     #123456,$LONUM
3485 007652 005037 001372                CLR     SECOND      INITIALIZE INTERNAL CLOCK
3486 007656 005037 001370                CLR     MINUTE
3487 007662 005037 001366                CLR     HOUR
3488 007666 012703 001524      MOV     #PBLKT,R3    ;LOAD PARAMETER BLOCK TABLE ADDRESS
3489 007672 012704 000010      MOV     #8.,R4       ;LOAD COUNT
3490 007676 012305                4$:      MOV     (R3)+,R5     ;GET PARAMETER BLOCK ADDRESS
3491 007700 005065 000014      CLR     P.PRST(R5)   ;CLEAR RK06 PROGRAM DEVICE STATUS REG.
3492 007704 005065 000212      CLR     P.ERR(R5)    ;CLEAR DRIVE ERROR FLAGS
3493 007710 105065 000210      CLRB   P.ASSN(R5)    ;CLEAR DRIVE ASSIGNMENT CODE
3494 007714 010500                MOV     R5,R0        ;STORE R0 FOR TRACK EXCLUSIONS
3495 007716 062700 000232      ADD     #P.EXAR,R0   ;DETERMINE TRACK EXCLUSION BLOCK
3496 007722 012701 000012      MOV     #10.,R1     ;LOAD COUNT FOR CLEARING TRACK EXCLUSION
3497      ;
3498 007726 005020                5$:      CLR     (R0)+       ;CLEAR TRACK EXCLUSION ENTRY

```

3499	007730	005301				DEC R1	: DECREMENT COUNT
3500	007732	001375				BNE 55	: CHECK IF WHOLE BLOCK CLEARED
3501	007734	005304				DEC R4	: DECREMENT COUNT
3502	007736	001357				BNE 45	: TEST IF DONE
3503	007740	122737	000001	001361		CMPB #1,FLAG	: CHECK IF PARAMETER START
3504	007746	001402				BEQ 105	: YES, ASK FOR PARAMETERS
3505	007750	000137	011102			JMP 515	: USE DEFAULT VALUES
3506							
3507	007754	104401	057161		105:	TYPE SYS000	: TYPE "CPU ID"
3508	007760	004037	020340			JSR R0,TSTDEF	: CHECK FOR DEFAULT PARAMETERS
3509	007764	010014				115	: COMMA DETECTED
3510	007766	010032				135	: CARRIAGE RETURN DETECTED
3511	007770	011102				515	: CONTROL Z (<z>) DETECTED
3512	007772	007754				105	: CONTROL C (<c>) DETECTED
3513	007774	010346				MOV R3,-(SP)	: LOAD BUFFER ADDRESS ON STACK
3514	007776	004737	054146			JSR PC,OCTBIN	: CONVERT TO BINARY
3515	010002	010014				115	: ERROR RETURN
3516	010004	012604				MOV (SP)+,R4	: STORE INPUT VALUE
3517	010006	022704	000400			CMP #400,R4	: CHECK IF 0-377
3518	010012	101010				BHI 145	: YES, LOAD CPU ID
3519	010014				115:		
3520	010014	010337	010022			MOV R3,645	: LOAD BUFFER ADDRESS
3521	010020	104401				TYPE	: TYPE RECEIVED INPUT
3522	010022	000000			645:	.WORD 0	: BUFFER ADDRESS
3523	010024	104401	001164			TYPE \$QUES	: TYPE QUESTION MARK
3524	010030	000751				BR 105	: TRY AGAIN
3525							
3526	010032	005004			135:	CLR R4	: CLEAR CPU ID (R4)
3527	010034	110437	001364		145:	MOV R4,CPUID	: LOAD CPU ID
3528							
3529	010040	013737	054550	001674	155:	MOV \$LSTAD,FBLKT+2	: LOAD LAST ADDRESS IN FIRST FREE BLOCK ENTRY
3530							
3531	010046	104401	057171			TYPE SYS001	: TYPE "OVERLAY LOADER"
3532	010052	004037	020340			JSR R0,TSTDEF	: CHECK FOR DEFAULT PARAMETERS
3533	010056	010126				185	: COMMA DETECTED
3534	010060	010074				165	: CARRIAGE RETURN DETECTED
3535	010062	011114				S25	: CONTROL Z (<z>) DETECTED
3536	010064	007754				105	: CONTROL C (<c>) DETECTED
3537	010066	122713	000131			CMPB #'Y,(R3)	: CHECK IF YES
3538	010072	001424				BEQ 195	: YES, INDICATE THAT LOADER HAS BEEN OVERLAYED
3539	010074	105037	001365		165:	CLRB OVLYLD	: INDICATE THAT LOADER HAS NOT BEEN OVERLAYED
3540	010100	105737	000041			TSTB 41	: SEE WHO LOADED THE PROGRAM
3541	010104	001004				BNE 175	: BRANCH IF XXDP
3542	010106	162737	000300	001674		SUB #96.*2.,FBLKT+2	: SUBTRACT ABS LOADER SIZE
3543	010114	000416				BR 205	: ASK FOR MAXIMUM TRANSFER SIZE
3544							
3545	010116	162737	006000	001674	175:	SUB #1536.*2.,FBLKT+2	: SUBTRACT XXDP LOADER SIZE
3546	010124	000412				BR 205	: ASK FOR TRANSFER SIZE
3547							
3548	010126				185:		
3549	010126	010337	010134			MOV R3,655	: LOAD BUFFER ADDRESS
3550	010132	104401				TYPE	: TYPE RECEIVED INPUT
3551	010134	000000			655:	.WORD 0	: BUFFER ADDRESS
3552	010136	104401	001164			TYPE \$QUES	: TYPE QUESTION MARK
3553	010142	000736				BR 155	: TRY AGAIN
3554							

Address	Op	Op2	Op3	Op4	Label	Code	Comment
3555	010144	112737	177777	001365	19\$:	MOV	#-1,OVLYLD ;INDICATE THAT LOADER HAS BEEN OVERLAYED
3556							
3557	010152	104401	057211		20\$:	TYPE	SYS002 ;TYPE "MAXIMUM TRANSFER"
3558	010156	013746	001674			MOV	FBLKT+2,-(SP) ;STORE LAST USABLE MEMORY LOCATION
3559	010162	163716	001672			SUB	FBLKT,(SP) ;CALCULATE MEMORY LEFT
3560	010166	022716	027000			CMP	#256.*23.*2.,(SP) ;CHECK IF GREATER THAN OF EQUAL TO 5888
3561	010172	101402				BLOS	21\$;YES, USE 5888
3562	010174	006216				ASR	(SP) ;DETERMINE NUMBER OF WORDS
3563	010176	000402				BR	22\$;TYPE VALUE
3564							
3565	010200	012716	013400		21\$:	MOV	#256.*23.,(SP) ;LOAD 5888
3566							
3567	010204	012637	001570		22\$:	MOV	(SP)+,T MPCNT ;SAVE TRANSFER COUNT
3568	010210	013746	001570			MOV	T MPCNT -(SP) ;SAVE TRANSFER COUNT FOR TYPE OUT
3569	010214	004737	052774			JSR	PC,BIN OCT ;CONVERT TO OCTAL
3570	010220	104401	057243			TYPE	S'5003
3571	010224	004037	020340			JSR	RD,TSTDEF ;CHECK FOR DEFAULT PARAMETERS
3572	010230	010272				24\$;COMMA DETECTED
3573	010232	010310				26\$;CARRIAGE RETURN DETECTED
3574	010234	011144				53\$;CONTROL Z (<Z>) DETECTED
3575	010236	007754				10\$;CONTROL C (<C>) DETECTED
3576	010240	010346				MOV	R3,-(SP) ;STORE BUFFER ADDRESS ON STACK
3577	010242	004737	054146			JSR	PC,OCTBIN ;CONVERT TO BINARY
3578	010246	010270				23\$;ERROR RETURN
3579	010250	012604				MOV	(SP)+,R4 ;STORE INPUT
3580	010252	001407				BEQ	24\$;CHECK IF ZERO
3581	010254	020437	001570			CMP	R4,T MPCNT ;CHECK IF LEGAL ENTRY
3582	010260	101004				BHI	24\$;NO, TRY AGAIN
3583	010262	010437	001354			MOV	R4,MAXBUF ;LOAD MAXIMUM TRANSFER
3584	010266	000422				BR	27\$;GET NO. OF SOFT COMPARES
3585							
3586	010270	005724			23\$:	TST	(SP)+ ;ADJUST STACK
3587	010272				24\$:		
3588	010272	010337	010300			MOV	R3,66\$;LOAD BUFFER ADDRESS
3589	010276	104401				TYPE	;TYPE RECEIVED INPUT
3590	010300	000000			66\$:	.WORD	0 ;BUFFER ADDRESS
3591	010302	104401	001164			TYPE	\$QUES ;TYPE QUESTION MARK
3592	010306	000721				BR	20\$;TRY AGAIN
3593							
3594	010310	013737	001570	001354	26\$:	MOV	T MPCNT,MAXBUF ;LOAD MAXIMUM TRANSFER
3595	010316	022737	013400	001354		CMP	#256.*23.,MAXBUF ;CHECK IF LESS THAN OR EQUAL TO 23 SECTORS
3596	010324	103003				BHIS	27\$;YES, USE CALCULATED VALUE
3597	010326	012737	013400	001354		MOV	#256.*23.,MAXBUF ;USE 23 SECTORS
3598							
3599	010334	104401	057247		27\$:	TYPE	SYS004 ;TYPE "NO. OF SOFTWARE COMPARISONS"
3600	010340	004037	020340			JSR	RD,TSTDEF ;CHECK FOR DEFAULT PARAMETERS
3601	010344	010402				29\$;COMMA DETECTED
3602	010346	010420				30\$;CARRIAGE RETURN DETECTED
3603	010350	011202				54\$;CONTROL Z (<Z>) DETECTED
3604	010352	007754				10\$;CONTROL C (<C>) DETECTED
3605	010354	010346				MOV	R3,-(SP) ;STORE PARAMETER
3606	010356	004737	054302			JSR	PC,DECBIN ;CONVERT TO BINARY
3607	010362	010402				29\$;ERROR RETURN
3608	010364	023716	001354			CMP	MAXBUF,(SP) ;CHECK IF LEGAL INPUT
3609	010370	103403				BLO	28\$;TYPE VALUE AND RETRY
3610	010372	012637	001362			MOV	(SP)+,SOFCMP ;LOAD NUMBER OF SOFTWARE COMPARES

3611	010376	000413			BR	31\$; DETERMINE RK06 UNIBUS ADDRESS
3612								
3613	010400	005726			28\$: TST	(SP)+		; ADJUST STACK
3614	010402				29\$:			
3615	010402	010337	010410		MOV	R3,67\$; LOAD BUFFER ADDRESS
3616	010406	104401			TYPE			; TYPE RECEIVED INPUT
3617	010410	000000			67\$: .WORD	0		; BUFFER ADDRESS
3618	010412	104401	001164		TYPE	\$QUES		; TYPE QUESTION MARK
3619	010416	000746			BR	27\$; TRY AGAIN
3620								
3621	010420	012737	000003	001362	30\$: MOV	#3,S0FCMP		; LOAD DEFAULT VALUE
3622								
3623	010426	105737	001374		31\$: TSTB	CLKFLG		; CHECK IF CLOCK ON SYSTEM
3624	010432	001434			BEQ	41\$; NO CONTINUE
3625	010434	104401	057442		TYPE	SYS009		; TYPE "STATISTIC INTERVAL="
3626	010440	004037	020340		JSR	R0,TSTDEF		; CHECK FOR DEFAULT PARAMETERS
3627	010444	010502			34\$; COMMA DETECTED
3628	010446	010520			35\$; CARRIAGE RETURN DETECTED
3629	010450	011210			55\$; CONTROL Z (<↑Z>) DETECTED
3630	010452	007754			10\$; CONTROL C (<↑C>) DETECTED
3631	010454	010346			MOV	R3,-(SP)		; STORE NUMBER FOR CONVERSION
3632	010456	004737	054302		JSR	PC,DECBIN		; CONVERT TO BINARY
3633	010462	010502			34\$; ILLEGAL NUMBER
3634	010464	012604			MOV	(SP)+,R4		; STORE NUMBER
3635	010466	022704	000377		CMP	#255.,R4		; CHECK IF GREATER THAN 255
3636	010472	103403			BLO	34\$; NO TRY AGAIN
3637	010474	110437	001360		MOV	R4,PERINV		; LOAD PERFORMANCE INTERVAL
3638	010500	000411			BR	41\$; CONTINUE
3639								
3640	010502				34\$:			
3641	010502	010337	010510		MOV	R3,68\$; LOAD BUFFER ADDRESS
3642	010506	104401			TYPE			; TYPE RECEIVED INPUT
3643	010510	000000			68\$: .WORD	0		; BUFFER ADDRESS
3644	010512	104401	001164		TYPE	\$QUES		; TYPE QUESTION MARK
3645	010516	000743			BR	31\$; TRY AGAIN
3646								
3647	010520	105037	001360		35\$: CLRB	PERINV		; SET UP FOR NO INTERVAL STATISTICS
3648								
3649	010524	104401	057301		41\$: TYPE	SYS005		; TYPE "RK06 BUS ADD."
3650	010530	004037	020340		JSR	R0,TSTDEF		; CHECK FOR DEFAULT PARAMETERS
3651	010534	010650			44\$; COMMA DETECTED
3652	010536	010666			45\$; CARRIAGE RETURN DETECTED
3653	010540	011214			56\$; CONTROL Z (<↑Z>) DETECTED
3654	010542	007754			10\$; CONTROL C (<↑C>) DETECTED
3655	010544	010346			MOV	R3,-(SP)		; STORE PARAMETER
3656	010546	004737	054146		JSR	PC,OCTBIN		; CONVERT TO BINARY
3657	010552	010650			44\$; ERROR RETURN
3658	010554	012604			MOV	(SP)+,R4		; STORE INPUT
3659	010556	022704	160000		CMP	#160000,R4		; CHECK IF I/O PAGE
3660	010562	101032			BHI	44\$; NO TRY AGAIN
3661	010564	012737	010624	000004	MOV	#42\$,ERRVEC		; LOAD ERRVEC TO NON-EXISTENT MEMORY
3662	010572	013737	000340	000006	MOV	PR7,ERRVEC+2		; PRIORITY 7
3663	010600	005714			TST	(R4)		; CHECK FOR NON-EXISTENT MEMORY
3664	010602	012737	000000	000006	MOV	#(HALT),ERRVEC+2		; REINSTATE TRAP CATCHER
3665	010610	012737	000006	000004	MOV	#ERRVEC+2,ERRVEC		
3666	010616	010437	001446		MOV	R4,RKBAS		; LEGAL I/O ADDRESS, LOAD RK06 BASE

3667	010622	000424				BR	48%		;GET VECTOR ADDRESS
3668									
3669	010624	012737	000000	000006	42%	MOV	#(HALT),ERRVEC+2		;REINSTATE TRAP CATCHER
3670	010632	012737	000006	000004		MOV	#ERRVEC+2,ERRVEC		
3671	010640	062706	000004			ADD	#4,SP		;ADJUST STACK
3672	010644	104401	057347			TYPE	,SYS007		;TYPE "NON-EXISTENT MEMORY"
3673	010650				44%				
3674	010650	010337	010656			MOV	R3,69%		;LOAD BUFFER ADDRESS
3675	010654	104401				TYPE			;TYPE RECEIVED INPUT
3676	010656	000000			69%	.WORD	0		;BUFFER ADDRESS
3677	010660	104401	001164			TYPE	\$QUES		;TYPE QUESTION MARK
3678	010664	000717				BR	41%		;TRY AGAIN
3679									
3680	010666	012737	177440	001446	45%	MOV	#177440,RKBAS		;LOAD DEFAULT BUS ADDRESS
3681									
3682	010674	104401	057324		48%	TYPE	,SYS006		;TYPE "RK06 VEC ADD"
3683	010700	004037	020340			JSR	RO,TSTDEF		;CHECK FOR DEFAULT PARAMETERS
3684	010704	010742				50%			;COMMA DETECTED
3685	010706	010760				51%			;CARRIAGE RETURN DETECTED
3686	010710	011222				57%			;CONTROL Z (<↑Z>) DETECTED
3687	010712	007754				10%			;CONTROL C (<↑C>) DETECTED
3688	010714	010346				MOV	R3,-(SP)		;STORE PARAMETER
3689	010716	004737	054146			JSR	PC,OCTBIN		;CONVERT TO BINARY
3690	010722	010742				50%			;ERROR RETURN
3691	010724	022716	001000			CMP	#1000,(SP)		;CHECK IF VECTOR SPACE
3692	010730	101403				BLOS	49%		;NO, TRY AGAIN
3693	010732	012637	001450			MOV	(SP)+,RKVEC		;LOAD VECTOR ADDRESS
3694	010736	000413				BR	53%		;GET PRIORITY
3695									
3696	010740	005726			49%	TST	(SP)+		;ADJUST STACK
3697	010742				50%				
3698	010742	010337	010750			MO'	R3,70%		;LOAD BUFFER ADDRESS
3699	010746	104401				TYPE			;TYPE RECEIVED INPUT
3700	010750	000000			70%	.WORD	0		;BUFFER ADDRESS
3701	010752	104401	001164			TYPE	\$QUES		;TYPE QUESTION MARK
3702	010756	000746				BR	48%		;TRY AGAIN
3703									
3704	010760	012737	000210	001450	51%	MOV	#210,RKVEC		;LOAD DEFAULT RK06 VECTOR
3705	010766	104401	057466		53%	TYPE	,SYS010		;TYPE "RK06 PRIOR="
3706	010772	004037	020340			JSR	RO,TSTDEF		;CHECK FOR DEFAULT PARAMETERS
3707	010776	011054				56%			;COMMA DETECTED
3708	011000	011072				57%			;CARRIAGE RETURN DETECTED
3709	011002	011230				58%			;CONTROL Z (<↑Z>) DETECTED
3710	011004	007754				10%			;CONTROL C (<↑C>) DETECTED
3711	011006	010346				MOV	R3,-(SP)		;STORE PARAMETER
3712	011010	004737	054146			JSR	PC,OCTBIN		;CONVERT TO BINARY
3713	011014	011054				56%			;ERROR RETURN
3714	011016	022716	000007			CMP	#7,(SP)		;CHECK IF OCTAL
3715	011022	103413				BLO	55%		
3716	011024	022716	000004			CMP	#4,(SP)		
3717	011030	101010				BHI	55%		
3718	011032	006316				ASL	(SP)		;SHIFT 5 BITS LEFT
3719	011034	006316				ASL	(SP)		
3720	011036	006316				ASL	(SP)		
3721	011040	006316				ASL	(SP)		
3722	011042	006316				ASL	(SP)		

3723	011044	012637	001452		MOV	(SP)+,RKPRI	;LOAD PRIORITY
3724	011050	000472			BR	REST1	;LOAD RK06 VECTOR AND TTY VECTOR
3725							
3726	011052	005726			55\$: TST	(SP)+	;ADJUST STACK
3727	011054				56\$:		
3728	011054	010337	011062		MOV	R3,71\$;LOAD BUFFER ADDRESS
3729	011060	104401			TYPE		;TYPE RECEIVED INPUT
3730	011062	000000			71\$: .WORD	0	;BUFFER ADDRESS
3731	011064	104401	001164		TYPE	\$QUES	;TYPE QUESTION MARK
3732	011070	000736			BR	53\$;TRY AGAIN
3733							
3734	011072	012737	000240	001452	57\$: MOV	#PRS,RKPRI	;LOAD PRIORITY
3735	011100	000456			BR	REST1	;LOAD RK06 VECTOR AND TTY VECTOR
3736							
3737	011102	105037	001364		51\$: CLRB	CPUID	;LOAD DEFAULT CPU ID
3738	011106	013737	054550	001674	MOV	\$LSTAD,FBLKT+2	;LOAD LAST ADDRESS
3739	011114	105037	001365		52\$: CLRB	OVLYLD	;INDICATE THAT THE LOADER IS NOT OVERLAYED
3740	011120	105737	000041		TSTB	41	;SEE WHO LOADED PROGRAM
3741	011124	001004			BNE	1\$;BRANCH IF XXDP
3742	011126	02737	000300	001674	SUB	#96.*2.,FBLKT+2	;SUBTRACT ABS LOADER SIZE
3743	011134	000403			BR	53\$;GO DETERMINE MAX BUFFER SIZE
3744							
3745	011136	162737	006000	001674	1\$: SUB	#1536.*2.,FBLKT+2	;SUBTRACT XXDP LOADER SIZE
3746	011144	013746	001674		53\$: MOV	FBLKT+2,(SP)	;STORE MAXIMUM ADDRESS
3747	011150	1F3716	001672		SUB	FBLKT,(SP)	;DETERMINE MEMORY AVAILIABLE
3748	011154	022716	027000		CMP	#256.*23.*2.,(SP)	;CHECK IF ENOUGH ROOM
3749	011160	101005			1\$;NO LOAD MEMORY AVAILIABLE
3750	011162	005726			TST	(SP)+	;ADJUST STACK
3751	011164	012737	013400	001354	MOV	#256.*23.,MAXBUF	;LOAD MAXIMUM BUFFER SIZE
3752	011172	000403			BR	54\$;LOAD NUMBER OF SOFTWARE COMPARES
3753							
3754	011174	006216			1\$: ASR	(SP)	;CHANGE TO WORDS
3755	011176	012637	001354		MOV	(SP)+,MAXBUF	;LOAD MAXIMUM WORDS TRANSFERRED
3756	011202	012737	000003	001362	54\$: MOV	#3,SOF CMP	;LOAD NO. OF SOFTWARE COMPARES
3757	011210	105037	001360		55\$: CLRB	PERINV	;SET UP FOR NO INTERVAL STATISTICS
3758	011214	012737	177440	001446	56\$: MOV	#177440,RKBAS	;LOAD DEFAULT BUS ADDRESS
3759	011222	012737	000210	001450	57\$: MOV	#210,RKVEC	;LOAD DEFAULT VECTOR ADDRESS
3760	011230	012737	000240	001452	58\$: MOV	#PRS,RKPRI	;LOAD RK06 PRIORITY
3761							
3762	011236	013704	001450		REST1: MOV	RKVEC,R4	;STORE VECTOR ADDRESS
3763	011242	012724	044266		MOV	#I.INTR,(R4)+	;LOAD RK06 VECTOR ADDRESS
3764	011246	013714	001452		MOV	RKPRI,(R4)	; AND INTERRUPT PRIORITY
3765	011252				REST2:		
3766	011252	122737	000013	000041	CMPB	#13,2#41	;LOAD FROM XXDP ?
3767	011260	001003			BNE	99\$;BRANCH IF NOT
3768	011262	104401	055730		TYPE	,XXDPMG	
3769	011266	000000			HALT		
3770	011270				99\$:		
3771	011270	004737	043464		JSR	PC,CLKINT	;INITIALIZE CLOCK VECTOR ADDRESS
3772	011274	104401	057374		TYPE	,SYSO08	;TYPE "READY TO BEGIN TESTING"
3773	011300	005227	177777		INC	#-1	;TYPE HELP MESSAGE ON FIRST PASS
3774	011304	001002			BNE	5\$	
3775	011306	104401	066006		TYPE	,HELP	
3776	011312	004737	053542		5\$: JSR	PC,STKINT	;SET UP TTY INTERRUPT
3777	011316	012737	001452	000036	MOV	RKPRI,TRAPVEC+2	;ALLOW L AND P CLOCK INTERRUPTS
3778	011324	005037	177776		CLR	PS	;ALLOW RK06 INTERRUPTS


```

3788      .SBTTL  MAIN IDLE LOOP
3789
3790 011356 022706 001100      MAIN:  CMP      #STACK,SP      ;CHECK IF STACK PROBLEM
3791 011362 001401          BEQ      1$              ;NO, CONTINUE
3792 011364 000000          HALT          ;*** PROGRAM PROBLEM
3793 011366 004737 051740      1$:   JSR      PC,C.OPT      ;CALL COMMAND OPTIMIZER
3794 011372 004737 044112      JSR      PC,W.WTCH      ;CALL WATCH DOG TIMER
3795 011376 005737 001564      TST      ERCONT        ;CHECK IF IN SPECIAL ERROR SEQUENCE
3796 011402 100765          BMI      MAIN          ;YES, DO NOT PRINT INTERVAL STATISTICS OR
3797                                     ADD DRIVES TO TESTING SEQUENCE
3798 011404 032777 060000 167526  BIT      #SW14,SW13,2SWR ;CHECK IF LOOP ON OPERATION OR
3799                                     INHIBIT PRINT OUT
3800 011412 001022          BNE      11$           ;YES, DO NOT PRINT OUT INTERVAL STATISTICS
3801 011414 105737 001633      TSTB     STATIS        ;CHECK FOR INTERVAL STATISTICS
3802 011420 001417          BEQ      11$           ;NO, CONTINUE SCAN
3803 011422 005004          CLR      R4            ;START WITH DRIVE 0
3804 011424 016405 001524      2$:   MOV      PBLKT(R4),R5 ;GET PARAMETER BLOCK ADDRESS
3805 011430 032765 000001 000014  BIT      #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IS IN USE
3806 011436 001432          BEQ      3$              ;NO, LOOK AT NEXT PARAMETER BLOCK
3807 011440 004737 013102      JSR      PC,STAT      ;GET STATISTICS
3808 011444 005724      3$:   TST      (R4)+        ;INCREMENT DRIVE INDEX
3809 011446 022704 000020      CMP      #20,R4       ;CHECK IF ALL DRIVES SCANNED
3810 011452 101364          BHI      2$              ;NO, DO NEXT DRIVE
3811 011454 105037 001633      CLRB     STATIS        ;RESET FLAG
3812
3813 011460 004037 051466      11$:  JSR      RO,Q.POP      ;GET FIRST DRIVE FROM DRIVE
3814 011464 001314          AVAILQ     ;AVAILABLE QUEUE
3815 011466 012605          MOV      (SP)+,R5     ;STORE PARAMETER BLOCK ADDRESS
3816 011470 001732          BEQ      MAIN         ;IF QUEUE EMPTY WAIT
3817 011472 032765 002000 000014  BIT      #DRPDRV,P.PRST(R5) ;CHECK IF DROP DRIVE
3818 011500 001403          BEQ      12$           ;NO, GENERATE NEW COMMAND
3819 011502 004737 013004      JSR      PC,DROP      ;DROP DRIVE
3820 011506 000723          BR       MAIN         ;GET NEXT AVAILABLE DRIVE
3821
3822 011510 032777 040000 167422 12$:  BIT      #SW14,2SWR    ;CHECK IF LOOP ON OPERATION
3823 011516 001002          BNE     15$           ;YES, DO NOT CLEAR SEEK TO PREVIOUS
3824                                     CYLINDER FLAG
3825 011520 105065 000211      15$:  CLRB     P.SEEK(R5)   ;CLEAR SEEK TO PREVIOUS CYLINDER
3826 011524 004737 021160      JSR      PC,GENERT    ;GENERATE NEW COMMAND
3827 011530 105765 000211      TSTB     P.SEEK(R5)   ;CHECK IF SEEK TO PREVIOUS CYLINDER
3828 011534 001310          BNE     MAIN         ;YES, GET NEXT AVAILIABLE DRIVE
3829 011536 013737 001452 177776  MOV      RKPRI,PS     ;LOCK OUT RK06 INTERRUPTS
3830 011544 005737 001320      TST      BWAITQ      ;CHECK IF BUFFER WAIT QUEUE EMPTY
3831 011550 001013          BNE     25$           ;NO, ENQUEUE PARAMETER BLOCK
3832 011552 004037 022440      JSR      RO,BUFFAL   ;ALLOCATE BUFFER
3833 011556 011600          25$:  CLRB     PS           ;NOT ENOUGH ROOM RETURN
3834 011560 005037 177776      CLR      PS          ;ALLOW RK06 INTERRUPTS
3835 011564 004737 023056      JSR      PC,LDDATA   ;LOAD BUFFER
3836 011570 004037 051406      JSR      RO,Q.PUSH   ;PUT PARAMETER BLOCK ON COMMAND
3837 011574 001324          CINITQ     ;INITIATION QUEUE
3838 011576 000667          BR       MAIN         ;GET NEXT AVAILABLE DRIVE
3839
3840 011600 004037 051406      25$:  JSR      RO,Q.PUSH   ;PUT PARAMETER BLOCK ON BUFFER
3841 011604 001320          BWAITQ     ;WAIT QUEUE
3842 011606 005037 177776      CLR      PS          ;ALLOW RK06 INTERRUPTS
3843 011612 000661          BR       MAIN         ;GET NEXT AVAILABLE DRIVE

```

.SBTTL OPERATOR COMMAND DECODER

```

*****
*
* THE LEGAL COMMANDS ARE:
*   TN-  INITIATE TESTING ON DRIVE N
*   PN-  CHANGE PARAMETER AND INITIATE TEST ON
*         DRIVE N
*   DN-  DROP DRIVE FROM TESTING SEQUENCE
*   SN-  DEMAND PERFORMANCE SUMMARY ON DRIVE N
*   WN-  WRITE RANDOM DATA PATTERNS ON DRIVE N
*         AND INITIATE TESTING.
*
*CALL JSR   PC,OPRCMD
*      RETURN
*
*
*   COMMAND          ROUTINE
*   -----          -
*   TN               TESTDV
*   PN               PARMOV
*   DN               DROPDV
*   SN               STATDV
*   WN               WRTEPK
*
*****

```

```

3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872 011614 010446
3873 011616 104401 056230
3874 011622 104403
3875 011624 012604
3876 011626 116437 000001 056065
3877
3878 011634 122737 000101 056065
3879 011642 001421
3880 011644 122737 000060 056065
3881 011652 101004
3882 011654 122737 000067 056065
3883 011662 103006
3884 011664 104401 056050 2$:
3885 011670 104401 056070
3886 011674 012604
3887 011676 000207
3888
3889 011700 142777 000370 056065 3$:
3890 011706 0105 5 4$:
3891 011710 122714 000124
3892 011714 001427
3893 011716 122714 000120
3894 011722 001512
3895 011724 122714 000104
3896 011730 001553
3897 011732 122714 000123
3898 011736 001002
3899 011740 000137 012440

```

```

OPRCMD: MOV R4, -(SP) ; STORE R4 ON STACK
        TYPE ,OPR007 ; TYPE "PLEASE ENTER COMMAND"
        RDLIN ; READ LINE FROM TTY
        MOV (SP)+, R4 ; STORE ADDRESS OF INPUT STRING
        MOVB 1(R4), OPR001 ; MOVE SECOND CHARACTER TO DRIVE
        ; REQUEST STORAGE
        CMPB #'A, OPR001 ; CHECK IF ALL DRIVES
        BEQ 4$ ; YES, PROCESS ALL DRIVES
        CMPB #'0, OPR001 ; CHECK IF LEGAL DRIVE NUMBER
        BHI 2$ ; NO, PRINT ERROR
        CMPB #'7, OPR001 ; CHECK IF 0-7
        BHIS 3$ ; YES, PROCESS DRIVE COMMAND
        2$: TYPE ,OPR000 ; TYPE "DRIVE NUMBER N ILLEGAL?"
        TYPE ,OPR002
        MOV (SP)+, R4 ; RESTORE R4
        RTS PC ; RETURN
        3$: BICB #'370, OPR001 ; CLEAR UNIMPORTANT BITS
        4$: MOV R5, -(SP) ; STORE R5 ON STACK
        CMPB #'t, (R4) ; CHECK IF TEST DRIVE
        BEQ TESTDV ; YES, GO TO TEST DRIVE
        CMPB #'P, (R4) ; CHECK IF TEST AND CHANGE PARAMETERS
        BEQ PARMOV ; YES, GO TEST AND CHANGE PARAMETERS
        CMPB #'D, (R4) ; CHECK IF DROP DRIVE
        BEQ DROPDV ; YES, GO DROP DRIVE
        CMPB #'S, (R4) ; CHECK IF DEMAND STATISTICS
        BNE 5$
        JMP STATDV ; YES, GO TO DEMAND STATISTICS

```

3900								
3901	011744	122714	000127	5\$:	CMPB	#'W,(R4)		;CHECK IF WRITE PACK AND TEST
3902	011750	001002			BNE	6\$		
3903	011752	000137	012614		JMP	WRTEPK		;YES, GO WRITE PACK AND TEST
3904								
3905	011756	111437	056042	6\$:	MOVB	(P4),ILLCMD		;STORE COMMAND FOR PRINT OUT
3906	011762	104401	056011		TYPE	,ILLCOM		;TYPE ERROR MESSAGE
3907	011766	012605			MOV	(SP)+,R5		;RESTORE R5
3908	011770	012604			MOV	(SP)+,R4		;RESTORE R4
3909	011772	000207			RTS	PC		;RETURN

```

3910 .SBTTL TEST DRIVE COMMAND
3911
3912 011774 122737 000101 056065 TESTDV: CMPB #'A,OPR001 ;CHECK IF ALL DRIVES ARE TO BE TESTED
3913 012002 001027 BNE 10$ ;NO, DO SPECIFIC DRIVE
3914 012004 105037 001632 CLR8 DRVCNT ;CLEAR DRIVE COUNT
3915 012010 005004 CLR R4 ;CLEAR DRIVE COUNT
3916 012012 016405 001524 1$: MOV PBLKT(R4),R5 ;GET PARAMETER BLOCK ADDRESS
3917 012016 032765 000001 000014 BIT #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
3918 012024 001004 BNE 2$ ;YES, GO TO NEXT DRIVE
3919 012026 004737 013360 JSR PC,TEST ;GO TEST DRIVE
3920 012032 105237 001632 INCB DRVCNT ;INCREMENT DRIVE COUNT
3921 012036 005724 2$: TST (R4)+ ;ADDRESS NEXT DRIVE (ADD 2)
3922 012040 022704 000020 CMP #20,R4 ;CHECK IF FINISHED
3923 012044 001362 BNE 1$ ;NO, SET UP NEXT DRIVE
3924 012046 105737 001632 TSTB DRVCNT ;CHECK IF ANY DRIVE IN ASSIGNMENT SEQUENCE
3925 012052 001033 BNE 15$ ;YES, RETURN
3926 012054 104401 056407 TYPE OPR013 ;TYPE "ALL DRIVES CURRENTLY UNDER TEST?"
3927 012060 000430 BR 15$ ;RETURN
3928
3929 012062 113704 056065 10$: MOVB OPR001,R4 ;STORE DRIVE NUMBER
3930 012066 006304 ASL R4 ;MULTIPLY DRIVE NUMBER BY 2
3931 012070 016405 001524 MOV PBLKT(R4),R5 ;GET PARAMETER ADDRESS
3932 012074 032765 000001 000014 BIT #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE ASSIGNED
3933 012102 001003 BNE 11$ ;YES, PRINT DRIVE ALREADY ASSIGNED
3934 012104 004737 013360 JSR PC,TEST ;GO TEST DRIVE
3935 012110 000414 BR 15$ ;RETURN
3936
3937 012112 013737 001452 177776 11$: MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
3938 012120 152737 000060 056065 BISB #60,OPR001 ;MAKE NUMBER ASCIZ
3939 012126 104401 056050 TYPE ,OPR000 ;TYPE "DRIVE N ALREADY ASSIGNED?"
3940 012132 104401 056116 TYPE ,OPR004
3941 012136 005037 177776 CLR PS ;ALLOW RK06 INTERRUPTS
3942 012142 012605 15$: MOV (SP)+,R5 ;RESTORE R5
3943 012144 012604 MOV (SP)+,R4 ;RESTORE R4
3944 012146 000207 RTS PC ;RETURN
3945
3946 .SBTTL CHANGE PARAMETERS COMMAND
3947
3948 012150 122737 000101 056065 PARMDV: CMPB #'A,OPR001 ;CHECK IF PARAMETERS FOR ALL DRIVES
3949 012156 001020 BNE 10$ ;NO, DO SPECIFIC DRIVE
3950 012160 005004 CLR R4 ;CLEAR DRIVE COUNT
3951 012162 016405 001524 1$: MOV PBLKT(R4),R5 ;GET PARAMETER BLOCK ADDRESS
3952 012166 004737 016170 JSR PC,PARM ;ASK FOR PARAMETERS
3953 012172 032765 000001 000014 BIT #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
3954 012200 001002 BNE 2$ ;YES, GO TO NEXT DRIVE
3955 012202 004737 013360 JSR PC,TEST ;GO TEST DRIVE
3956 012206 005724 2$: TST (R4)+ ;ADDRESS NEXT DRIVE (ADD 2)
3957 012210 022704 000020 CMP #20,R4 ;CHECK IF FINISHED
3958 012214 001362 BNE 1$ ;NO, SET UP NEXT DRIVE
3959 012216 000415 BR 15$ ;RETURN
3960
3961 012220 113704 056065 10$: MOVB OPR001,R4 ;STORE DRIVE NUMBER
3962 012224 006304 ASL R4 ;MULTIPLY DRIVE NUMBER BY 2
3963 012226 016405 001524 MOV PBLKT(R4),R5 ;GET PARAMETER ADDRESS
3964 012232 004737 016170 JSR PC,PARM ;GET NEW PARAMETERS
3965 012236 032765 000001 000014 BIT #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE

```



```

3966 012244 001002          BNE      15$      ;YES, RETURN
3967 012246 004737 013360   JSR      PC,TEST ;GO TEST DRIVE
3968 012252 012605          15$:    MOV      (SP)+,R5 ;RESTORE R5
3969 012254 012604          MOV      (SP)+,R4 ;RESTORE R4
3970 012256 000207          RTS      PC       ;RETURN
3971
3972          .SBTTL  DROP DRIVE COMMAND
3973
3974 012260 122737 000101 056065  DROPDV:  CMPB     #'A,OPR001 ;CHECK IF ALL DRIVES ARE TO BE DROPPED
3975 012266 001030          BNE     10$      ;NO, DO SPECIFIC DRIVE
3976 012270 105037 001632   CLR     CLRB     ;CLEAR DRIVE COUNT
3977 012274 005004          CLR     R4       ;CLEAR DRIVE COUNT
3978 012276 016405 001524   1$:    MOV     PBLKT(R4),R5 ;GET PARAMETER BLOCK ADDRESS
3979 012302 032765 000001 000014   BIT     #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
3980 012310 001405          BEQ     2$      ;NO, GO TO NEXT DRIVE
3981 012312 052765 002000 000014   BIS     #DRPDRV,P.PRST(R5) ;INDICATE DRIVE IS TO BE DROPPED
3982 012320 105237 001632   INCB   DRVCNT   ;INCREMENT DRIVE COUNT
3983 012324 005724          2$:    TST     (R4)+   ;ADDRESS NEXT DRIVE (ADD 2)
3984 012326 022704 000020   CMP     #20,R4  ;CHECK IF FINISHED
3985 012332 001361          BNE     1$      ;NO, SET UP NEXT DRIVE
3986 012334 105737 001632   TST     DRVCNT  ;CHECK IF ANY DRIVES AVAILIABLE
3987 012340 001034          BNE     15$     ;YES, RETURN
3988 012342 104401 056451   TYPE   OPR014  ;TYPE "NO DRIVES IN USE?"
3989 012346 000431          BR      15$     ;RETURN
3990
3991 012350 113704 056065   10$:   MOV     OPR001,R4 ;STORE DRIVE NUMBER
3992 012354 006304          ASL     R4       ;MULTIPLY DRIVE NUMBER BY 2
3993 012356 016405 001524   MOV     PBLKT(R4),R5 ;GET PARAMETER ADDRESS
3994 012362 032765 000001 000014   BIT     #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE ASSIGNED
3995 012370 001404          BEQ     11$     ;NO, PRINT DRIVE N NOT ASSIGNED
3996 012372 052765 002000 000014   BIS     #DRPDRV,P.PRST(R5) ;INDICATE DRIVE IS TO BE DROPPED
3997 012400 000414          BR      15$     ;RETURN
3998
3999 012402 013737 001452 177776 11$:   MOV     RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
4000 012410 152737 000060 056065   BISB   #60,OPR001 ;MAKE NUMBER ASCII
4001 012416 104401 056050          TYPE   ,OPR000 ;TYPE "DRIVE N NOT ASSIGNED?"
4002 012422 104401 056142          TYPE   ,OPR005
4003 012426 005037 177776          CLR     PS      ;ALLOW RK06 INTERRUPTS
4004 012432 012605          15$:   MOV     (SP)+,R5 ;RESTORE R5
4005 012434 012604          MOV     (SP)+,R4 ;RESTORE R4
4006 012436 000207          RTS     PC      ;RETURN
4007
4008          .SBTTL  GET DRIVE STATISTICS COMMAND
4009
4010 012440 122737 000101 056065  STATDV:  CMPB     #'A,OPR001 ;CHECK IF STATISTICS FOR ALL DRIVES
4011 012446 001027          BNE     10$     ;NO, OP SPECIFIC DRIVE
4012 012450 105037 001632   CLR     CLRB     ;CLEAR DRIVE COUNT
4013 012454 005004          CLR     R4       ;CLEAR DRIVE COUNT
4014 012456 016405 001524   1$:    MOV     PBLKT(R4),R5 ;GET PARAMETER BLOCK ADDRESS
4015 012462 032765 000001 000014   BIT     #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
4016 012470 001404          BEQ     2$      ;NO, GO TO NEXT DRIVE
4017 012472 004737 013102   JSR     PC,STAT  ;GATHER STATISTICS
4018 012476 105237 001632   INCB   DRVCNT  ;INCREMENT DRIVE COUNT
4019 012502 005724          2$:    TST     (R4)+   ;ADDRESS NEXT DRIVE (ADD 2)
4020 012504 022704 000020   CMP     #20,R4  ;CHECK IF FINISHED
4021 012510 001362          BNE     1$      ;NO, SET UP NEXT DRIVE

```

```

4022 012512 105737 001632          TSTB  DRVCNT          ;CHECK IF ANY DRIVES AVAILIABLE
4023 012516 001033          BNE   15$            ;YES, RETURN
4024 012520 104401 056451          TYPE  OPRO14        ;TYPE "NO DRIVES IN USE?"
4025 012524 000430          BR    15$            ;RETURN
4026
4027 012526 113704 056065          10$: MOVB  OPRO01,R4    ;STORE DRIVE NUMBER
4028 012532 006304          ASL   R4             ;MULTIPLY DRIVE NUMBER BY 2
4029 012534 016405 001524          MOV   PBLKT(R4),R5   ;GET PARAMETER ADDRESS
4030 012540 032765 000001 000014          BIT   #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE ASSIGNED
4031 012546 001403          BEQ  11$            ;PRINT DRIVE N NOT ASSIGNED
4032 012550 004737 013102          JSR   PC,STAT        ;GO GATHER DRIVE STATISTICS
4033 012554 000414          BR    15$            ;RETURN
4034
4035 012556 013737 001452 177776 11$: MOV   RKPRI,PS      ;LOCK OUT RK06 INTERRUPTS
4036 012564 152737 000060 056065          BISB  #60,OPRO01    ;MAKE NUMBER ASCII
4037 012572 104401 056050          TYPE  ,OPRO00        ;TYPE "DRIVE N NOT ASSIGNED?"
4038 012576 104401 056142          TYPE  ,OPRO05
4039 012602 005037 177776          CLR   PS            ;ALLOW RK06 INTERRUPTS
4040 012606 012605          15$: MOV   (SP)+,R5    ;RESTORE R5
4041 012610 012604          MOV   (SP)+,R4      ;RESTORE R4
4042 012612 000207          RTS   PC             ;RETURN
4043
4044          .SBTTL WRITE PACK AND TEST COMMAND
4045
4046 012614 122737 000101 056065 WRTEPK: CMPB  #'A,OPRO01    ;CHECK IF ALL DRIVES ARE TO BE TESTED
4047 012622 001032          BNE  10$            ;NO, DO SPECIFIC DRIVE
4048 012624 105037 001632          CLRB  DRVCNT        ;CLEAR DRIVE COUNT
4049 012630 005004          CLR   R4            ;CLEAR DRIVE COUNT
4050 012632 016405 001524          1$: MOV   PBLKT(R4),R5 ;GET PARAMETER BLOCK ADDRESS
4051 012636 032765 000001 000014          BIT   #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
4052 012644 001007          BNE  2$            ;YES, GO TO NEXT DRIVE
4053 012646 112765 000020 000210          MOVB  #BIT4,P.ASSN(R5) ;SET FLAG FOR WRITE PACK
4054 012654 004737 013360          JSR   PC,TEST       ;GO ASSIGN DRIVE
4055 012660 105237 001632          INCB  DRVCNT        ;INCREMENT DRIVE COUNT
4056 012664 005724          2$: TST   (R4)+      ;ADDRESS NEXT DRIVE (ADD 2)
4057 012666 022704 000020          CMP   #20,R4        ;CHECK IF FINISHED
4058 012672 001357          BNE  1$            ;NO, SET UP NEXT COMMAND
4059 012674 105737 001632          TSTB  DRVCNT        ;CHECK IF ANY DRIVES NOT IN USE
4060 012700 001436          BEQ  15$            ;NO, RETURN
4061 012702 104401 056407          TYPE  OPRO13        ;TYPE "ALL DRIVES CURRENTLY UNDER TEST"
4062 012706 000433          BR    15$            ;RETURN
4063
4064 012710 113704 056065          10$: MOVB  OPRO01,R4    ;STORE DRIVE NUMBER
4065 012714 006304          ASL   R4             ;MULTIPLY DRIVE NUMBER BY 2
4066 012716 016405 001524          MOV   PBLKT(R4),R5   ;GET PARAMETER ADDRESS
4067 012722 032765 000001 000014          BIT   #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE ASSIGNED
4068 012730 001006          BNE  11$            ;YES, PRINT DRIVE ALREADY ASSIGNED
4069 012732 112765 000020 000210          MOVB  #BIT4,P.ASSN(R5) ;SET FLAG FOR WRITE PACK
4070 012740 004737 013360          JSR   PC,TEST       ;GO ASSIGN DRIVE
4071 012744 000414          BR    15$            ;RETURN
4072
4073 012746 013737 001452 177776 11$: MOV   RKPRI,PS      ;LOCK OUT RK06 INTERRUPTS
4074 012754 152737 000060 056065          BISB  #60,OPRO01    ;MAKE NUMBER ASCII
4075 012762 104401 056050          TYPE  ,OPRO00        ;TYPE "DRIVE N ALREADY ASSIGNED?"
4076 012766 104401 056116          TYPE  ,OPRO04
4077 012772 005037 177776          CLR   PS            ;ALLOW RK06 INTERRUPTS

```

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 91
WRITE PACK AND TEST COMMAND

SEQ 0089

4078 012776 012605
4079 013000 012604
4080 013002 000207
4081

15\$: MOV (SP)+,R5 ;RESTORE R5
MOV (SP)+,R4 ;RESTORE R4
RTS PC ;RETURN

.SBTTL DROP DRIVE AND GATHER STATISTICS ROUTINES

4082									
4083									
4084	013004	013746	177776						
4085	013010	013737	001452	177776	DROP:	MOV	PS, -(SP)	; STORE PSW	
4086	013016	005065	000014			MOV	RKPRI, PS	; LOCK OUT TTY AND RK06 INTERRUPTS	
4087	013022	005065	000212			CLR	P.PRSI(R5)	; CLEAR PROGRAM STATUS REGISTER	
4088	013026	116537	000000	056357		CLR	P.ERR(R5)	; CLEAR ERROR FLAGS	
4089	013034	152737	000060	056357		MOV	P.DRVN(R5), OPR011	; LOAD DRIVE NUMBER	
4090	013042	104401	056340			BISB	#60, OPR011	; MAKE IT ASCII	
4091	013046	104401	056162			TYPE	, OPR010	; TYPE "DRIVE N HAS BEEN DROPPED	
4092	013052	132765	000001	000210		TYPE	, OPR006	; FROM TEST SEQUENCE"	
4093	013060	001404				BITB	#BIT0, P.ASSN(R5)	; SEE IF SIZING FOR DRIVE	
4094	013062	105065	000210			BEG	2\$; BR IF NOT	
4095	013066	000137	013352			CLRB	P.ASSN(R5)	; CLEAR ASSIGNMENT CODE BITS	
4096	013072	105065	000210			JMP	STAT01		
4097	013076	000137	013114			2\$:	CLRB	P.ASSN(R5)	; CLEAR ASSIGNMENT CODE BITS
4098						JMP	STAT00		
4099	013102	013746	177776			STAT:	MOV	PS, -(SP)	; STORE PSW
4100	013106	013737	001452	177776		STAT00:	MOV	RKPRI, PS	; LOCK OUT TTY AND RK06 INTERRUPTS
4101	013114	104401	056526				TYPE	DRVST	; **DRIVE STATISTICS
4102	013120	116537	000000	056357			MOV	P.DRVN(R5), OPR011	; LOAD DRIVE NUMBER FOR PRINT OUT
4103	013126	152737	000060	056357			BISB	#60, OPR011	; MAKE IT ASCII
4104	013134	104401	056340				TYPE	, OPR010	; TYPE "DRIVE N"
4105	013140	004737	027616				JSR	PC, PRISER	; PRINT DRIVE AND PACK SERIAL NUMBERS
4106	013144	004737	043670				JSR	PC, PRITIM	; PRINT TIME IF CLOCK AVAILIABLE
4107	013150	104401	056604				TYPE	ST001	; TYPE "ORDERS PERFORMED"
4108	013154	010546					MOV	R5, -(SP)	; GET PARAMETER BLOCK ADDRESS
4109	013156	062716	000152				ADD	#P.NODR, (SP)	; CALCULATE ADDRESS FOR PRINT OUT
4110	013162	004737	055174				JSR	PC, \$DB20	; CONVERT TO ASCII
4111	013166	104401	056626				TYPE	ST002	; TYPE "WORDS WRITTEN*65K"
4112	013172	010546					MOV	R5, -(SP)	; GET PARAMETER BLOCK ADDRESS
4113	013174	062716	000160				ADD	#P.NWRT+2, (SP)	; CALCULATE ADDRESS FOR PRINT OUT
4114	013200	004737	055174				JSR	PC, \$DB20	; CONVERT TO ASCII
4115	013204	104401	056653				TYPE	ST003	; TYPE "WORDS READ*65K"
4116	013210	010546					MOV	R5, -(SP)	; GET PARAMETER BLOCK ADDRESS
4117	013212	062716	000166				ADD	#P.NRD+2, (SP)	; CALCULATE ADDRESS FOR PRINT OUT
4118	013216	004737	055174				JSR	PC, \$DB20	; CONVERT TO ASCII
4119	013222	104401	056675				TYPE	ST004	; TYPE "SOFT DATA ERRORS"
4120								"ECC"	
4121	013226	016546	000176				MOV	P.SECC(R5), -(SP)	; STORE SOFT ECC ERROR COUNT
4122	013232	004737	055132				JSR	PC, \$SB20	; CONVERT TO DECIMAL
4123	013236	104401	056730				TYPE	ST005	; TYPE "REREAD"
4124	013242	016546	000172				MOV	P.SRAD(R5), -(SP)	; SAVE SOFT REREAD CORRECTABLE
4125	013246	004737	055132				JSR	PC, \$SB20	; CONVERT TO DECIMAL
4126	013252	104401	056744				TYPE	ST006	; TYPE "OFFSET"
4127	013256	016546	000174				MOV	P.SOFF(R5), -(SP)	; SAVE SOFT OFFSET CORRECTABLE
4128	013262	004737	055132				JSR	PC, \$SB20	; CONVERT TO DECIMAL
4129	013266	104401	056760				TYPE	ST007	; TYPE "HARD DATA ERRORS"
4130	013272	016546	000200				MOV	P.HARD(R5), -(SP)	; SAVE HARD DATA ERRORS
4131	013276	004737	055132				JSR	PC, \$SB20	; CONVERT TO DECIMAL
4132	013302	104401	057004				TYPE	ST008	; TYPE "SEEK INCOMPLETES
4133	013306	016546	070202				MOV	P.NSKI(R5), -(SP)	; TYPE NUMBER OF SEEK INCOMPLETES
4134	013312	004737	055132				JSR	PC, \$SB20	; CONVERT TO DECIMAL
4135	013316	104401	057030				TYPE	ST009	; TYPE "OPERATION INCOMPLETES"
4136	013322	016546	000204				MOV	P.NOPI(R5), -(SP)	; SAVE NUMBER OF OPERATION INCOMPLETES
4137	013326	004737	055132				JSR	PC, \$SB20	; CONVERT TO DECIMAL

4138	013332	104401	057076
4139	013336	016546	000206
4140	013342	004737	055132
4141	013346	104401	001165
4142	013352	012637	177776
4143	013356	000207	
4144			

	TYPE	STTD10	;TYPE "OTHER ERRORS"
	MOV	P.NER(R5),-(SP)	;SAVE NUMBER OF OTHER ERRORS
	JSR	PC,\$SB2C	;CONVERT TO DECIMAL
	TYPE	,\$CRLF	;TYPE <CR><LF>
STAT01:	MOV	(SP)+,PS	;RESTORE PSW
	RTS	PC	;RETURN

```

4145
4146
4147 013360 116546 000210
4148 013364 010446
4149 013366 010346
4150 013370 010503
4151 013372 062703 000122
4152
4153 013376 010504
4154 013400 062704 000232
4155
4156 013404 005023
4157 013406 020304
4158 013410 001375
4159 013412 116504 000000
4160 013416 156437 001514 001513
4161 013424 012603
4162 013426 012604
4163 013430 112665 000210
4164
4165 013434 105065 000117
4166 013440 005037 001500
4167 013444 152765 000001 000210
4168 013452 052765 000001 000014
4169 013460 112765 000141 000001
4170 013466 112765 000141 000123
4171 013474 004037 051406
4172 013500 001324
4173 013502 000207

```

```

.SBTTL INITIATE DRIVE ASSIGNMENT SEQUENCE
TEST:  MOVB  P.ASSN(R5),-(SP) ;SAVE ASSIGNMENT CODE
      MOV   R4,-(SP) ;STORE R4 ON STACK
      MOV   R3,-(SP) ;STORE R3 ON STACK
      MOV   R5,R3 ;LOAD PARAMETER BLOCK ADDRESS INTO R3
      ADD   #P.RDPT,R3 ;CALCULATE BEGINNING OF STATISTICS AND
      ;ERROR AREA
      MOV   R5,R4 ;LOAD PARAMETER BLOCK ADDRESS INTO R4
      ADD   #P.EXAR,R4 ;CALCULATE END OF STATISTICS AND
      ;ERROR AREA
IS:   CLR   (R3)+ ;CLEAR STATISTIC AND ERROR AREA
      CMP   R3,R4 ;CHECK IF FINISHED
      BNE   IS ;NO CONTINUE
      MOVB  P.DRVN(R5),R4 ;STORE DRIVE NUMBER
      BISB  INTMSK(R4),0.OVER ;SET UP FOR OVERLAPPING SEEKS
      MOV   (SP)+,R3 ;RESTORE R3
      MOV   (SP)+,R4 ;RESTORE R4
      MOVB  (SP)+,P.ASSN(R5) ;RESTORE ASSIGNMENT CODE
TEST1: CLRB  P.ECMP(R5) ;CLEAR ECC COMPARE FLAG
      CLR   OPTFLG ;CLEAR CMND OPTIMIZER FLAG
      BISB  #BIT0,P.ASSN(R5) ;SET READ STATUS COMMAND FLAG
      BIS   #DRVUSE,P.PRST(R5) ;SET DRIVE IN USE
      MOVB  #RDSTAT,P.CMND(R5) ;READ ALL DRIVE STATUS
      MOVB  #RDSTAT,P.RCMD(R5)
      JSR   RD,Q.PUSH ;ENQUEUE PARAMETER BLOCK IN
      ;COMMAND INITIATION QUEUE
      RTS   PC ;RETURN

```

```

.SBTTL DRIVE ASSIGNMENT SEQUENCE
4174
4175
4176 013504 132765 000001 000210 ASNORM: BITB #BIT0,P.ASSN(R5) ;CHECK IF READ STATUS
4177 013512 001467 BEQ ASN4$ ;NO, CHECK IF START SPINDLE
4178 013514 005737 027156 TST NEWFLG ;SEE IF DTYE ERROR ON ASSIGNING
4179 013520 001404 BEQ 1$ ;OR IF NO
4180 013522 005037 027156 CLR NEWFLG ;ELSE CLEAR DTYE ON ASSIGN FLAG
4181 013526 000137 013434 JMP TEST1 ;REPEAT SELECT FOR DRIVE THAT HAD DTYE
4182 013532 142765 000001 000210 1$: BICB #BIT0,P.ASSN(R5) ;CLEAR READ STATUS COMMAND ISSUED
4183 013540 016565 000054 000224 MOV P.A11(R5),P.SERL(R5) ;STORE SERIAL NUMBER
4184 013546 042765 100007 000224 BIC #C(M.SER),P.SERL(R5) ;KEEP SERIAL NUMBER
4185 013554 006265 000224 ASR P.SERL(R5) ;SHIFT 3 BITS RIGHT
4186 013560 006265 000224 ASR P.SERL(R5)
4187 013564 006265 000224 ASR P.SERL(R5)
4188 013570 032765 000200 000036 BIT #DRDY,P.DS(R5) ;CHECK IF DRIVE READY
4189 013576 001415 BEQ ASN2$ ;NO, ISSUE START SPINDLE
4190 013600 116565 000123 000135 ASN1$: MOVB P.RCMD(R5),P.LCMD(R5) ;STORE PREVIOUS COMMAND
4191 013606 112765 000103 000001 MOVB #PACK,P.CMND(R5) ;ISSUE PACK ACKNOWLEDGE
4192 013614 112765 000103 000123 MOVB #PACK,P.RCMD(R5)
4193 013622 152765 000004 000210 BISB #BIT2,P.ASSN(R5) ;SET PACK ACKNOWLEDGE FLAG
4194 013630 000414 BR ASN3$ ;ISSUE PACK ACKNOWLEDGE AND WAIT
4195
4196 013632 116565 000123 000135 ASN2$: MOVB P.RCMD(R5),P.LCMD(R5) ;STORE PREVIOUS COMMAND
4197 013640 112765 000111 000001 MOVB #SRTSPL,P.CMND(R5) ;ISSUE START SPINDLE
4198 013646 112765 000111 000123 MOVB #SRTSPL,P.RCMD(R5)
4199 013654 152765 000002 000210 BISB #BIT1,P.ASSN(R5) ;SET START SPINDLE FLAG
4200 013662 004037 051406 ASN3$: JSR RO,Q.PUSH ;ENQUEUE PARAMETER BLOCK IN
4201 013666 001324 CINITQ ;COMMAND INITIATION QUEUE
4202 013670 000207 RTS PC ;RETURN
4203
4204 013672 132765 000002 000210 ASN4$: BITB #BIT1,P.ASSN(R5) ;CHECK IF START SPINDLE
4205 013700 001414 BEQ 15$ ;NO, CHECK IF PACK ACKNOWLEDGE
4206 013702 142765 000002 000210 BICB #BIT1,P.ASSN(R5) ;CLEAR START SPINDLE FLAG
4207 013710 032765 000200 000036 BIT #DRDY,P.DS(R5) ;CHECK IF DRIVE READY
4208 013716 001330 BNE ASN1$ ;YES, ISSUE PACK ACKNOWLEDGE
4209 013720 052765 002000 000212 BIS #BIT10,P.ERR(R5) ;SET DRIVE NOT READY AFTER START SPINDLE
4210 013726 000137 025312 JMP ABNORM ;REPORT ERROR
4211
4212 013732 132765 000004 000210 15$: BITB #BIT2,P.ASSN(R5) ;CHECK IF PACK ACKNOWLEDGE
4213 013740 001431 BEQ 18$ ;NO, CHECK IF RECAL
4214 013742 142765 000004 000210 BICB #BIT2,P.ASSN(R5) ;CLEAR PACK ACKNOWLEDGE FLAG
4215 013750 032765 000100 000036 BIT #VV,P.DS(R5) ;CHECK IF VOLUME VALID IS SET
4216 013756 001005 BNE 16$ ;YES, CHECK IF PACK IS TO BE WRITTEN
4217 013760 052765 004000 000212 BIS #BIT11,P.ERR(R5) ;SET VOLUME VALID NOT SET AFTER PACK
4218 ;ACKNOWLEDGE
4219 013766 000137 025312 JMP ABNORM ;REPORT ERROR
4220
4221 013772 116565 000123 000135 16$: MOVB P.RCMD(R5),P.LCMD(R5) ;STORE PREVIOUS COMMAND
4222 014000 112765 000113 000001 MOVB #RECAL,P.CMND(R5) ;ISSUE RECAL
4223 014006 112765 000113 000123 MOVB #RECAL,P.RCMD(R5)
4224 014014 152765 000100 000210 BISB #BIT6,P.ASSN(R5) ;SET RECAL ISSUED
4225 014022 000717 BR ASN3$
4226
4227 014024 132765 000100 000210 18$: BITB #BIT6,P.ASSN(R5) ;CHECK IF RECAL
4228 014032 001457 BEQ 23$ ;NO, CHECK IF READ PACK SERIAL NUMBER
4229 014034 142765 000100 000210 BICB #BIT6,P.ASSN(R5) ;RESET RECAL ISSUED

```

4230	014042	152765	000010	000210		BISB	#BIT3,P.ASSN(R5) ;SET ASSIGNMENT CODE FOR PACK
4231							; SERIAL NUMBER READ
4232	014050	012765	001000	000004		MOV	#1000,P.SECT(R5) ; TRACK 2 SECTOR 0
4233	014056	116565	000123	000135		MOV	P.RCMD(R5),P.LCMD(R5) ;STORE PREVIOUS COMMAND
4234	014064	112765	000121	000001	20\$:	MOV	#RDATA,P.CMND(R5) ;LOAD READ COMMAND FOR PACK
4235	014072	112765	000121	000123		MOV	#RDATA,P.RCMD(R5) ; SERIAL NUMBER
4236	014100	012765	177774	000012		MOV	#-4,P.WC(R5) ;LOAD WORD COUNT FOR THE FIRST FOUR WORDS
4237	014106	004737	027074			JSR	PC,GETNUM ;GET CORRECT END CYL NUM
4238	014118	013765	027154	000002		MOV	LSTCYL,P.CYLN(R5) ;SET CYLINDER TO 632/1456
4239	014120	013765	027154	000124		MOV	LSTCYL,P.RCYL(R5)
4240	014126	016565	000004	000126		MOV	P.SECT(R5),P.RSEC(R5) ;STORE TRACK AND SECTOR
4241	014134	005737	001320			TST	BWAITQ ;CHECK IF BUFFER WAIT QUEUE IS EMPTY
4242	014140	001010				BNE	22\$;NO, ENQUEUE PARAMETER BLOCK IN BUFFER
4243							WAIT Q
4244	014142	004037	022440			JSR	RD,BUFFAL ;ALLOCATE BUFFER
4245	014146	014162				22\$;NOT ENOUGH ROOM RETURN
4246	014150	005037	177776			CLR	PS ;ALLOW RK06 INTERRUPTS
4247	014154	004737	023056			JSR	PC,LDDATA ;CLEAR BUFFER
4248	014160	000640				BR	ASN3\$;ISSUE COMMAND
4249							
4250	014162	004037	051406		22\$:	JSR	RD,Q.PUSH ;PUT PARAMETER BLOCK ON BUFFER
4251	014166	001320				BWAITQ	; WAIT QUEUE
4252	014170	000207				RTS	PC ;RETURN
4253							
4254	014172	032777	040000	164740	23\$:	BIT	#SW14,2SWR ;CHECK IS CYCLE ON OPERATION
4255	014200	001542				BEQ	43\$;NO, CONTINUE WITH ASSIGNMENT PROCESS
4256	014202	105765	000211			TSTB	P.SEEK(R5) ;CHECK IF SEEK TO PREVIOUS POSITION
4257	014206	001444				BEQ	28\$;NO, SERVICE INTERRUPT
4258	014210	105065	000211			CLRB	P.SEEK(R5) ;CLEAR FLAG
4259	014214	116565	000123	000001		MOV	P.RCMD(R5),P.CMND(R5) ;REISSUE LAST COMMAND
4260	014222	016565	000124	000002		MOV	P.RCYL(R5),P.CYLN(R5) ;LOAD CYLINDER ADDRESS
4261	014230	016565	000126	000004		MOV	P.RSEC(R5),P.SECT(R5) ;LOAD PREVIOUS TRACK AND SECTOR
4262	014236	016565	000130	000012		MOV	P.RWC(R5),P.WC(R5) ;LOAD PREVIOUS WORD COUNT
4263	014244	116565	000122	000121		MOV	P.RDPT(R5),P.DPAT(R5) ;LOAD PREVIOUS PATTERN
4264	014252	132765	000010	000210		BITB	#BIT3,P.ASSN(R5) ;CHECK IF READ PACK SERIAL NUMBER
4265	014260	001003				BNE	25\$;YES, DO NOT SET WRITE BEFORE WRITE CHECK
4266	014262	052765	000200	000014		BIS	#W.WCK,P.PRST(R5) ;SET WRITE BEFORE WRITE CHECK
4267	014270	005737	001320		25\$:	TST	BWAITQ ;CHECK IF ANY DRIVE WAITING FOR BUFFER
4268	014274	001332				BNE	22\$;YES, ENQUEUE PARAMETER BLOCK
4269	014276	004037	022440			JSR	RD,BUFFAL ;GO ALLOCATE BUFFER
4270	014302	014162				22\$;NOT ENOUGH ROOM RETURN
4271	014304	005037	177776			CLR	PS ;ALLOW RK06 INTERRUPTS
4272	014310	004737	023056			JSR	PC,LDDATA ;GO LOAD DATA
4273	014314	000137	013662			JMP	ASN3\$;GO ISSUE COMMAND
4274							
4275	014320	004037	025032		28\$:	JSR	RD,CHKADD ;CHECK SECTOR, TRACK, CYLINDER,
4276							; WORD COUNT, AND BUS ADDRESS
4277	014324	015232				60\$;ERROR RETURN
4278	014326	132765	000010	000210		BITB	#BIT3,P.ASSN(R5) ;CHECK IF READ PACK SERIAL NUMBER
4279	014334	001424				BEQ	35\$;NO, RELEASE BUFFER
4280	014336	010446				MOV	R4,-(SP) ;STORE R4 ON STACK
4281	014340	016504	000010			MOV	P.BALO(R5),R4 ;LOAD R4 TO GET PACK SERIAL NUMBER
4282	014344	012465	000226			MOV	(R4)+,P.PKSR(R5) ;GET PACK SERIAL NUMBER
4283	014350	012465	000230			MOV	(R4)+,P.PKSR+2(R5)
4284	014354	005724				TST	(R4)+ ;ADJUST R4 TO ADDRESS ALLIGNMENT INDICATION
4285	014356	005714				TST	(R4) ;CHECK IF DATA PACK

4286	014360	001423				BEQ	37\$; YES, CONTINUE
4287	014362	012604			30\$:	MOV	(SP)+,R4		; RESTORE R4
4288	014364	004737	022642			JSR	PC,BUFREL		; RELEASE BUFFER
4289	014370	104401	056473			TYPE	OPRO15		; TYPE "ALIGNMENT PACT IN DRIVE"
4290	014374	004737	013004			JSR	PC,DROP		; GO DROP DRIVE
4291	014400	004737	022342			JSR	PC,GETBUF		; GO GET BUFFER FOR NEXT COMMAND
4292	014404	000207				RTS	PC		; RETURN
4293									
4294	014406	032765	000200	000014	35\$:	BIT	#W,WCK,P.PRST(R5)		; CHECK IF WRITE CHECK HAS BEEN ISSUED
4295	014414	001406				BEQ	38\$; YES, GO RELEASE BUFFER
4296	014416	042765	000200	000014		BIC	#W,WCK,P.PRST(R5)		; CLEAR WRITE BEFORE WRITE CHECK
4297	014424	000137	013662			JMP	ASN3\$; GO ISSUE COMMAND
4298									
4299	014430	012604			37\$:	MOV	(SP)+,R4		; RESTORE R4
4300	014432	004737	022642		38\$:	JSR	PC,BUFREL		; RELEASE BUFFER
4301	014436	032765	002000	000014		BIT	#DAPDRV,P.PRST(R5)		; CHECK IF DROP DRIVE
4302	014444	001402			40\$:	BEQ	40\$; NO, SEEK TO PREVIOUS POSITION
4303	014446	000137	016040			JMP	WV1\$; GO DROP DRIVE
4304									
4305	014452	112765	177777	000211	40\$:	MOVB	#-1,P.SEEK(R5)		; SET FLAG
4306	014460	016565	000136	000002		MOV	P.LCYL(R5),P.CYL(R5)		; LOAD LAST CYLINDER
4307	014466	016565	000140	000004		MOV	P.LSEC(R5),P.SECT(R5)		; LOAD LAST SECTOR AND TRACK
4308	014474	112765	000117	000001		MOVB	#SEEK,P.CMD(R5)		; ISSUE SEEK
4309	014502	000137	013662			JMP	ASN3\$; GO SEEK TO PREVIOUS POSITION
4310									
4311	014506	132765	00 10	000210	43\$:	BITB	#BIT3,P.ASSN(R5)		; CHECK IF READ PACK SERIAL NUMBER
4312	014514	001002				BNE	45\$; YES, SET UP FOR READ
4313	014516	000137	015234			JMP	WRTVER		; CHECK IF WRITE PACK
4314									
4315	014522	132765	000200	000210	45\$:	BITB	#BIT7,P.ASSN(R5)		; CHECK IF RECAL AND RETRY
4316									; READ OF PACK SERIAL NUMBER
4317	014530	001427				BEQ	48\$; NO, SERVICE READ OF PACK SERIAL NUMBER
4318	014532	142765	000200	000210		PICB	#BIT7,P.ASSN(R5)		; RESET RECAL FOR REREAD OF PACK SERIAL NO.
4319	014540	116565	000123	000135		MOVB	P.RCMD(R5),P.LCMD(R5)		; STORE LAST COMMAND
4320	014546	016565	000124	000136		MOV	P.RCYL(R5),P.LCYL(R5)		; STORE LAST CYLINDER
4321	014554	016565	000126	000140		MOV	P.RSEC(R5),P.LSEC(R5)		; STORE LAST SECTOR AND TRACK
4322	014562	016565	000130	000142		MOV	P.RWC(R5),P.LWC(R5)		; STORE LAST WORD COUNT
4323	014570	062765	000002	000126		ADD	#2,P.RSEC(R5)		; TRY NEXT 16-BIT FORMAT SECTOR
4324	014576	016565	000126	000004		MOV	P.RSEC(R5),P.SECT(R5)		; STORE SECTOR TO BE READ
4325	014604	000137	014064			JMP	20\$; GO ISSUE READ OF PACK SERIAL NUMBER
4326									
4327	014610	142765	000010	000210	48\$:	BICB	#BIT3,P.ASSN(R5)		; CLEAR FLAG
4328	014616	004037	025032			JSR	RO,CHKADD		; CHECK IF DATA TRANSFER CORRECT
4329	014622	015232				60\$; ERROR RETURN
4330	014624	010446				MOV	R4,-(SP)		; STORE R4
4331	014626	016504	000132			MOV	P.RBAL(R5),R4		; LOAD R4 TO ADDRESS SERIAL NUMBER
4332	014632	012465	000226			MOV	(R4)+,P.PKSR(R5)		; LOAD LEAST SIGNIFICANT BITS
4333									; OF SERIAL NUMBER
4334	014636	012465	000230			MOV	(R4)+,P.PKSR+2(R5)		; LOAD MOST SIGNIFICANT BITS
4335									; OF SERIAL NUMBER
4336	014642	005724				TST	(R4)+		; ADJUST R4 TO ADDRESS ALLIGNMENT INDICATION
4337	014644	005714				TST	(R4)		; CHECK IF ALLIGNMENT PACK
4338	014646	001245				BNE	30\$; YES, DROP DRIVE
4339	014650	012604				MOV	(SP)+,R4		; RESTORE R4
4340	014652	004737	022642			JSR	PC,BUFREL		; RELEASE BUFFER
4341	014656	132765	000020	000210		BITB	#BIT4,P.ASSN(R5)		; CHECK IF WRITE PACK

```

4342 014664 001042          BNE      55$      ;YES, SET UP PATTERN AND ALLOCATE BUFFER
4343 014666 105765 000076    TSTB     P.RATE(RS) ;CHECK IF READ ONLY MODE
4344 014672 001413          BEQ      52$      ;YES, START TESTING
4345 014674 032765 004000 000036    BIT      #WRL,P.DS(RS) ;CHECK IF WRITE LOCKED
4346 014702 001407          BEQ      52$      ;NO, START TESTING
4347
4348 014704 104401 056362      50$:     TYPE     OPRO12 ;TYPE "DRIVE WRITE LOCKED"
4349 014710 004737 013004      JSR      PC,DROP   ;DROP DRIVE FROM TEST SEQUENCE
4350 014714 004737 022342      JSR      PC,GETBUF ;GET BUFFER FOR NEXT COMMAND
4351 014720 000207          RTS       PC       ;RETURN
4352
4353 014722 116537 000000 056357 52$:     MOVB     P.DRVN(RS),OPRO11 ;LOAD DRIVE NUMBER
4354 014730 152737 000060 056357      BISB     #60,OPRO11 ;MAKE IT ASCII
4355 014736 104401 056340      TYPE     ,OPRO10 ;TYPE "DRIVE NUMBER N UNDER TEST"
4356 014742 104401 056103      TYPE     ,OPRO03
4357 014746 004737 027616      JSR      PC,PRISR  ;PRINT DRIVE AND PACK SERIAL NUMBERS
4358 014752 004737 043670      JSR      PC,PRITIM ;PRINT TIME
4359 014756 104401 001165      TYPE     $CRLF
4360 014762 004037 051406      JSR      RD,Q.PUSH ;PUT PARAMETER BLOCK IN DRIVE
4361 014766 001314          AVAILQ   ; AVAILIABLE QUEUE
4362 014770 000207          RTS       PC       ;RETURN
4363
4364          .SBTTL  WRITE WHOLE PACK WITH RANDOM DATA
4365
4366 014772 032765 004000 000036 55$:     BIT      #WRL,P.DS(RS) ;CHECK IF WRITE LOCKED
4367 015000 001341          BNE     50$      ;YES, DROP DRIVE
4368 015002 005037 177776      CLR      PS      ;ALLOW RK06 INTERRUPTS
4369 015006 116565 000123 000135      MOVB     P.RCMD(RS),P.LCMD(RS) ;STORE LAST COMMAND
4370 015014 016565 000124 000136      MOV      P.RCYL(RS),P.LCYL(RS) ;STORE LAST CYLINDER
4371 015022 016565 000126 000140      MOV      P.RSEC(RS),P.LSEC(RS) ;STORE LAST TRACK AND SECTOR
4372 015030 016565 000130 000142      MOV      P.RWC(RS),P.LWC(RS) ;STORE LAST WORD COUNT
4373 015036 112765 000131 000001      MOVB     #WATCHK,P.CMND(RS) ;LOAD WRITE COMMAND
4374 015044 112765 000131 000123      MOVB     #WATCHK,P.RCMD(RS)
4375 015052 052765 000200 000014      BIS     #W.WCK,P.PRST(RS) ;SET WRITE BEFORE WRITE CHECK
4376 015060 005065 000002      CLR      P.CYLN(RS) ;CLEAR CYLINDER ADDRESS
4377 015064 005065 000124      CLR      P.RCYL(RS)
4378 015070 005065 000004      CLR      P.SECT(RS) ;CLEAR TRACK AND SECTOR
4379 015074 005065 000126      CLR      P.RSEC(RS)
4380 015100 105065 000121      CLR      P.DPAT(RS) ;LOAD RANDOM PATTERN
4381 015104 013765 001354 000012      MOV      MAXBUF,P.WC(RS) ;LOAD WORD COUNT
4382 015112 042765 000377 000012      BIC     #377,P.WC(RS) ;MAKE IT AN EVEN NUMBER OF SECTORS
4383 015120 001003          BNE     57$      ;IF NOT ZERO USE THIS AS WORD COUNT
4384 015122 012765 000400 000012      MOV      #256,P.WC(RS) ;TRANSFER AT LEAST 256 WORDS
4385 015130 062765 000002 000152 57$:     ADD     #2,P.NORD(RS) ;INCREMENT NUMBER OF ORDERS
4386 015136 066565 000012 000156      ADD     P.WC(RS),P.NWRT(RS) ;ADD NUMBER OF WORDS WRITTEN
4387 015144 005565 000160      ADC     P.NWRT+2(RS)
4388 015150 066565 000012 000164      ADD     P.WC(RS),P.NRD(RS) ;ADD NUMBER OF WORDS READ
4389 015156 005565 000166      ADC     P.NRD+2(RS)
4390 015162 005465 000012          NEG     P.WC(RS) ;MAKE THE WORD COUNT NEGATIVE
4391 015166 013737 001452 177776      MOV      RKPRI,PS ;LOCK RK06 INTERRUPTS
4392 015174 005737 001320      TST     BWAITQ   ;CHECK IF COMMANDS WAITING FOR BUFFER
4393 015200 001011          BNE     59$      ;YES, ENQUEUE COMMAND IN BUFFER WAIT QUEUE
4394 015202 004037 022440      JSR      RD,BUFFAL ;GO ALLOCATE BUFFER
4395 015206 015224          59$:     ;NOT ENOUGH ROOM, ENQUEUE IN BUFFER WAIT QUEUE
4396 015210 005037 177776      CLP     PS      ;ALLOW RK06 INTERRUPTS
4397 015214 004737 023056      JSR      PC,LDDATA ;LOAD BUFFER

```

```

4398 015220 000137 013662          JMP      ASN3$          ;GO ISSUE COMMAND
4399
4400 015224 004037 051406          59$:   JSR      R0,Q.PUSH ;PUT PARAMETER BLOCK IN BUFFER
4401 015230 001320                BWAITQ                ;WAIT QUEUE
4402 015232 000207          60$:   RTS      PC          ;RETURN
4403
4404 015234 132765 000020 000210 WRTVER: BITB    #BIT4,P.ASSN(R5) ;CHECK IF WRITE PACK IN PROGRESS
4405 015242 001001                BNE                    ;YES, CONTINUE
4406 015244 000000                HALT                  ;*** PROGRAM PROBLEM
4407 015246 032765 000200 000014 1$:   BIT      #W.WCK,P.PRST(R5) ;CHECK IF WRITE CHECK ISSUED
4408 015254 001410                BEQ      WVD$         ;YES, CHECK FOR DROP DRIVE
4409 015256 042765 000200 000014        BIC      #W.WCK,P.PRST(R5) ;CLEAR ISSUE WRITE BEFORE WRITE CHECK
4410 015264 112765 000131 000001        MOVB    #WRTCHK,P.CMND(R5) ;RESTORE WRITE CHK CMD
4411 015272 000137 013662          JMP      ASN3$          ;GO ISSUE WRITE CHECK
4412
4413 015276 032765 002000 000014 WVD$:  BIT      #DRPDRV,P.PRST(R5) ;CHECK IF DROP DRIVE
4414 015304 001402                BEQ      10$         ;NO, CONTINUE
4415 015306 000137 016030                JMP      30$         ;YES, TERMINATE PACK WRITING
4416
4417 015312 132765 000040 000210 10$:  BITB    #BIT5,P.ASSN(R5) ;CHECK IF END OF PACK
4418 015320 001402                BEQ      15$         ;NO, CONTINUE
4419 015322 000137 016052                JMP      WV2$        ;YES, RETURN BUFFER
4420 015326 052765 000200 000014 15$:  BIS      #W.WCK,P.PRST(R5) ;SET ISSUE WRITE BEFORE WRT CHK
4421 015334 122765 000140 000001        CMPB    #RELEAS,P.CMND(R5) ;SEE IF LAST CMD WAS RELEASE
4422 015342 001452                BEQ      3$         ;BR IF YES, ELSE FILL PARAM BLOCK
4423 015344 112737 177777 001506        MOVB    #-1,I.ISRL ;INDICATE THAT AN INTERRUPT HAS BEEN ISSUED
4424 015352 012762 000300 000000        MOV     #INTR,RKCS1(R2) ;GO SCAN FOR DRIVE INTERRUPTS
4425 015360 005037 177776                CLR     PS           ;ALLOW RK06 INTERRUPTS
4426 015364 116565 000123 000135        MOVB    P.RCMD(R5),P.LCMD(R5) ;STORE PREVIOUS COMMAND
4427 015372 016565 000124 000136        MOV     P.RCYL(R5),P.LCYL(R5) ;STORE PREVIOUS CYLINDER
4428 015400 016565 000126 000140        MOV     P.RSEC(R5),P.LSEC(R5) ;STORE PREVIOUS TRACK AND SECTOR
4429 015406 116565 000122 000144        MOVB    P.RDPT(R5),P.LDPT(R5) ;STORE PREVIOUS DATA PATTERN
4430 015414 016565 000130 000142        MOV     P.RWC(R5),P.LWC(R5) ;STORE PREVIOUS WORD COUNT
4431 015422 016565 000026 000004        MOV     P.DTS(R5),P.SECT(R5) ;LOAD NEW SECTOR AND TRACK
4432 015430 016565 000026 000126        MOV     P.DTS(R5),P.RSEC(R5)
4433 015436 016565 000030 000002        MOV     P.DCYL(R5),P.CYLN(R5) ;LOAD NEW CYLINDER
4434 015444 016565 000030 000124        MOV     P.DCYL(R5),P.RCYL(R5)
4435 015452 016565 000132 000010        MOV     P.RBAL(R5),P.BALO(R5) ;REINITIALIZE BUS ADDRESS FOR POSSIBLE
4436                                     ;BAD SECTOR RECOVERY ON LAST COMMAND
4437 015460 122765 000131 000001        CMPB    #WRTCHK,P.CMND(R5) ;SEE IF LAST CMD WAS WRT CHK
4438 015466 001404                BEQ      4$         ;BR IF YES
4439 015470 112765 000131 000001 3$:   MOVB    #WRTCHK,P.CMND(R5) ;ELSE SET TO DO WRT DATA (W.WCK=1)
4440 015476 000410                BR      5$
4441 015500 042765 000200 000014 4$:   BIC      #W.WCK,P.PRST(R5) ;RELEAS AFTER WRT CHK
4442 015506 112765 000140 000001        MOVB    #RELEAS,P.CMND(R5) ;RELEAS AFTER WRT CHK
4443 015514 000137 013662          JMP      ASN3$          ;GO RELEASE
4444 015520 016546 000002          5$:   MOV     P.CYLN(R5),-(SP) ;STORE CYLINDER ADDRESS ON STACK
4445 015524 006316                ASL    (SP)          ;MULTIPLY CYLINDER BY 3
4446 015526 066516 000002                ADD    P.CYLN(R5),(SP)
4447 015532 005046                CLR    -(SP)        ;MAKE ROOM ON STACK FOR TRACK
4448 015534 116516 000005        MOVB    P.TRCK(R5),(SP) ;STORE TRACK ON STACK
4449 015540 061666                ADD    (SP),2(SP)   ;ADD TRACK
4450 015544 012716 000026        MOV     #22,(SP)    ;STORE 22 ON STACK (MULTIPLIER)
4451 015550 004737 054714        JSR     PC,$MULT    ;CALCULATE NUMBER OF SECTORS TRANSFERRED
4452 015554 012616                MOV     (SP)+,(SP)  ;THROW AWAY MOST SIGNIFICANT BITS
4453 015556 005046                CLR    -(SP)        ;CLEAR LOCATION FOR SECTOR

```

H08

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 100
WRITE WHOLE PACK WITH RANDOM DATA

SEQ 0098

4454	015560	116516	000004		MOV B	P. SECT(R5), (SP)	; STORE SECTOR COUNT
4455	015564	061666	000002		ADD	(SP), 2(SP)	; DETERMINE TOTAL NUMBER OF SECTORS TRANSFERRED
4456	015570	004737	027074		JSR	PC, GETNUM	; GET CORRECT LAST CYL NUM
4457	015574	013716	027160		MOV	HOLD, (SP)	; STORE NUMBER OF SECTORS ON STACK
4458	015600	166616	000002		SUB	2(SP), (SP)	; DETERMINE NUMBER OF SECTORS LEFT ON ON PACK TO BE WRITTEN
4459							
4460	015604	016546	000130		MOV	P. RWC(R5), -(SP)	; STORE WORD COUNT
4461	015610	011665	000012		MOV	(SP), P.WC(R5)	
4462	015614	005416			NEG	(SP)	; MAKE IT POSITIVE
4463	015616	116616	000001		MOV B	1(SP), (SP)	; KEEP SECTOR COUNT
4464	015622	105066	000001		CLRB	1(SP)	; CLEAR MOST SIGNIFICANT BITS
4465	015626	021666	000002		CMP	(SP), 2(SP)	; CHECK IF PACK OVERRUN
4466	015632	103413			BLO	20\$; NO, GO ISSUE COMMAND
4467	015634	001407			BEQ	19\$; CHECK IF NO NEW WORD COUNT NEEDED
4468	015636	005016			CLR	(SP)	; MAKE ROOM FOR NEW WORD COUNT
4469	015640	116666	000002	000001	MOV B	2(SP), 1(SP)	; MOVE IN NUMBER OF SECTORS LEFT
4470	015646	005416			NEG	(SP)	; MAKE WORD COUNT NEGATIVE
4471	015650	011665	000012		MOV	(SP), P.WC(R5)	; LOAD NEW WORD COUNT
4472	015654	152765	000040	000210	19\$: BISB	#BITS, P.ASSN(R5)	; SET END OF PACK TRANSFER
4473							
4474	015662	062706	000006		20\$: ADD	#6, SP	; ADJUST STACK
4475	015666	016546	000012		MOV	P.WC(R5), -(SP)	; STORE PRESENT WORD COUNT
4476	015672	005416			NEG	(SP)	; MAKE IT POSITIVE
4477	015674	062765	000002	000152	ADD	#2, P.NODR(R5)	; INCREMENT NUMBER OF ORDERS
4478	015702	061665	000156		ADD	(SP), P.NWRT(R5)	; ADD NUMBER OF WORDS WRITTEN
4479	015706	005765	000160		ADC	P.NWRT+2(R5)	
4480	015712	062665	000164		ADD	(SP)+, P.NRD(R5)	; ADD NUMBER OF WORDS READ
4481	015716	005665	000166		ADC	P.NRD+2(R5)	
4482	015722	105265	000122		INCB	P.RDPT(R5)	; GENERATE NEXT DATA PATTERN TO BE WRITTEN
4483	015726	142765	000360	000122	BICB	#360, P.RDPT(R5)	; KEEP 4 LEAST SIGNIFICANT BITS
4484	015734	116565	000122	000121	MOV B	P.RDPT(R5), P.DPAT(R5)	; STORE DATA PATTERN USED
4485	015742	013737	001452	177776	MOV	RKPRI, PS	; LOCK OUT RK06 INTERRUPTS
4486	015750	132765	000040	000210	BITB	#BITS, P.ASSN(R5)	; CHECK IF WRITE TO END OF PACK
4487	015756	001014			BNE	25\$; YES, RELEASE BUFFER
4488	015760	005737	001320		TST	BWAITQ	; CHECK IF BUFFER WAIT QUEUE EMPTY
4489	015764	001011			BNE	25\$; NO, RELEASE BUFFER
4490	015766	016565	000012	000130	MOV	P.WC(R5), P.RWC(R5)	; STORE CURRENT GENERATED WORD COUNT
4491	015774	005037	177776		CLR	PS	; ALLOW RK06 INTERRUPTS
4492	016000	004737	023056		JSR	PC, LDDATA	; LOAD NEW RANDOM DATA
4493	016004	000137	013662		JMP	ASN3\$; GO ISSUE COMMAND
4494							
4495	016010	004737	022642		25\$: JSR	PC, BUFREL	; RELEASE BUFFER FOR OTHER DRIVES
4496	016014	004037	051406		JSR	RO, Q.PUSH	; PUT PARAMETER BLOCK ON BUFFER WAIT QUEUE
4497	016020	001320			BWAITQ		
4498	016022	004737	022342		JSR	PC, GETBUF	; GET NEW BUFFER
4499	016026	000207			RTS	PC	; RETURN
4500							
4501	016030	105065	000210		30\$: CLRB	P.ASSN(R5)	; CLEAR ASSIGNMENT CODE
4502	016034	004737	022642		JSR	PC, BUFREL	; RELEASE BUFFER
4503	016040	104401	056260		WV1\$: TYPE	, OPRO08	; TYPE "OPERATOR INITIATED DROP DRIVE"
4504	016044	004737	013004		JSR	PC, DROP	; GO DROP DRIVE
4505	016050	000444			BF	WV5\$; GO ALLOCATE BUFFER
4506							
4507	016052	105065	000210		WV2\$: CLRB	P.ASSN(R5)	; CLEAR ASSIGNMENT FLAGS
4508	016056	116537	000000	056357	MOV B	P.DRVN(R5), OPRO11	; GET DRIVE NUMBER
4509	016064	152737	000060	056357	BISB	#60, OPRO11	; MAKE IT ASCII

4510	016072	104401	056340		TYPE	,OPR010		;TYPE "DRIVE NUMBER N UNDER TEST"
4511	016076	104401	056103		TYPE	,OPR003		
4512	016102	004737	027616		JSR	PC,PRISER		;PRINT DRIVE AND PACK SERIAL NUMBERS
4513	016106	004737	043670		JSR	PC,PRITIM		;PRINT TIME
4514	016112	104401	001165		TYPE	,SCLF		
4515	016116	010346			MOV	R3,-(SP)		;STORE R3 ON STACK
4516	016120	010446			MOV	R4,-(SP)		;STORE R4 ON STACK
4517	016122	010503			MOV	R5,R3		;LOAD PARAMETER BLOCK BASE
4518	016124	062703	000152		ADD	#P.NODR,R3		;CALCULATE BEGINNING OF STATISTICS
4519	016130	010504			MOV	R5,R4		;LOAD PARAMETER BLOCK BASE
4520	016132	062704	000210		ADD	#P.ASSN,R4		;CALCULATE END OF STATISTICS
4521	016136	005023		7\$:	CLR	(R3)+		;CLEAR STATISTICS BEFORE STARTING TEST
4522	016140	020304			CMP	R3,R4		;CHECK IF FINISHED
4523	016142	001375			BNE	7\$;NO CONTINUE
4524	016144	012604			MOV	(SP)+,R4		;RESTORE R4
4525	016146	012603			MOV	(SP)+,R3		;RESTORE R3
4526	016150	004737	022642		JSR	PC,BUFREL		;RELEASE BUFFER
4527	016154	004037	051406		JSR	RD,Q.PUSH		;ENQUEUE PARAMETER BLOCK IN DR...L
4528	016160	001314			AVAILQ			; AVAILIABLE QUEUE
4529	016162	004737	022342	WV5\$:	JSR	PC,GETBUF		;GET NEW BUFFER
4530	016166	000207		WV9\$:	RTS	PC		;RETURN

```

4531          .SBTTL  ALTER DRIVE PARAMETERS
4532
4533 016170 010446          PARM:  MOV      R4,-(SP)      ;STORE R4 ON STACK
4534 016172 010346          MOV      R3,-(SP)      ;STORE R3 ON STACK
4535 016174 116537 000000 057533 POS:  MOVB    P.DRVN(R5),PAR001 ;LOAD DRIVE NUMBER FOR PRINT OUT
4536 016202 152737 000060 057533      BISB    #60,PAR001    ;MAKE IT ASCII
4537 016210 104401 057507      TYPE   PAR000        ;TYPE "PARAMETERS FOR DRIVE N"
4538 016214 004737 020410      JSR    PC,DTYPE     ;INPUT DRIVE TYPE
4539 016220 104401 057540      2$:   TYPE   PAR002        ;TYPE "CYLINDER MAX,MIN"
4540 016224 004037 020340      JSR    R0,TSTDEF   ;CHECK FOR DEFAULT PARAMETERS
4541 016230 016346          5$     ;COMMA DETECTED
4542 016232 016362          6$     ;CARRIAGE RETURN DETECTED
4543 016234 020200          P1$    ;CONTROL Z (<↑Z>) DETECTED
4544 016236 016174          POS    ;CONTROL C (<↑C>) DETECTED
4545 016240 010346          MOV      R3,-(SP)      ;STORE BUFFER ADDRESS
4546 016242 004737 054146      JSR    PC,OCTBIN    ;CONVERT TO BINARY
4547 016246 016330          4$     ;ERROR RETURN
4548 016250 012637 001602      MOV      (SP)+,PARCYL ;STORE MAX CYLINDER ENTRY
4549 016254 004737 027074      JSR    PC,GETNUM    ;GET CORRECT LAST CYL NUM
4550 016260 023737 027154 001602 CMP     LSTCYL,PARCYL ;CHECK IF LESS OR EQUAL TO
4551          5$     ;MAXIMUM CYLINDER ADDRESS
4552 016266 103420          BLO    4$           ;NO, TRY AGAIN
4553 016270 010304          MOV     R3,R4        ;STORE R3 IN R4 FOR LINE SCAN
4554 016272 105714          64$:  TSTB   (R4)        ;CHECK FOR CARRIAGE RETURN
4555 016274 001435          BEQ    7$           ;YES, EXIT
4556 016276 122427 000054      CMPB   (R4)+,#'     ;CHECK FOR COMMA
4557 016302 001373          BNE    64$         ;NO, GO TEST NEXT CHARACTER
4558 016304 010446          3$:   MOV     R4,-(SP)    ;STORE BUFFER ADDRESS
4559 016306 004737 054146      JSR    PC,OCTBIN    ;CONVERT TO BINARY
4560 016312 016330          4$     ;ERROR RETURN
4561 016314 012637 001604      MOV     (SP)+,PARCYL+2 ;STORE MIN CYLINDER ENTRY
4562 016320 023737 001602 001604 CMP     PARCYL,PARCYL+2 ;CHECK IF LEGAL ENTRY
4563 016326 103022          BHS    8$           ;YES, LOAD MAX AND MIN CYLINDER
4564 016330          4$:   MOV     R3,65$     ;LOAD BUFFER ADDRESS
4565 016330 010337 016336          TYPE   R3,65$     ;TYPE RECEIVED INPUT
4566 016334 104401          65$:  .WORD  0           ;BUFFER ADDRESS
4567 016336 000000          TYPE   ,#QUES     ;TYPE QUESTION MARK
4568 016340 104401 001164      BR     2$           ;TRY AGAIN
4569 016344 000725
4570
4571 016346 013737 027154 001602 5$:   MOV     LSTCYL,PARCYL ;LOAD MAX CYLINDER = 632/1456
4572 016354 010304          MOV     R3,R4        ;STORE R3 IN R4 FOR LINE SCAN
4573 016356 105724          TSTB   (R4)+        ;ADJUST R4
4574 016360 000751          BR     3$           ;GET MINIMUM CYLINDER
4575
4576 016362 013737 027154 001602 6$:   MOV     LSTCYL,PARCYL ;LOAD MAX CYLINDER = 632/1456
4577 016370 005037 001604          CLR    PARCYL+2     ;LOAD MIN CYLINDER = 0
4578 016374 013765 001602 000070 7$:   MOV     PARCYL,P.MXCL(R5) ;LOAD MAX CYLINDER
4579 016402 013765 001604 000066 8$:   MOV     PARCYL+2,P.MNCL(R5) ;LOAD MIN CYLINDER
4580
4581 016410 104401 057563          10$:  TYPE   PAR003        ;TYPE "TRACK MAX,MIN"
4582 016414 004037 020340      JSR    R0,TSTDEF   ;CHECK FOR DEFAULT PARAMETERS
4583 016420 016532          13$    ;COMMA DETECTED
4584 016422 016546          14$    ;CARRIAGE RETURN DETECTED
4585 016424 020212          P2$    ;CONTROL Z (<↑Z>) DETECTED
4586 016426 016174          POS    ;CONTROL C (<↑C>) DETECTED

```

4587	016430	010346				MOV	R3, -(SP)	: STORE BUFFER ADDRESS
4588	016432	004737	054146			JSR	PC, OCTBIN	: CONVERT TO BINARY
4589	016436	016514				12\$: ERROR RETURN
4590	016440	012637	001606			MOV	(SP)+, PARTRK	: STORE MAX TRACK ENTRY
4591	016444	022737	000002	001606		CMP	#D.MXTR, PARTRK	: CHECK IF LESS OR EQUAL TO
4592								: MAXIMUM TRACK ADDRESS
4593	016452	103420				BLO	12\$: NO, TRY AGAIN
4594	016454	010304				MOV	R3, R4	: STORE R3 IN R4 FOR LINE SCAN
4595	016456	105714			66\$:	TSTB	(R4)	: CHECK FOR CARRIAGE RETURN
4596	016460	001435				BEQ	15\$: YES, EXIT
4597	016462	122427	000054			CMPB	(R4)+, #'	: CHECK FOR COMMA
4598	016466	001373				BNE	66\$: NO, GO TEST NEXT CHARACTER
4599	016470	010446			11\$:	MOV	R4, -(SP)	: STORE BUFFER ADDRESS
4600	016472	004737	054146			JSR	PC, OCTBIN	: CONVERT TO BINARY
4601	016476	016514				12\$: ERROR RETURN
4602	016500	012637	001610			MOV	(SP)+, PARTRK+2	: STORE MIN TRACK ENTRY
4603	016504	023737	001606	001610		CMP	PARTRK, PARTRK+2	: CHECK IF LEGAL ENTRY
4604	016512	103022				BHIS	16\$: YES, LOAD MAX AND MIN TRACK
4605	016514				12\$:			
4606	016514	010337	016522			MOV	R3, 67\$: LOAD BUFFER ADDRESS
4607	016520	104401				TYPE		: TYPE RECEIVED INPUT
4608	016522	000000			67\$:	.WORD	0	: BUFFER ADDRESS
4609	016524	104401	001164			TYPE	\$QUES	: TYPE QUESTION MARK
4610	016530	000727				BR	10\$: TRY AGAIN
4611								
4612	016532	012737	000002	001606	13\$:	MOV	#D.MXTR, PARTRK	: LOAD MAX TRACK = 2
4613	016540	010304				MOV	R3, R4	: STORE R3 IN R4 FOR LINE SCAN
4614	016542	105724				TSTB	(R4)+	: ADJUST R4
4615	016544	000751				BR	11\$: GET MINIMUM TRACK
4616								
4617	016546	012737	000002	001606	14\$:	MOV	#D.MXTR, PARTRK	: LOAD MAX TRACK = 2
4618	016554	005037	001610		15\$:	CLR	PARTRK+2	: LOAD MIN TRACK = 0
4619	016560	113765	001606	000073	16\$:	MOV	PARTRK, P.MXTR(R5)	: LOAD MAX TRACK
4620	016566	113765	001610	000072		MOV	PARTRK+2, P.MNTR(R5)	: LOAD MIN TRACK
4621								
4622	016574	104401	057603		18\$:	TYPE	PAR004	: TYPE "SECTOR MAX, MIN"
4623	016600	004037	020340			JSR	RD, TSTDEF	: CHECK FOR DEFAULT PARAMETERS
4624	016604	016716				21\$: COMMA DETECTED
4625	016606	016732				22\$: CARRIAGE RETURN DETECTED
4626	016610	020224				P3\$: CONTROL Z (<Z>) DETECTED
4627	016612	016174				POS		: CONTROL C (<C>) DETECTED
4628	016614	010346				MOV	R3, -(SP)	: STORE BUFFER ADDRESS
4629	016616	004737	054146			JSR	PC, OCTBIN	: CONVERT TO BINARY
4630	016622	016700				20\$: ERROR RETURN
4631	016624	012637	001612			MOV	(SP)+, PARSEC	: STORE MAX SECTOR ENTRY
4632	016630	022737	000025	001612		CMP	#D.MXSC, PARSEC	: CHECK IF LESS OR EQUAL TO
4633								: MAXIMUM SECTOR ADDRESS
4634	016636	103420				BLO	20\$: NO, TRY AGAIN
4635	016640	010304				MOV	R3, R4	: STORE R3 IN R4 FOR LINE SCAN
4636	016642	105714			68\$:	TSTB	(R4)	: CHECK FOR CARRIAGE RETURN
4637	016644	001435				BEQ	23\$: YES, EXIT
4638	016646	122427	000054			CMPB	(R4)+, #'	: CHECK FOR COMMA
4639	016652	001373				BNE	68\$: NO, GO TEST NEXT CHARACTER
4640	016654	010446			19\$:	MOV	R4, -(SP)	: STORE BUFFER ADDRESS
4641	016656	004737	054146			JSR	PC, OCTBIN	: CONVERT TO BINARY
4642	016662	016700				20\$: ERROR RETURN

```

4643 016664 012637 001614      MOV      (SP)+, PARSEC+2 ;STORE MIN SECTOR ENTRY
4644 016670 023737 001612 001614  CMP      PARSEC, PARSEC+2 ;CHECK IF LEGAL ENTRY
4645 016676 103022      BHIS     24$ ;YES, LOAD MAX AND MIN SECTOR
4646 016700      20$:
4647 016700 010337 016706      MOV      R3, 69$ ;LOAD BUFFER ADDRESS
4648 016704 104401      TYPE     ;TYPE RECEIVED INPUT
4649 016706 000000      .WORD   0 ;BUFFER ADDRESS
4650 016710 104401 001164      TYPE     $QUES ;TYPE QUESTION MARK
4651 016714 000727      BR       18$ ;TRY AGAIN
4652
4653 016716 012737 000025 001612 21$: MOV      #D.MXSC, PARSEC ;LOAD MAX SECTOR = 21
4654 016724 010304      MOV      R3, R4 ;STORE R3 IN R4 FOR LINE SCAN
4655 016726 105724      TSTB    (R4)+ ;ADJUST R4
4656 016730 000751      BR       19$ ;GET MINIMUM SECTOR
4657
4658 016732 012737 000025 001612 22$: MOV      #D.MXSC, PARSEC ;LOAD MAX SECTOR = 21
4659 016740 005037 001614      CLR      PARSEC+2 ;LOAD MIN SECTOR = 0
4660 016744 113765 001612 000075 23$: MOVVB   PARSEC, P.MXSC(R5) ;LOAD MAX SECTOR
4661 016752 113765 001614 000074 24$: MOVVB   PARSEC+2, P.MNSC(R5) ;LOAD MIN SECTOR
4662
4663 016760 104401 057624      TYPE     PARO05 ;TYPE "READ/WRITE RATIO ="
4664 016764 004037 020340      JSR      R0, TSTDEF ;CHECK FOR DEFAULT PARAMETERS
4665 016770 017020      27$:
4666 016772 017044      29$:
4667 016774 020236      P4$
4668 016776 016174      P0$
4669 017000 010346      MOV      R3, -(SP) ;STORE BUFFER ADDRESS
4670 017002 004737 054146      JSR      PC, OCTBIN ;CONVERT BINARY
4671 017006 017020      27$:
4672 017010 012604      MOV      (SP)+, R4 ;STORE INPUT
4673 017012 022704 000010      CMP      #10, R4 ;CHECK IF 0-7
4674 017016 101007      BHI     28$ ;YES, LOAD READ/WRITE RATIO
4675 017020      27$:
4676 017020 010337 017026      MOV      R3, 70$ ;LOAD BUFFER ADDRESS
4677 017024 104401      TYPE     ;TYPE RECEIVED INPUT
4678 017026 000000      .WORD   0 ;BUFFER ADDRESS
4679 017030 104401 001164      TYPE     $QUES ;TYPE QUESTION MARK
4680 017034 000751      BR       26$ ;TRY AGAIN
4681
4682 017036 110465 000076      MOVVB   R4, P.RATE(R5) ;LOAD READ/WRITE RATIO
4683 017042 000403      BR       30$ ;ASK "WRITE CHECK AFTER WRITE ="
4684
4685 017044 112765 000003 000076 29$: MOVVB   #D.RATE, P.RATE(R5) ;LOAD DEFAULT READ/WRITE
4686
4687 017052 104401 057647      TYPE     PARO06 ;TYPE "WRITE CHECK AFTER WRITE ="
4688 017056 004037 020340      JSR      R0, TSTDEF ;CHECK FOR DEFAULT PARAMETERS
4689 017062 017110      31$:
4690 017064 017126      32$:
4691 017066 020244      P5$
4692 017070 016174      P0$
4693 017072 122713 000131      CMPB    #'Y', (R3) ;CHECK IF YES
4694 017076 001013      BNE     32$ ;NO, LOAD DEFAULT
4695 017100 112765 177777 000077      MOVVB   #-1, P.AWCK(R5) ;SET WRITE CHECK AFTER WRITE
4696 017106 000411      BR       34$ ;GET CORRECTABLE READ ERROR
4697 ; THRESHOLD
4698

```


4699	017110				31\$:	MOV	R3,71\$;LOAD BUFFER ADDRESS
4700	017110	010337	017116			TYPE		;TYPE RECEIVED INPUT
4701	017114	104401			71\$:	.WORD	0	;BUFFER ADDRESS
4702	017116	000000				TYPE	\$QUES	;TYPE QUESTION MARK
4703	017120	104401	001164			BR	3C\$;TRY AGAIN
4704	017124	000752						
4705								
4706	017126	105065	000077		32\$:	CLRB	P.AWCK(R5)	;CLEAR WRITE CHECK AFTER WRITE
4707								
4708	017132	104401	057701		34\$:	TYPE	,PAR007	;TYPE "CORRECTABLE READ ERROR
4709								;THRESHOLD
4710	017136	004037	020340			JSR	RO,TSTDEF	;CHECK FOR DEFAULT PARAMETERS
4711	017142	017170				35\$;COMMA DETECTED
4712	017144	017206				36\$;CARRIAGE RETURN DETECTED
4713	017146	020250				P6\$;CONTROL Z (<Z>) DETECTED
4714	017150	016174				POS		;CONTROL C (<C>) DETECTED
4715	017152	010346				MOV	R3,-(SP)	;STORE BUFFER ADDRESS
4716	017154	004737	054302			JSR	PC,DECBIN	;CONVERT TO BINARY
4717	017160	017170				35\$;ERROR RETURN
4718	017162	012665	000100			MOV	(SP)+,P.CERT(R5)	;LOAD CORRECTABLE READ ERROR
4719								;THRESHOLD
4720	017166	100012				APL	38\$;IF NOT NEGATIVE, GET UNCORRECTABLE
4721								;READ ERROR THRESHOLD
4722								
4723	017170				35\$:	MOV	R3,72\$;LOAD BUFFER ADDRESS
4724	017170	010337	017176			TYPE		;TYPE RECEIVED INPUT
4725	017174	104401			72\$:	.WORD	0	;BUFFER ADDRESS
4726	017176	000000				TYPE	\$QUES	;TYPE QUESTION MARK
4727	017200	104401	001164			BR	34\$;TRY AGAIN
4728	017204	000752						
4729								
4730	017206	012765	000012	000100	36\$:	MOV	#D.CERT,P.CERT(R5)	;LOAD DEFAULT CORRECTABLE READ
4731								;ERROR THRESHOLD
4732								
4733	017214	104401	057744		38\$:	TYPE	,PAR008	;TYPE "UNCORRECTABLE READ ERROR
4734								;THRESHOLD ="
4735	017220	004037	020340			JSR	RO,TSTDEF	;CHECK FOR DEFAULT PARAMETERS
4736	017224	017252				39\$;COMMA DETECTED
4737	017226	017270				40\$;CARRIAGE RETURN DETECTED
4738	017230	020256				P7\$;CONTROL Z (<Z>) DETECTED
4739	017232	016174				POS		;CONTROL C (<C>) DETECTED
4740	017234	010346				MOV	R3,-(SP)	;STORE BUFFER ADDRESS
4741	017236	004737	054302			JSR	PC,DECBIN	;CONVERT TO BINARY
4742	017242	017252				39\$;ERROR RETURN
4743	017244	012665	000102			MOV	(SP)+,P.UERT(R5)	;LOAD UNCORRECTABLE READ ERROR
4744								;THRESHOLD
4745	017250	100012				BPL	42\$;IF NOT NEGATIVE, GET SEEK
4746								;ERROR THRESHOLD
4747	017252				39\$:	MOV	R3,73\$;LOAD BUFFER ADDRESS
4748	017252	010337	017260			TYPE		;TYPE RECEIVED INPUT
4749	017256	104401			73\$:	.WORD	0	;BUFFER ADDRESS
4750	017260	000000				TYPE	\$QUES	;TYPE QUESTION MARK
4751	017262	104401	001164			BR	38\$;TRY AGAIN
4752	017266	000752						
4753								
4754	017270	012765	000005	000102	40\$:	MOV	#D.UERT,P.UERT(R5)	;LOAD DEFAULT UNCORRECTABLE READ

```
4755 ; ERROR THRESHOLD
4756
4757 017276 104401 060011 42$: TYPE PAR009 ; TYPE "SEEK ERROR THRESHOLD ="
4758 017302 004037 020340 JSR RO,STDEF ; CHECK FOR DEFAULT PARAMETERS
4759 017306 017334 43$ ; COMMA DETECTED
4760 017310 017352 44$ ; CARRIAGE RETURN DETECTED
4761 017312 020264 P8$ ; CONTROL Z (<↑Z>) DETECTED
4762 017314 016174 P0$ ; CONTROL C (<↑C>) DETECTED
4763 017316 010346 MOV R3,-(SP) ; STORE BUFFER ADDR SS
4764 017320 004737 054302 JSR PC,DECBIN ; CONVERT TO BINARY
4765 017324 017334 43$ ; ERROR RETURN
4766 017326 012665 000104 MOV (SP)+,P.SKET(R5) ; LOAD SEEK ERROR THRESHOLD
4767 017332 100012 BPL 46$ ; CHECK IF INHIBIT ERROR COPRECTION
4768 ; AND RETRY
4769
4770 017334 010337 017342 43$: MOV R3,74$ ; LOAD BUFFER ADDRESS
4771 017340 104401 TYPE ; TYPE RECEIVED INPUT
4772 017342 000000 74$: .WORD 0 ; BUFFER ADDRESS
4773 017344 104401 001164 TYPE $QUES ; TYPE QUESTION MARK
4774 017350 000752 BR 42$ ; TRY AGAIN
4775
4776 017352 012765 000005 000104 44$: MOV #D.SKET,P.SKET(R5) ; LOAD DEFAULT SEEK ERROR THRESHOLD
4777
4778 017360 104401 060040 46$: TYPE ,PAR010 ; TYPE "INHIBIT ERROR CORREC ION
4779 ; AND RETRY
4780 017364 004037 020340 JSR RO,TSTDEF ; CHECK FOR DEFAULT PARAMETERS
4781 017370 017416 47$ ; COMMA DETECTED
4782 017372 017434 48$ ; CARRIAGE RETURN DETECTED
4783 017374 020272 P9$ ; CONTROL Z (<↑Z>) DETECTED
4784 017376 016174 P0$ ; CONTROL C (<↑C>) DETECTED
4785 017400 122713 000131 CMPB #'Y,(R3) ; CHECK IF YES
4786 017404 001013 BNE 48$ ; LOAD DEFAULT
4787 017406 112765 177777 000120 MOVB #-1,P.IERC(R5) ; SET INHIBIT ERROR CORRECTION
4788 017414 000411 BR 50$ ; GET COMMAND COUNT THRESHOLD
4789
4790 017416 47$: MOV R3,75$ ; LOAD BUFFER ADDRESS
4791 017416 010337 017424 TYPE ; TYPE RECEIVED INPUT
4792 017422 104401 75$: .WORD 0 ; BUFFER ADDRESS
4793 017424 000000 TYPE $QUES ; TYPE QUESTION MARK
4794 017426 104401 001164 BR 46$ ; TRY AGAIN
4795 017432 000752
4796
4797 017434 105065 000120 48$: CLRB P.IERC(R5) ; CLEAR INHIBIT ERROR CORRECTION
4798
4799 017440 104401 060105 50$: TYPE PAR011 ; TYPE "OPERATION COUNT THRESHOLD*65K ="
4800 017444 004037 020340 JSR RO,TSTDEF ; CHECK FOR DEFAULT PARAMETERS
4801 017450 017514 51$ ; COMMA DETECTED
4802 017452 017532 52$ ; CARRIAGE RETURN DETECTED
4803 017454 020273 P10$ ; CONTROL Z (<↑Z>) DETECTED
4804 017456 016174 P0$ ; CONTROL C (<↑C>) DETECTED
4805 017460 010346 MOV R3,-(SP) ; STORE BUFFER ADDRESS
4806 017462 004737 054302 JSR PC,DECBIN ; CONVERT TO BINARY
4807 017466 017514 51$ ; ERROR RETURN
4808 017470 012665 000110 MOV (SP)+,P.MXCD+2(R5) ; STORE NUMBER OF 65K BLOCKS
4809 017474 001407 BEQ 51$ ; IF ZERO, TRY AGAIN
4810 017476 100406 BMI 51$ ; IF MINUS, TRY AGAIN
```

```

4811 017500 005365 000110          DEC      P.MXCD+2(R5)      ;DECREMENT BY 1
4812 017504 012765 177777 000106  MOV      #-1,P.MXCD(R5)
4813 017512 000415          BR       54$           ;GET WORD TRANSFERREC THRESHOLD
4814
4815 017514          51$:
4816 017514 010337 017522          MOV      R3,76$      ;LOAD BUFFER ADDRESS
4817 017520 104401          TYPE    ;TYPE RECEIVED INPUT
4818 017522 000000 76$:          .WORD   0           ;BUFFER ADDRESS
4819 017524 104401 001164          TYPE    $QUES       ;TYPE QUESTION MARK
4820 017530 000743          BR       50$         ;TRY AGAIN
4821
4822 017532 012765 077777 000110 52$:          MOV      #D.CTRH,P.MXCD+2(R5) ;LOAD DEFAULT COMMAND THRESHOLD
4823 017540 012765 177770 000106  MOV      #D.CTRL,P.MXCD(R5)
4824
4825 017546 104401 060145 54$:          TYPE    PAR012      ;TYPE "WORD TRANSFERRED THRESHOLD*520K ="
4826 017552 004037 020340          JSR     R0,TSTDEF   ;CHECK FOR DEFAULT PARAMETERS
4827 017556 017654          55$
4828 017560 017672          56$
4829 017562 020312          P11$
4830 017564 016174          POS
4831 017566 010346          MOV     R3,-(SP)    ;STORE BUFFER ADDRESS
4832 017570 004737 054302          JSR     PC,DECBIN   ;CONVERT TO BINARY
4833 017574 017654          55$
4834 017576 012665 000112          MOV     (SP)+,P.MXWT(R5) ;STORE NUMBER OF 520K BLOCKS
4835 017602 001424          BEQ     53$         ;IF ZERO, TRY AGAIN
4836 017604 100423          BMI     55$         ;IF NEGATIVE, TRY AGAIN
4837 017606 005065 000114          CLR     P.MXWT+2(R5) ;CLEAR HIGH ORDER BITS
4838 017612 006165 000112          ROL     P.MXWT(R5)   ;SHIFT 4 BITS LEFT
4839 017616 006165 000114          ROL     P.MXWT+2(R5)
4840 017622 006165 000112          ROL     P.MXWT(R5)
4841 017626 006165 000114          ROL     P.MXWT+2(R5)
4842 017632 006165 000112          ROL     P.MXWT(R5)
4843 017636 006165 000114          ROL     P.MXWT+2(R5)
4844 017642 006165 000112          ROL     P.MXWT(R5)
4845 017646 006165 000114          ROL     P.MXWT+2(R5)
4846 017652 000415          BR       60$         ;CHECK FOR SAMPLED COMPARES
4847
4848 017654          55$:
4849 017654 010337 017662          MOV     R3,77$      ;LOAD BUFFER ADDRESS
4850 017660 104401          TYPE    ;TYPE RECEIVED INPUT
4851 017662 000000 77$:          .WORD   0           ;BUFFER ADDRESS
4852 017664 104401 001164          TYPE    $QUES       ;TYPE QUESTION MARK
4853 017670 000726          BR       54$         ;TRY AGAIN
4854
4855 017672 012765 077777 000114 56$:          MOV     #D.WTHI,P.MXWT+2(R5) ;LOAD DEFAULT WORD TRANSFER
4856 017700 012765 177770 000112  MOV     #D.WTLO,P.MXWT(R5)   ; THRESHOLD
4857
4858 017706 005737 001362 60$:          TST     SOFCMP      ;CHECK IF ANY SOFTWARE COMPARES ARE TO BE MADE
4859 017712 001430          BEQ     PREXAR      ;NO, DETERMINE TRACK EXCLUSION
4860 017714 104401 060443          TYPE    PAR021      ;TYPE "SAMPLED COMPARES ="
4861 017720 004037 020340          JSR     R0,TSTDEF   ;CHECK FOR DEFAULT PARAMETERS
4862 017724 017752          61$
4863 017726 017770          62$
4864 017730 020326          P12$
4865 017732 016174          POS
4866 017734 122713 000116          CMPB   #'N,(R3)    ;CHECK IF NO

```

4867	017740	001013		BNE	62\$; NO, MAKE SAMPLED COMPARES
4868	017742	112765	177777	MOV	#-1,P	SMPL(R5)	; DO NOT MAKE SAMPLED COMPARES
4869	017750	000411	000116	BR	PREXAR		; DETERMINE TRACK EXCLUSION
4870							
4871	017752			61\$:	MOV	R3,78\$; LOAD BUFFER ADDRESS
4872	017752	010337	017760		TYPE		; TYPE RECEIVED INPUT
4873	017756	104401		78\$:	.WORD	0	; BUFFER ADDRESS
4874	017760	000000			TYPE	\$QUES	; TYPE QUESTION MARK
4875	017762	104401	001164		BR	60\$; TRY AGAIN
4876	017766	000747					
4877				62\$:	CLRB	P.SMPL(R5)	; MAKE SAMPLED COMPARES
4878	017770	105065	000116				
4879				PREXAR:	TYPE	,PAR019	; TYPE "CHANGE EXCLUDED PACK AREA ="
4880	017774	104401	060300		TYPE	,PAR018	
4881	020000	104401	060252		JSR	R0,TSTDEF	; CHECK FOR DEFAULT PARAMETERS
4882	020004	004037	020340		IS		; COMMA DETECTED
4883	020010	020032			50\$; CARRIAGE RETURN DETECTED
4884	020012	020172			50\$; CONTROL Z (↑Z) DETECTED
4885	020014	020172			POS		; CONTROL C (↑C) DETECTED
4886	020016	016174			CMPI	#'Y,(R3)	; CHECK IF YES
4887	020020	122713	000131		BEQ	2\$; YES, LOAD NEW TRACK EXCLUSIONS
4888	020024	001411			JMP	50\$; NO, RETURN
4889	020026	000137	020172				
4890				1\$:	MOV	R3,64\$; LOAD BUFFER ADDRESS
4891	020032				TYPE		; TYPE RECEIVED INPUT
4892	020032	010337	020040	64\$:	.WORD	0	; BUFFER ADDRESS
4893	020036	104401			TYPE	\$QUES	; TYPE QUESTION MARK
4894	020040	000000			BR	PREXAR	; TRY AGAIN
4895	020042	104401	001164				
4896	020046	000752					
4897				2\$:	MOV	R1,-(SP)	; STORE R1 ON STACK
4898	020050	010146			MOV	R5,R1	; STORE PARAMETER BLOCK ADDRESS
4899	020052	010501			ADD	#P.EXAR,R1	; CALCULATE EXCLUSION AREA
4900	020054	062701	000232		MOV	#10,R4	; LOAD COUNT
4901	020060	012704	000012	5\$:	CLR	(R1)+	; CLEAR EXCLUSION AREA
4902	020064	005021			DEC	R4	; DECREMENT COUNT
4903	020066	005304			BNE	5\$; CHECK IF DONE
4904	020070	001375			SUB	#20,R1	; REINITIALIZE POINTER
4905	020072	162701	000024		JSR	R0,LDEXAR	; GET FIRST EXCLUDED AREA
4906	020076	004037	020506		PAR013		
4907	020102	060212			45\$; DEFAULT RETURN
4908	020104	020170			40\$; CONTROL C (↑C) RETURN
4909	020106	020162			JSR	R0,LDEXAR	; GET SECOND EXCLUDED AREA
4910	020110	004037	020506		PAR014		
4911	020114	060220			45\$; DEFAULT RETURN
4912	020116	020170			40\$; CONTROL C (↑C) RETURN
4913	020120	020162			JSR	R0,LDEXAR	; GET THIRD EXCLUDED AREA
4914	020122	004037	020506		PAR015		
4915	020126	060227			45\$; DEFAULT RETURN
4916	020130	020170			40\$; CONTROL C (↑C) RETURN
4917	020132	020162			JSR	R0,LDEXAR	; GET FOURTH EXCLUDED AREA
4918	020134	004037	020506		PAR016		
4919	020140	060235			45\$; DEFAULT RETURN
4920	020142	020170			40\$; CONTROL C (↑C) RETURN
4921	020144	020162			JSR	R0,LDEXAR	; GET FIFTH EXCLUDED AREA
4922	020146	004037	020506				

```

4923 020152 060244          PAR017
4924 020154 020170          45$          ;DEFAULT RETURN
4925 020156 020162          40$          ;CONTROL C (<tc>) RETURN
4926 020160 000403          BR          45$          ;RETURN
4927
4928 020162 012601          40$: MOV      (SP)+,R1          ;RESTORE R1
4929 020164 000137 016174  JMP          POS          ;START AGAIN
4930
4931 020170 012601          45$: MOV      (SP)+,R1          ;RESTORE R1
4932 020172 012603          50$: MOV      (SP)+,R3          ;RESTORE R3
4933 020174 012604          MOV      (SP)+,R4          ;RESTORE R4
4934 020176 000207          RTS          PC          ;RETURN
4935
4936 020200 013765 027154 000070 P1$: MOV      LSTCYL,P.MXCL(R5) ;LOAD DEFAULT MAXIMUM CYLINDER = 632/1456
4937 020206 005065 000066          CLR          P.MNCL(R5) ;LOAD DEFAULT MINIMUM CYLINDER
4938 020212 112765 000002 000073 P2$: MOV      #D.MXTR,P.MXTR(R5) ;LOAD DEFAULT MAXIMUM TRACK = 2
4939 020220 105065 000072          CLR          P.MNTR(R5) ;LOAD DEFAULT MINIMUM TRACK
4940 020224 112765 000025 000075 P3$: MOV      #D.MXSC,P.MXSC(R5) ;LOAD DEFAULT MAXIMUM SECTOR = 21
4941 020232 105065 000074          CLR          P.MNSC(R5) ;LOAD DEFAULT MINIMUM SECTOR
4942 020238 112765 000003 000076 P4$: MOV      #D.RATE,P.RATE(R5) ;LOAD READ/WRITE RATIO
4943 020244 105065 000077          CLR          P.AWCK(R5) ;LOAD WRITE CHECK AFTER WRITE
4944 020250 012765 000012 000100 P6$: MOV      #D.CERT,P.CERT(R5) ;LOAD CORRECTABLE READ ERROR THRESHOLD
4945 020256 012765 000005 000102 P7$: MOV      #D.UERT,P.UERT(R5) ;LOAD UNCORRECTABLE READ ERROR THRESHOLD
4946 020264 012765 000005 000104 P8$: MOV      #D.SKET,P.SKET(R5) ;LOAD SEEK ERROR THRESHOLD
4947 020272 105065 000120          CLR          P.IERC(R5) ;CLEAR INHIBIT ERROR CORRECTION
4948 020276 012765 077777 000110 P10$: MOV      #D.CTRH,P.MXCD+2(R5) ;LOAD COMMAND THRESHOLD
4949 020304 012765 177770 000106          MOV          #D.CTRL,P.MXCD(R5)
4950 020312 012765 077777 000114 P11$: MOV      #D.WTHI,P.MXWT+2(R5) ;LOAD WORD TRANSFERED THRESHOLD
4951 020320 012765 177770 000112          MOV          #D.WTLO,P.MXWT(R5)
4952 020326 105065 000116          CLR          P.SMPL(R5) ;MAKE SAMPLED COMPARES
4953
4954 020332 012603          MOV      (SP)+,R3          ;RESTORE R3
4955 020334 012604          MOV      (SP)+,R4          ;RESTORE R4
4956 020336 000207          RTS          PC          ;RETURN

```

.SBTTL TEST DEFAULT PARAMETERS ROUTINE

```

:*****
:* THIS ROUTINE WILL READ A LINE OF ASCII TEXT AND CHECK
:* IF PARAMETER OR REST OF PARAMETERS WILL BE DEFAULTED.
:* IT RETURNS WITH R3 POINTING TO THE ADDRESS OF THE BUFFER.
:*
:*CALL JSR RO,TSTDEF
:* <ADDRESS OF COMMA DETECTION RETURN>
:* <ADDRESS OF CARRIAGE RETURN DETECTION RETURN>
:* <ADDRESS OF CONTROL Z (<z>) DETECTION RETURN>
:* <ADDRESS OF CONTROL C (<c>) DETECTION RETURN>
:*
:* RETURN
:*****

```

```

4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974 020340 104403
4975 020342 012603
4976 020344 121327 000054
4977 020350 001415
4978 020352 005720
4979 020354 105713
4980 020356 001412
4981 020360 005720
4982 020362 122713 000032
4983 020366 001406
4984 020370 005720
4985 020372 122713 000003
4986 020376 001402
4987 020400 005720
4988 020402 000200
4989
4990 020404 011000
4991 020406 000200
4992
4993 020410 104401 060466
4994 020414 004037 020340
4995 020420 020444
4996 020422 020444
4997 020424 020444
4998 020426 020444
4999
5000 020430 121327 000131
5001 020434 001406
5002 020436 121327 000116
5003 020442 001412
5004
5005 020444 104401 056042
5006 020450 000757
5007
5008 020452 142765 000004 000007 2S:
5009 020460 012737 000632 027154
5010 020466 000207
5011 020470 152765 000004 000007 3S:
5012 020476 012737 001456 027154

```

```

TSTDEF: RDLIN          ; GET RESPONSE
          MOV          (SP)+,R3 ; STORE BUFFER ADDRESS
          CMPB        (R3),#' , ; CHECK IF COMMA
          BEQ         10$       ; YES, RETURN
          TST         (R0)+     ; ADJUST RO
          TSTB       (R3)       ; CHECK IF CARRIAGE RETURN
          BEQ         10$       ; YES, RETURN
          TST         (R0)+     ; ADJUST RO
          CMPB       #32,(R3)   ; CHECK IF CONTROL Z (<z>)
          BEQ         10$       ; YES, RETURN
          TST         (R0)+     ; ADJUST RO
          CMPB       #3,(R3)    ; CHECK IF CONTROL C (<c>)
          BEQ         10$       ; YES, RETURN
          TST         (R0)+     ; ADJUST RO
          RTS         RO        ; RETURN

10$:     MOV          (R0),RO    ; STORE RETURN ADDRESS
          RTS         RO        ; RETURN

DTYPE:  TYPE         PARC22     ; TYPE "TESTING RK06? Y OR N"
          JSR         RO,TSTDEF ; GET INPUT
          IS          ; COMMA
          IS          ; CR
          IS          ; CONT Z
          IS          ; CONT C

          CMPB       (R3),#' Y ; SEE IF RK06
          BEQ         2$       ; BR IF YES
          CMPB       (R3),#' N ; SEE IF RK07
          BEQ         3$       ; BR IF YES

1$:     TYPE         ILLCMD     ; ELSE ERROR
          BR          DTYPE     ; TRY AGAIN

2$:     BICB        #B.CDT,P.CS1H(R5) ; RK06
          MOV        #632,LSTCYL ; LAST CYLINDER ADDRESS
          RTS        PC

3$:     BISB        #B.CDT,P.CS1H(R5) ; RK07
          MOV        #1456,LSTCYL ; LAST CYL ADDRESS

```

F09

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 111
TEST DEFAULT PARAMETERS ROUTINE

SEQ 0109

S013 020504 000207
S014
S015

RTS PC

.SBTTL LOAD DISK EXCLUDED AREAS

```

*****
* THIS ROUTINE WILL ACCEPT TTY INPUT FOR EXCLUDED AREAS AND
* LOAD APPROPRIATE IDENTIFICATION NUMBERS IN THE EXCLUDED AREAS OF
* THE PARAMETER BLOCK. THE FOLLOWING THREE FORMATS ARE
* ACCEPTED BY THIS ROUTINE:
*
*       CYL
*       CYL,TRK
*       CYL1,TRK1,CYL2,TRK2
*
* THE FOLLOWING FORMULA IS USED:
*
*       CYL1*66+TRK1*22+1 = BEGINNING OF EXCLUDED AREA
*       CYL2*66+TRK2*22+1 = END OF EXCLUDED AREA
*
* REGISTER      USE
* -----      ---
*
* R1             ADDRESS IN EXCLUDED AREA
* R3             ADDRESS OF INPUT STRING
* R4             ADDRESS IN INPUT STRING
* R5             PARAMETER BLOCK ADDRESS
*
*CALL JSR      RD,LDEXAR
*      <ADDRESS OF MESSAGE TO PRINT>
*      <ADDRESS FOR CARRIAGE RETURN OR CONTROL Z <↑Z> DETECTION>
*      <ADDRESS FOR CONTROL C <↑C> DETECTION>
*      RETURN
*****

```

```

5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050 020506 012037 020514
5051
5052 020512 104401
5053 020514 000000
5054 020516 104401 060252
5055 020522 004037 020340
5056 020526 021132
5057 020530 021154
5058 020532 021154
5059 020534 021152
5060 020536 010346
5061 020540 004737 054146
5062 020544 021132
5063 020546 023716 027154
5064 020552 103566
5065 020554 011637 001616
5066 020560 006316
5067 020562 062637 001616
5068 020566 010304
5069
5070 020570 105714
5071 020572 001525

```

```

LDEXAR: MOV      (R0)+,2$      ;LOAD NUMBER OF EXCLUSION AREA
1$:      TYPE
2$:      .WORD      0      ;TYPE "NTH EXCLUDED AREA ="
          TYPE      PAR018
          JSR      RD,TSTDEF      ;CHECK FOR DEFAULT PARAMETERS
          30$      ;COMMA DETECTED
          45$      ;CARRIAGE RETURN DETECTED
          45$      ;CONTROL Z <↑Z> DETECTED
          40$      ;CONTROL C <↑C> DETECTED
          MOV      R3,-(SP)      ;STORE BUFFER ADDRESS
          JSR      PC,OCTBIN      ;CONVERT TO BINARY
          30$      ;ERROR RETURN
          CMP      LSTCYL,(SP)      ;CHECK IF LEGAL CYLINDER
          BLO      29$      ;NO, TRY AGAIN
          MOV      (SP),EXAREA      ;MULTIPLY CYLINDER ADDRESS BY 3
          ASL      (SP)
          ADD      (SP)+,EXAREA
          MOV      R3,R4      ;STORE BEGINNING OF COMMAND STRING
64$:     TSTB      (R4)      ;CHECK FOR CARRIAGE RETURN
          BEQ      10$      ;YES, EXIT
          10$

```


5072	020574	122427	000054	CMPB	(R4)+, #'	:CHECK FOR COMMA
5073	020600	001373		BNE	64\$:NO, GO TEST NEXT CHARACTER
5074	020602	105714		TSTB	(R4)	:CHECK IF CARRIAGE RETURN
5075	020604	001552		BEQ	30\$:YES, TRY AGAIN
5076	020606	121427	000054	CMPB	(R4), #'	:CHECK IF COMMA
5077	020612	001547		BEQ	30\$:YES, TRY AGAIN
5078	020614	010446		MOV	R4, -(SP)	:STORE BUFFER ADDRESS
5079	020616	004737	054146	JSR	PC, OCTBIN	:CONVERT TO BINARY
5080	020622	021132		30\$:ERROR RETURN
5081	020624	022716	000002	CMP	#2 (SP)	:CHECK IF LEGAL TRACK
5082	020630	103537		BLO	29\$:TRY AGAIN
5083	020632	063716	001616	ADU	EXAREA, (SP)	:PUT MULTIPLIER ON STACK
5084	020636	012746	000026	MOV	#22, -(SP)	:PUT MULTIPLIER ON STACK
5085	020642	004737	054714	JSR	PC, \$MULT	:DO MULTIPLICATION
5086	020646	005216		INC	(SP)	:ADD 1
5087	020650	012637	001616	MOV	(SP)+, EXAREA	:STORE VALUE
5088	020654	005726		TST	(SP)+	:THROW AWAY MOST SIGNIFICANT BITS
5089						
5090	020656	105714		65\$: TSTB	(R4)	:CHECK FOR CARRIAGE RETURN
5091	020660	001511		BEQ	20\$:YES, EXIT
5092	020662	122427	000054	CMPB	(R4)+, #'	:CHECK FOR COMMA
5093	020666	001373		BNE	65\$:NO, GO TEST NEXT CHARACTER
5094	020670	105714		TSTB	(R4)	:CHECK IF CARRIAGE RETURN
5095	020672	001517		BEQ	30\$:YES, TRY AGAIN
5096	020674	121427	000054	CMPB	(R4), #'	:CHECK IF COMMA
5097	020700	001514		BEQ	30\$:YES, TRY AGAIN
5098	020702	010446		MOV	R4, -(SP)	:STORE BUFFER ADDRESS
5099	020704	004737	054146	JSR	PC, OCTBIN	:CONVERT TO BINARY
5100	020710	021132		30\$:ERROR RETURN
5101	020712	023716	027154	CMP	LSTCYL, (SP)	:CHECK IF LEGAL CYLINDER
5102	020716	103504		BLO	29\$:NO, TRY AGAIN
5103	020720	011637	001620	MOV	(SP), EXAREA+2	:MULTIPLY CYLINDER ADDRESS BY 3
5104	020724	006316		ASL	(SP)	
5105	020726	062637	001620	ADD	(SP)+, EXAREA+2	
5106						
5107	020732	105714		66\$: TSTB	(R4)	:CHECK FOR CARRIAGE RETURN
5108	020734	001476		BEQ	30\$:YES, EXIT
5109	020736	122427	000054	CMPB	(R4)+, #'	:CHECK FOR COMMA
5110	020742	001373		BNE	66\$:NO, GO TEST NEXT CHARACTER
5111	020744	105714		TSTB	(R4)	:CHECK IF CARRIAGE RETURN
5112	020746	001471		BEQ	30\$:YES, TRY AGAIN
5113	020750	121427	000054	CMPB	(R4), #'	:CHECK IF COMMA
5114	020754	001466		BEQ	30\$:YES, TRY AGAIN
5115	020756	010446		MOV	R4, -(SP)	:STORE BUFFER ADDRESS
5116	020760	004737	054146	JSR	PC, OCTBIN	:CONVERT TO BINARY
5117	020764	021132		30\$:ERROR RETURN
5118	020766	022716	000002	CMP	#2 (SP)	:CHECK IF LEGAL TRACK
5119	020772	103456		BLO	29\$:TRY AGAIN
5120	020774	063716	001620	ADD	EXAREA+2, (SP)	:PUT MULTIPLIER ON STACK
5121	021000	012746	000026	MOV	#22, -(SP)	:PUT MULTIPLIER ON STACK
5122	021004	004737	054714	JSR	PC, \$MULT	:DO MULTIPLICATION
5123	021010	005216		INC	(SP)	:ADD 1
5124	021012	012637	001620	MOV	(SP)+, EXAREA+2	:STORE VALUE
5125	021016	005726		TST	(SP)+	:THROW AWAY MOST SIGNIFICANT BITS
5126	021020	023737	001616 001620	CMP	EXAREA, EXAREA+2	:CHECK IF LEGAL RANGE
5127	021026	103041		BHIS	30\$:NO, TRY AGAIN

S128	021030	013721	001616		MOV	EXAREA, (R1)+	; LOAD EXCLUDED AREA
S129	021034	013721	001620		MOV	EXAREA+2, (R1)+	
S130	021040	062700	000004		ADD	#4, RO	; ADJUST RO
S131	021044	000200			RTS	RO	; RETURN
S132							
S133	021046	013746	001616	10\$:	MOV	EYAREA, -(SP)	; LOAD MULTIPLIER
S134	021052	012746	000026		MOV	#22, -(SP)	; LOAD MULTICAND
S135	021056	004737	054714		JSR	PC, \$MULT	; CALCULATE VALUE
S136	021062	005216			INC	(SP)	; INCREMENT RESULTS
S137	021064	011621			MOV	(SP), (R1)+	; STORE EXCLUDED AREA
S138	021066	062716	000102		ADD	#66, (SP)	
S139	021072	012621			MOV	(SP)+, (R1)+	
S140	021074	005726			TST	(SP)+	; THROW AWAY MUST SIGNIFICANT BITS
S141	021076	062700	000004		ADD	#4, RO	; ADJUST RO
S142	021102	000200			RTS	RO	; RETURN
S143							
S144	021104	013721	001616	20\$:	MOV	EXAREA, (R1)+	; STORE EXCLUDEED AREA
S145	021110	062737	000026	001616	ADD	#22, EXAREA	
S146	021116	013721	001616		MOV	EXAREA, (R1)+	
S147	021122	062700	000004		ADD	#4, RO	; ADJUST RO
S148	021126	000200			RTS	RO	; RETURN
S149							
S150	021130	005726		29\$:	TST	(SP)+	; ADJUST STACK
S151	021132	010337	021140	30\$:	MOV	R3, 31\$; LOAD BUFFER ADDRESS
S152	021136	104401			TYPE		; TYPE RECEIVED INPUT
S153	021140	000000		31\$:	.WORD	0	; BUFFER ADDRESS
S154	021142	104401	001164		TYPE	\$QUES	; TYPE QUESTION MARK
S155	021146	000137	020512		JMP	1\$; TRY AGAIN
S156							
S157	021152	005720		40\$:	TST	(RO)+	; ADJUST RO
S158	021154	011000		45\$:	MOV	(RO), RO	; STORE SPECIAL RETURN ADDRESS
S159	021156	000200			RTS	RO	; RETURN

.SBTTL GENERATE NEW RANDOM COMMAND

S160
S161
S162
S163
S164
S165
S166
S167
S168
S169
S170
S171
S172
S173
S174
S175
S176
S177
S178
S179
S180
S181
S182
S183
S184
S185
S186
S187
S188
S189
S190
S191
S192
S193
S194
S195
S196
S197
S198
S199
S200
S201
S202
S203
S204
S205
S206
S207
S208
S209
S210
S211
S212
S213
S214
S215

```

*****
*
* IF SWITCH 14 IS RESET, THIS ROUTINE WILL STORE THE PREVIOUS
* COMMAND AND GENERATE A NEW RANDOM COMMAND. A CHECK IS MADE
* ON THE DISK ADDRESS AND THE DISK ADDRESS + THE NUMBER OF
* SECTORS TO BE READ OR WRITTEN. IF THE DATA TRANSFER WILL
* CAUSE DATA TO BE READ OR WRITTEN BEYOND THE MAXIMUM PACK
* ADDRESS OR USES ANY PART OF AN EXCLUDED AREA, A NEW DISK
* ADDRESS WILL BE RANDOMLY GENERATED.
*
* REGISTER          USE
* -----          ---
*
* R1                EXCLUSION AREA COUNT
* R3                EXCLUSION AREA INDEX
* R4                SECTOR POSITION
* R5                PARAMETER BLOCK ADDRESS
* (SP)             SECTORS TO BE TRANSFERRED
*
* IF SWITCH 14 IS SET, THIS ROUTINE WILL SEEK TO THE PREVIOUS
* DISK POSITION OR REISSUE THE LAST COMMAND DEPENDENT ON THE
* P.SEEK FLAG IN THE PARAMETER BLOCK.
*
* ROUTINES USED
* -----
*
* Q.PUSH
* $RAND
* $MULT
*
*CALL JSR      PC, GENERT
* RETURN
*****

```

```

GENERT: CLR      P.RECT(R5)      ;CLEAR ERROR FLAGS AND
CLR      P.DSTT(R5)           ;RETRY COUNTS
CLR      P.ERR(R5)
BIT      #SW14,JSWR           ;CHECK IF NEW COMMAND
                                ;SHOULD BE GENERATED
BEQ      10$                  ;YES, GENERATE NEW COMMAND
TSTB    P.SEEK(R5)           ;CHECK IF PREVIOUS CYLINDER ISSUED
BNE     1$                    ;YES, SET UP DATA TRANSFER
MOVB    #-1,P.SEEK(R5)       ;INDICATE SEEK HAS BEEN ISSUED
MOVB    #SEEK,P.CMND(R5)     ;LOAD SEEK PARAMETERS
MOV     P.LCYL(R5),P.CYLN(R5) ;LOAD PRIOR CYLINDER
MOV     P.LSEC(R5),P.SECT(R5) ;LOAD PRIOR SECTOR AND TRACK
JSR     RD,Q.PUSH            ;PUT PARAMETER BLOCK ON
CINITQ  ;COMMAND INITIATION QUEUE
RTS     PC                   ;RETURN

```

```

021160 005065 000146
021164 005065 000150
021170 005065 000212
021174 032777 040000 157736
021202 001454
021204 105765 000211
021210 001020
021212 112765 177777 000211
021220 112765 000117 000001
021226 016565 000136 000002
021234 016565 000140 000004
021242 004037 051406
021246 001324
021250 000207

```

5216	021252	105065	000211		15:	CLRB	P. SEEK(R5)	: INDICATE DATA TRANSFER ISSUED
5217	021256	116565	000122	000121		MOVB	P. RDPT(R5), P. DPAT(R5)	: LOAD LAST DATA PATTERN
5218	021264	116565	000123	000001		MOVB	P. RCMD(R5), P. CMND(R5)	: LOAD LAST COMMAND
5219	021272	016565	000124	000002		MOV	P. RCYL(R5), P. CYLN(R5)	: LOAD LAST CYLINDER ADDRESS
5220	021300	122765	000131	000001		CMPB	#WRTCHK, P. CMND(R5)	: CHECK IF WRITE CHECK
5221	021306	001003				BNE	25	: NO, DO NOT SET FLAG
5222	021310	052765	000200	000014		BIS	#W.WCK, P. PRST(R5)	: SET ISSUE WRITE FOR WRITE CHECK
5223	021316	016565	000126	000004	25:	MOV	P. RSEC(R5), P. SECT(R5)	: LOAD LAST SECTOR AND TRACK
5224	021324	016565	000130	000012		MOV	P. RWC(R5), P. WC(R5)	: LOAD LAST WORD COUNT
5225	021332	000207				RTS	PC	: RETURN
5226								
5227	021334	005077	157604		105:	CLR	#STKS	: LOCK OUT TTY INTERRUPTS
5228	021340	010446				MOV	R4, -(SP)	: STORE R4 ON STACK
5229	021342	010346				MOV	R3, -(SP)	: STORE R3 ON STACK
5230	021344	010146				MOV	R1, -(SP)	: STORE R1 ON STACK
5231	021346	010046				MOV	R0, -(SP)	: STORE R0 ON STACK
5232	021350	116565	000123	000135		MOVB	P. RCMD(R5), P. LCMO(R5)	: STORE LAST COMMAND
5233	021356	016565	000124	000136		MOV	P. RCYL(R5), P. LCYL(R5)	: STORE LAST CYLINDER
5234	021364	016565	000126	000140		MOV	P. RSEC(R5), P. LSEC(R5)	: STORE LAST SECTOR AND TRACK
5235	021372	016565	000130	000142		MOV	P. RWC(R5), P. LWC(R5)	: STORE LAST WORD COUNT
5236	021400	116565	000122	000144		MOVB	P. RDPT(R5), P. LDPT(R5)	: STORE LAST PATTERN
5237	021406	004737	055030			JSR	PC, \$RAND	: GENERATE RANDOM NUMBER
5238	021412	013746	055130			MOV	\$LONUM, -(SP)	: LOAD MULTIPLIER
5239	021416	013746	001354			MOV	MAXBUF, -(SP)	: LOAD MULTIPLICAND
5240	021422	005216				INC	(SP)	: INCREMENT WORD COUNT MAX.
5241	021424	004737	054714			JSR	PC, \$MULT	: CALCULATE WORD COUNT
5242	021430	005726				TST	(SP)+	: THROW AWAY LEAST SIGNIFICANT BITS
5243	021432	042716	100003			BIC	#100003, (SP)	: MAKE DIVISIBLE BY 4 AND < 32K
5244	021436	001002				BNE	115	: IF NOT ZERO USE WORD COUNT
5245	021440	012716	000004			MOV	#4, (SP)	: MAKE WORD COUNT 4
5246	021444	116604	000001		115:	MOVB	1(SP), R4	: CALCULATE NUMBER OF SECTORS TO
5247	021450	105716				TSTB	(SP)	: TRANSFERRED
5248	021452	001401				BEQ	125	: IF EVEN SECTORS DO NOT MODIFY
5249	021454	005204				INC	R4	: SECTOR COUNT
5250	021456	011646			125:	MOV	(SP), -(SP)	: SAVE WORD COUNT
5251	021460	005416				NEG	(SP)	: GENERATE NEGATIVE NUMBER
5252	021462	012665	000012			MOV	(SP)+, P. WC(R5)	: LOAD PRESENT WORD COUNT
5253	021466	062765	000001	000152		ADD	#1, P. NODR(R5)	: INCREMENT NUMBER OF ORDERS
5254	021474	005565	000154			ADC	P. NODR+2(R5)	
5255	021500	113746	055126			MOVB	\$HINUM, -(SP)	: RANDOM NUMBER FOR COMMAND
5256	021504	042716	177770			BIC	#177770, (SP)	: KEEP LOW 3 BITS
5257	021510	126526	000076			CMPB	P. RATE(R5), (SP)+	: DETERMINE IF READ OR WRITE
5258	021514	101462				BLOS	155	: ISSUE READ
5259	021516	113746	055127			MOVB	\$HINUM+1, -(SP)	: RANDOM NUMBER FOR PATTERN
5260	021522	042716	177760			BIC	#177760, (SP)	: KEEP LOW 4 BITS
5261	021526	111655	000122			MOVB	(SP), P. RDPT(R5)	: STORE DATA PATTERN
5262	021532	112665	000121			MOVB	(SP)+, P. DPAT(R5)	
5263	021536	105765	000077			TSTB	P. AWCK(R5)	: CHECK IF AUTO WRITE CHECK
5264	021542	001017				BNE	135	: YES, ISSUE WRITE CHECK
5265	021544	113746	055126			MOVB	\$HINUM, -(SP)	: RANDOM NUMBER FOR WRITE CHECK
5266	021550	042726	177707			BIC	#177707, (SP)+	: CHECK IF WRITE COMMAND
5267	021554	001412				BEQ	135	: YES, ISSUE WRITE CHECK
5268	021556	112765	000123	000123		MOVB	#WRDATA, P. RCMD(R5)	: ISSUE WRITE DATA
5269	021564	062665	000156			ADD	(SP)+, P. NWRT(R5)	: ADD NUMBER OF WORDS TO BE WRITTEN
5270	021570	005565	000160			ADC	P. NWRT+2(R5)	
5271	021574	005565	000162			ADC	P. NWRT+4(R5)	

```

5272 021600 000441 BR 18$
5273
5274 021602 112765 000131 000123 13$: MOVB #WRTCHK,P.RCMD(R5) ;ISSUE WRITE CHECK
5275 021610 052765 000200 000014 BIS #W.WCK,P.PRST(R5) ;SET NO WRITE FOR WRITE CHECK
5276 021616 062765 000001 000152 ADD #1,P.NODR(R5) ;INCREMENT NUMBER OF COMMAND ISSUED
5277 021624 005565 000154 ADC P.NODR+2(R5)
5278 021630 061665 000156 ADD (SP),P.NWRT(R5) ;ADD NUMBER OF WORDS TO BE WRITTEN
5279 021634 005565 000160 ADC P.NWRT+2(R5)
5280 021640 005565 000162 ADC P.NWRT+4(R5)
5281 021644 062665 000164 ADD (SP)+,P.NRD(R5) ;ADD NUMBER OF WORDS READ
5282 021650 005565 000166 ADC P.NRD+2(R5)
5283 021654 005565 000170 ADC P.NRD+4(R5)
5284 021660 000411 BR 18$
5285
5286 021662 112765 000121 000123 15$: MOVB #RDDATA,P.RCMD(R5) ;ISSUE READ DATA
5287 021670 062665 000164 ADD (SP)+,P.NRD(R5) ;ADD NUMBER OF WORDS READ
5288 021674 005565 000166 ADC P.NRD+2(R5)
5289 021700 005565 000170 ADC P.NRD+4(R5)
5290 021704 116565 000123 000001 18$: MOVB P.RCMD(R5),P.CMND(R5) ;LOAD COMMAND
5291 021712 012700 000100 MOV #100,R0 ;LOAD REGENERATE COUNT
5292
5293 021716 010503 20$: MOV R5,R3 ;LOAD INDEX
5294 021720 062703 000232 ADD #P.EXAR,R3 ;CALCULATE EXCLUDED AREA INDEX
5295 021724 012701 000005 MOV #5,R1 ;LOAD BAD TRACK COUNT
5296 021730 004737 055030 JSR PC,$RAND ;GENERATE RANDOM NUMBER
5297 021734 016546 000070 MOV P.MXCL(R5),-(SP) ;STORE MAXIMUM CYLINDER
5298 021740 166516 000066 SUB P.MNCL(R5),(SP) ;CALCULATE DIFFERENCE
5299 021744 001406 BEQ 21$ ;IF 0, SKIP MULTIPLY
5300 021746 005216 INC (SP) ;INCREMENT DIFFERENCE
5301 021750 013746 055130 MOV $LONUM,-(SP) ;PUT MULTICAND ON STACK
5302 021754 004737 054714 JSR PC,$MULT ;CALCULATE CYLINDER ADDRESS
5303 021760 005726 TST (SP)+ ;THROW AWAY LEASE SIGNIFICANT BITS
5304
5305 021762 066516 000066 21$: ADD P.MNCL(R5),(SP) ;CALCULATE CYLINDER
5306 021766 011665 000124 MOV (SP),P.RCYL(R5) ;STORE CYLINDER
5307 021772 116537 000073 001572 MOVB P.MXTR(R5),TMPTRK ;STORE MAXIMUM TRACK
5308 022000 116537 000072 001574 MOVB P.MNTR(R5),TMPTRK+2 ;STORE MINIMUM TRACK
5309 022006 163737 001574 001572 SUB TMPTRK+2,TMPTRK ;CALCULATE PERFERENCE
5310 022014 001415 BEQ 22$ ;IF=0, SKIP MULTIPLY
5311 022016 005046 CLR -(SP) ;MAKE ROOM ON STACK
5312 022020 113716 055127 MOVB $HINUM+1,(SP) ;PUT IN MULTIPLIER ON STACK
5313 022024 013746 001572 MOV TMPTRK,-(SP) ;PUT TRACK NUMBER ON STACK
5314 022030 005216 INC (SP) ;INCREMENT DIFFERENCE
5315 022032 004737 054714 JSR PC,$MULT ;CALCULATE TRACK
5316 022036 116637 000001 001572 MOVB 1,(SP),TMPTRK ;STORE TRACK
5317 022044 062706 000004 ADD #4,SP ;ADJUST STACK
5318
5319 022050 063737 001574 001572 22$: ADD TMPTRK+2,TMPTRK ;CALCULATE TRACK
5320 022056 113765 001572 000127 MOVB TMPTRK,P.RTRK(R5) ;STORE TRACK NUMBER
5321 022064 116537 000075 001576 MOVB P.MXSC(R5),TMPSEC ;STORE MAXIMUM SECTOR
5322 022072 116537 000074 001600 MOVB P.MNSC(R5),TMPSEC+2 ;STORE MINIMUM SECTOR
5323 022100 163737 001600 001576 SUB TMPSEC+2,TMPSEC ;CALCULATE DIFFERENCE
5324 022106 001415 BEQ 23$ ;IF=0, SKIP MULTIPLY
5325 022110 005046 CLR -(SP) ;MAKE ROOM ON STACK
5326 022112 113716 055126 MOVB $HINUM,(SP) ;PUT IN MULTIPLIER
5327 022116 013746 001576 MOV TMPSEC,-(SP) ;PUT SECTOR NUMBER ON STACK

```

5328	022122	005216				INC	(SP)	; INCREMENT DIFFERENCE
5329	022124	004737	054714			JSR	PC, \$MULT	; CALCULATE SECTOR
5330	022130	116637	000001	001576		MOV	1(SP), TMPSEC	; STORE SECTOR
5331	022136	062706	000004			ADD	#4, SP	; ADJUST STACK
5332								
5333	022142	063737	001600	001576	23\$:	ADD	TMPSEC+2, TMPSEC	; CALCULATE SECTOR
5334	022150	113765	001576	000126		MOV	TMPSEC, P.RSEC(R5)	; STORE SECTOR NUMBER
5335	022156	006316				ASL	(SP)	; MULTIPLY CYLINDER BY 3
5336	022160	066516	000124			ADD	P.RCYL(R5), (SP)	
5337	022164	063716	001572			ADD	TMPTRK, (SP)	; ADD TRACK
5338	022170	012746	000026			MOV	#22, -(SP)	; MULTIPLICAND - 22 SECTORS/TRACK
5339	022174	004737	054714			JSR	PC, \$MULT	
5340	022200	012616				MOV	(SP)+, (SP)	; THROW AWAY MOST SIGNIFICANT BITS
5341	022202	063716	001576			ADD	TMPSEC, (SP)	; ADD SECTOR
5342	022206	005216				INC	(SP)	; CONSTANT FOR BAD TRACKS
5343	022210	061604				ADD	(SP), R4	; ADD NUMBER OF SECTORS
5344	022212	004737	027074			JSR	PC, GETNUM	
5345	022216	023704	027162			CMP	HOLD1, R4	; CHK IF DATA XFER EXCEEDS DISK
5346	022222	103421				BLO	40\$; CALCULATE NEW DISK ADDRESS
5347								
5348	022224	005713			25\$:	TST	(R3)	; CHECK IF BAD TRACK INTERVAL
5349								; EXIST
5350	022226	001407				BEQ	30\$; NO, ISSUE COMMAND
5351	022230	021623				CMP	(SP), (R3)+	; CHECK IF IN BAD TRACK INTERVAL
5352	022232	103402				BLO	26\$; NO, CHECK NEXT BAD TRACK 1
5353	022234	020413				CMP	R4, (R3)	
5354	022236	103413				BLO	40\$; YES, CALCULATE NEW DISK ADDRESS
5355	022240	005723			26\$:	TST	(R3)+	; ADJUST R3
5356	022242	005301				DEC	R1	; DECREMENT BAD BLOCK COUNT
5357	022244	001367				BNE	25\$; IF NOT 0 GET NEXT BAD BLOCK
5358								
5359	022246	005726			30\$:	TST	(SP)+	; ADJUST STACK
5360	022250	016565	000124	000002		MOV	P.RCYL(R5), P.CYL(R5)	; LOAD CYLINDER
5361	022256	016565	000126	000004		MOV	P.RSEC(R5), P.SEC(R5)	; LOAD TRACK AND SECTOR
5362	022264	000416				BR	45\$; RETURN
5363								
5364	022266	162604			40\$:	SUB	(SP)+, R4	; GET NUMBER OF SECTOR TRANSFERRED
5365	022270	005300				DEC	R0	; DECREMENT REGENERATE COUNT
5366	022272	001402				BEQ	41\$; DROP DRIVE CANNOT GENERATE
5367								; RANDOM DISK ADDRESS
5368	022274	000137	021716			JMP	20\$; GET ANOTHER DISK ADDRESS
5369								
5370	022300	013737	001452	177776	41\$:	MOV	RKPRI, PS	; LOCK OUT RK06 AND TTY INTERRUPTS
5371	022306	104401	060307			TYPE	PAR020	; TYPE "UNABLE TO GENERATE NEW COMMAND"
5372	022312	004737	013004			JSR	PC, DROP	; DROP DRIVE
5373	022316	005037	177776			CLR	PS	; ALLOW RK06 INTERRUPTS
5374								
5375	022322	012600			45\$:	MOV	(SP)+, R0	; RESTORE R0
5376	022324	012601				MOV	(SP)+, R1	; RESTORE R1
5377	022326	012603				MOV	(SP)+, R3	; RESTORE R3
5378	022330	012604				MOV	(SP)+, R4	; RESTORE R4
5379	022332	012777	000100	156604		MOV	#IE, \$TKS	; ALLOW TTY INTERRUPTS
5380	022340	000207				RTS	PC	; RETURN

```

.SBTTL GET AVAILABLE BUFFER ROUTINE
5381
5382
5383 022342 013746 177776 GETBUF: MOV PS (SP) ;STORE PSW ON STACK
5384 022346 013737 001452 177776 MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
5385 022354 013705 001320 MOV BWAITQ,R5 ;CHECK IF COMMAND WAITING FOR BUFFER
5386 022360 001424 BEQ 10$ ;RETURN IF QUEUE IS EMPTY
5387 022362 004037 022440 1$: JSR RO,BUFFAL ;ALLOCATE BUFFER
5388 022366 022432 IO$ ;NOT ENOUGH MEMORY RETURN
5389 022370 004037 051466 JSR RO,Q.POP ;GET NEXT PARAMETER BLOCK FROM
5390 022374 001320 BWAITQ ; BUFFER WAIT QUEUE
5391 022376 005726 TST (SP)+ ;ADJUST STACK
5392 022400 005037 177776 CLR PS ;ALLOW RK06 INTERRUPTS
5393 022404 004737 023056 JSR PC,LDDATA ;LOAD BUFFER
5394 022410 004037 051406 JSR RO,Q.PUSH ;PUT PARAMETER BLOCK ON
5395 022414 001324 CINITQ ; COMMAND INITIATION QUEUE
5396 022416 013737 001452 177776 MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
5397 022424 013705 001320 MOV BWAITQ,R5 ;CHECK IF COMMAND WAITING FOR BUFFER
5398 022430 001354 BNE 1$ ;YES, TRY TO ALLOCATE BUFFER
5399
5400 022432 012637 177776 10$: MOV (SP)+,PS ;RESTORE PSW
5401 022436 000207 RTS PC ;RETURN

```

.SBTTL BUFFER ALLOCATION ROUTINE

```

*****
*
* THIS ROUTINE WILL TRY TO ALLOCATE A MAIN MEMORY BUFFER
* IF MEMORY IS AVAILABLE. (CALLED BY PRIORITY 7)
*
* REGISTER      USE
* -----      ---
*
* R1             WORD COUNT * 2
* R3             FREE BLOCK TABLE INDEX
* R4             FREE BLOCK TABLE COUNT
* R5             PARAMETER BLOCK ADDRESS
*
*CALL JSR      R0,BUFFAL
*      <ADDRESS OF BUFFER MEMORY NOT AVAILABLE ROUTINE>
*      RETURN <BUFFER ALLOCATED>
*
*****

```

```

5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5420
5421
5422
5423 022440 010446
5424 022442 010346
5425 022447 010146
5426 022446 016501 000C12
5427 022452 010165 000130
5428 022456 005401
5429 022460 006301
5430 022462 013704 001670
5431 022466 001420
5432 022470 012703 001672
5433 022474 022704 000011
5434 022500 101001
5435 022502 000000
5436
5437 022504 012346
5438 022506 060116
5439 022510 103411
5440 022512 021316
5441 022514 001423
5442 022516 101010
5443 022520 005723
5444 022522 005726
5445 022524 005304
5446 022526 001366
5447 022530 011000
5448 022532 000437
5449
5450 022534 005726
5451 022536 000774
5452
5453 022540 014365 000132
5454 022544 011365 000010
5455 022550 012613
5456 022552 005720
5457 022554 112765 177777 000145

```

```

BUFFAL:  MOV      R4,-(SP)      ;STORE R4 ON STACK
         MOV      R3,-(SP)      ;STORE R3 ON STACK
         MOV      R1,-(SP)      ;STORE R1 ON STACK
         MOV      P.WC(R5),R1    ;STORE WORD COUNT
         MOV      R1,P.RWC(R5)  ;STORE WORD COUNT
         NEG      R1             ;GET ACTUAL WORD COUNT
         ASL      R1             ;MULTIPLY WORD COUNT BY 2
         MOV      FBLKC,R4      ;STORE FREE BLOCK TABLE ADDRESS
         BEQ      2$            ;RETURN, IF NO BLOCKS AVAILABLE
         MOV      #FBLKT,R3     ;LOAD ADDRESS OF FREE BLOCK TABLE
         CMP      #9,R4         ;HANG IF FREE BLOCK COUNT IS GREATER
         BHI     1$            ; OR EQUAL TO 9
         HALT

1$:      MOV      (R3)+,-(SP)    ;STORE BEGINNING OF FREE BLOCK
         ADD      R1,(SP)      ;ADD WORD COUNT
         BCS     5$            ;CHECK IF ADDRESS OVERFLOW
         CMP      (R3),(SP)    ;CHECK IF ENOUGH ROOM
         BEQ     15$           ;YES, ELIMINATE BLOCK
         BHI     10$           ;YES, SHORTEN BLOCK
         TST     (R3)+         ;ADJUST R3
         TST     (SP)+         ;ADJUST STACK
         DEC     R4             ;DECREMENT FREE BLOCK COUNT
         BNE     1$            ;EXAMINE NEXT FREE BLOCK
2$:      MOV      (R0),R0       ;LOAD RETURN ADDRESS
         BR      25$           ;RETURN

5$:      TST     (SP)+         ;ADJUST STACK
         BR      2$            ;RETURN

10$:     MOV      -(R3),P.RCAL(R5) ;LOAD BUS ADDRESS
         MOV      (R3),P.BALO(R5) ;LOAD BUS ADDRESS
         MOV      (SP)+,(R3)    ;LOAD NEW BASE
         TST     (R0)+         ;ADJUST RETURN
         MOVB   # ,P.BUFF(R5)  ;INDICATE THAT BUFFER HAS BEEN ASSIGNED

```



```

5458 022562 000423          BR      25$          ;RETURN
5459
5460 022564 010301          15$:  MOV     R3,R1          ;STORE WORD COUNT
5461 022566 014365 000132      MOV     -(R3),P.RBAL(R5)
5462 022572 011365 000010      MOV     (R3),P.BALO(R5) ;LOAD BUS ADDRESS
5463 022576 005726          TST     (SP)↓           ;ADJUST STACK
5464 022600 005721          TST     (R1)↑           ;ADJUST R1
5465 022602 005720          TST     (R0)↑           ;ADJUST RETURN
5466 022604 112765 177777 000145      MOVB   #-1,P.BUFF(R5) ;INDICATE THAT BUFFER HAS BEEN ASSIGNED
5467 022612 005337 001670      DEC     FBLKC           ;DECREMENT FREE BLOCK COUNT
5468 022616 005304          DEC     R4              ;DECREMENT REMAIN BLOCK COUNT
5469 022620 001404          BEQ     25$             ;IF 0, RETURN
5470
5471 022622 012123          20$:  MOV     (R1)↑,(R3)↑   ;TRANSFER TWO WORDS
5472 022624 012123          MOV     (R1)↑,(R3)↑
5473 022626 005304          DEC     R4              ;DECREMENT REMAINING BLOCKS TO BE ADJUSTED
5474 022630 001374          BNE    20$             ;IF NOT 0, TRANSFER NEXT TWO WORDS
5475 022632 012601          25$:  MOV     (SP)↑,R1       ;RESTORE R1
5476 022634 012603          MOV     (SP)↑,R3       ;RESTORE R3
5477 022636 012604          MOV     (SP)↑,R4       ;RESTORE R4
5478 022640 000200          RTS     R0              ;RETURN
5479

```

.SBTTL BUFFER RELEASE ROUTINE

```
*****  
* THIS ROUTINE WILL RELEASE BUFFER USED.  
* (MUST BE CALLED IN PRIORITY 7)  
*  
* REGISTER USE  
* -----  
*  
* R0 NEW END OF FREE BLOCK TABLE  
* R1 LAST ADDRESS OF RELEASED BUFFER  
* R3 FREE BLOCK INDEX  
* R4 FREE BLOCK COUNT  
* R5 PARAMETER BLOCK ADDRESS  
* (SP) BEGINNING ADDRESS OF RELEASED BLOCK  
*  
*CALL JSR PC,BUFREL  
* RETURN  
*  
*****
```

5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502 022642 010446
5503 022644 010346
5504 022646 010146
5505 022650 010046
5506 022652 105765 000145
5507 022656 001472
5508 022660 105065 000145
5509 022664 016501 000130
5510 022670 005401
5511 022672 006301
5512 022674 016546 000132
5513 022700 061601
5514 022702 012703 001672
5515 022706 013704 001670
5516 022712 001424
5517
5518 022714 022301
5519 022716 001427
5520 022720 101005
5521 022722 022316
5522
5523 022724 001426
5524 022726 005304
5525 022730 001371
5526 022732 000414
5527
5528 022734 010446
5529 022736 006316
5530 022740 006316
5531 022742 005743
5532 022744 062603
5533 022746 010300
5534 022750 062700 000004
5535 022754 014340

```
BUFREL: MOV R4, -(SP) ; STORE R4 ON STACK  
MOV R3, -(SP) ; STORE R3 ON STACK  
MOV R1, -(SP) ; STORE R1 ON STACK  
MOV R0, -(SP) ; STORE R0 ON STACK  
TSTB P, BUFF(R5) ; CHECK IF BUFFER ASSIGNED  
BEQ 25$ ; NO, RETURN  
CLR B P, BUFF(R5) ; CLEAR BUFFER ALLOCATED  
MOV P, RAL(R5), R1 ; STORE ALLOCATED WORD COUNT  
NEG R1 ; GET ACTUAL WORD COUNT  
ASL R1 ; MULTIPLY BY 2  
MOV P, RBAL(R5), -(SP) ; STORE BUFFER ADDRESS  
ADD (SP), R1 ; CALCULATE FINAL ADDRESS  
MOV #FBLKT, R3 ; LOAD FREE BLOCK TABLE ADDRESS  
MOV FBLKC, R4 ; STORE FREE BLOCK COUNT  
BEQ 8$ ; ADD FREE BLOCK  
  
1$: CMP (R3)+, R1 ; COMPARE END OF BUFFER AND FREE BLOCK  
BEQ 10$ ; EQUAL MERGE BLOCKS  
BHI 5$ ; LESS THEN ADD FREE BLOCK  
CMP (R3)+, (SP) ; COMPARE BEGINNING OF BUFFER  
AND END OF FREE BLOCK  
BEQ 15$ ; EQUAL MERGE  
DEC R4 ; DECREMENT FREE BLOCK COUNT  
BNE 1$ ; GO TO NEXT FREE BLOCK  
BR 8$ ; GO ADD BLOCK  
  
5$: MOV R4, -(SP) ; PUT REMAINING COUNT ON STACK  
ASL (SP) ; MULTIPLY BY 4  
ASL (SP)  
TST -(R3) ; DECREMENT R3 BY 2  
ADD (SP)+, R3 ; CALCULATE OLD END OF FREE BLOCK TABLE  
MOV R3, R0  
ADD #4, R0 ; CALCULATE NEW END OF FREE BLOCK TABLE  
6$: MOV -(R3), -(R0) ; MOVE TABLE
```

```

5536 022756 014340          MOV    -(R3),-(R0)
5537 022760 005304          DEC    R4                ; DECREMENT FREE BLOCK COUNT
5538 022762 001374          BNE    6$                ; IF NOT ZERO, COUNT
5539
5540 022764 005237 001670      8$:   INC    FBLKC          ; INCREMENT FREE BLOCK COUNT
5541 022770 012623          MOV    (SP)+,(R3)+      ; STORE BEGINNING OF FREE BLOCK
5542 022772 010123          MOV    R1,(R3)+        ; STORE END OF FREE BLOCK
5543 022774 000423          BR     25$              ; RETURN
5544
5545 022776 012643      10$:  MOV    (SP)+,-(R3)      ; LOAD BEGINNING OF FREE BLOCK
5546 023000 010421          BR     25$              ; RETURN
5547
5548 023002 005726      15$:  TST    (SP)+            ; ADJUST STACK
5549 023004 005304          DEC    R4                ; DECREMENT FREE BLOCKS LEFT
5550 023006 001402          BEQ    16$              ; NO MORE FREE BLOCK ADJUST END
5551 023010 021301          CMP    (R3),R1         ; CHECK IF RELEASED BLOCK END=
5552                                ; FREE BLOCK BEGINNING
5553 023012 001402          BEQ    20$              ; YES, MERGE BLOCKS
5554 023014 010143      16$:  MOV    R1,-(R3)        ; LOAD END OF FREE BLOCK
5555 023016 000412          BR     25$              ; RETURN
5556
5557 023020 010300      20$:  MOV    R3,R0            ; STORE R3 FOR BUFFER MOVEMENT
5558 023022 005743          TST    -(R3)            ; ADJUST R3
5559 023024 005720          TST    (R0)+            ; ADJUST R0
5560 023026 006304          ASL    R4                ; MULTIPLY COUNT BY TWO
5561 023030 005304          DEC    R4                ; DECREMENT TRANSFER COUNT
5562 023032 012023      21$:  MOV    (R0)+,(R3)+      ; MOVE TABLE ENTRY
5563 023034 005304          DEC    R4                ; DECREMENT COUNT
5564 023036 001375          BNE    21$              ; IF NOT 0, CONTINUE MOVE
5565 023040 005337 001670      21$:  DEC    FBLKC          ; DECREMENT FREE BLOCK COUNT
5566
5567 023044 012600      25$:  MOV    (SP)+,R0         ; RESTORE R0
5568 023046 012601          MOV    (SP)+,R1         ; RESTORE R1
5569 023050 012603          MOV    (SP)+,R3         ; RESTORE R3
5570 023052 012604          MOV    (SP)+,R4         ; RESTORE R4
5571 023054 000207          RTS    PC                ; RETURN
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
5624
5625
5626
5627
5628

.SBTTL LOAD DATA BUFFER

```

*****
THE REGISTERS BE USED AS FOLLOWS:
REGISTER          USE
-----          ---
R5                ADDRESS OF PARAMETER BLOCK
R4                BUFFER ADDRESS
R3                PATTERN ADDRESS
R2                WORDS LEFT IN SECTOR
R1                WORD COUNT
R0                PATTERN COUNT

CALL JSR PC,LDDATA
RETURN

THE DATA WILL LOOK AS FOLLOWS:
-----*-----*-----*
CPU ID           PAT           PAT           WORD 0
NO. OF PAT. WORDS (N) NO. OF PAT. WORDS (N) WORD 1
                                RANDOM DATA  WORDS 2
                                TO N+1
-----*-----*-----*
                                ZERO FILL  WORDS N+2
                                TO 255
-----*-----*-----*
*****

```

```

023056 010446
023060 010346
023062 010246
023064 010146
023066 010046
023070 016504 000132
023074 016501 000130
023100 005401
023102 122765 000121 000001
023110 001005
023112 005024
023114 005301
023116 001375
023120 000137 023270
023124 116500 000121
023130 010046
023132 006316
023134 006316
023136 006316
023140 006316
023142 05001

```

```

LDDATA: 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 P.ABAL(R5),R4 ;GET BUS ADDRESS
        MOV P.RWC(R5),R1 ;STORE WORD COUNT
        NEG R1 ;GET POSITIVE NUMBER FOR WORD COUNT
        CMPB #RDATA,P.CMND(R5) ;CHECK IF READ DATA
        BNE 5$ ;NO, LOAD BUFFER
3$: CLR (R4)+ ;CLEAR BUFFER BEFORE READ
    DEC R1 ;CHECK IF DONE
    BNE 3$ ;NO, CONTINUE
    JMP 25$ ;RESTORE REGISTERS

5$: MOVB P.DPAT(R5),R0 ;STORE PATTERN COUNT
    MOV R0,-(SP) ;STORE PATTERN FOR FIRST WORD
    ASL (SP) ;MULTIPLY PATTERN BY 4 FOR
    ; REDUNACY
    ASL (SP)
    ASL (SP)
    BIS R0,(SP) ;STORE FIRST WORD OF SECTOR

```

5629	023144	113766	001764	000001	MOVB	CPUID,1(SP)	; ON STACK
5630	023152	006300			ASL	R0	; MULTIPLY DATA PATTERN BY 2
5631	023154	016003	004602		MOV	PATTAB(PO),R3	; LOAD DATA PATTERN ADDRESS
5632	023160	012700	000040		MOV	#32.,R0	; LOAD PATTERN COUNT
5633	023164	011624		11\$:	MOV	(SP),(R4)+	; STORE PATTERN AND CPU ID IN 1ST WORD
5634	023166	162701	000002		SUB	#2,R1	; DECREMENT WORD COUNT BY 2
5635	023172	022701	000376		CMP	#254.,R1	; CHECK IF REST OF SECTOR WILL BE WRITTEN
5636							
5637	023176	101007			BHI	12\$; NO USE R1 AS WORD COUNT
5638	023200	012746	000376		MOV	#254.,-(SP)	; STORE SECTOR COUNT = 254
5639	023204	000401			BR	13\$	
5640							
5641	023206	010146		12\$:	MOV	R1, -(SP)	; STORE REMAINING WORD COUNT
5642	023210	111666	000001	13\$:	MOVB	(SP),1(SP)	; ADD REDUNDACY
5643	023214	012624			MOV	(SP)+,(R4)+	; STORE WORD COUNT IN 2ND WORD
5644	023216	012702	000376		MOV	#254.,R2	; LOAD SECTOR WORD COUNT
5645							
5646	023222	012324		15\$:	MOV	(R3)+,(R4)+	; STORE DATA PATTERN
5647	023224	005301			DEC	R1	; DECREMENT WORD COUNT
5648	023226	001417			BEQ	R0	; BRANCH IF FINISHED
5649	023230	005302			DEC	R2	; DECREMENT SECTOR COUNT
5650	023232	005300			DEC	R0	; DECREMENT PATTERN COUNT
5651	023234	001407			BEQ	17\$; IF ZERO, INITIALIZE PATTERN
5652	023236	005702			TST	R2	; ARE WE AT END OF SECTOR
5653	023240	001370			BNE	15\$; NO, CONTINUE TO LOAD DATA PATTERN
5654	023242	162703	000074		SUB	#60.,R3	; INITIALIZE PATTERN ADDRESS
5655	023246	012700	000040		MOV	#32.,R0	; INITIALIZE PATTERN COUNT
5656	023252	000744			BR	11\$; START NEXT SECTOR
5657							
5658	023254	162703	000100	17\$:	SUB	#64.,R3	; INITIALIZE PATTERN ADDRESS
5659	023260	012700	000040		MOV	#32.,R0	; INITIALIZE PATTERN
5660	023264	000756			BR	15\$; REPEAT PATTERN
5661							
5662	023266	005726		20\$:	TST	(SP)+	; ADJUST STACK
5663	023270	012600		25\$:	MOV	(SP)+,R0	; RESTORE R0
5664	023272	012601			MOV	(SP)+,R1	; RESTORE R1
5665	023274	012602			MOV	(SP)+,R2	; RESTORE R2
5666	023276	012603			MOV	(SP)+,R3	; RESTORE R3
5667	023300	012604			MOV	(SP)+,R4	; RESTORE R4
5668	023302	000207			RTS	PC	; RETURN

!

```

5669          .SBTTL  FIND FIRST DATA ERROR
5670
5671 023304 032777 000002 155626  BUFER1: BIT      #SW1,@SWR      ;CHECK IF INHIBIT SOFTWARE DATA
5672          ;COMPARE
5673          BNE      SS      ;YES, RETURN
5674 023312 001017          MOV      SOFCMP,P.CMLG(R5) ;LOAD COMPARISON LENGTH
5675 023322 001413          BEQ      SS      ;IF NO MORE COMPARISONS, RETURN
5676 023324 016565 000010 000214  MOV      P.BALO(R5),P.ERHD(R5) ;LOAD ADDRESS OF START OF SECTOR
5677 023332 016565 000012 000222  MOV      P.WC(R5),P.BFCP(R5) ;LOAD TRANSFER LENGTH
5678 023340 005465 000222          NEG      P.BFCP(R5) ;MAKE IT POSITIVE IF POSITIVE
5679 023344 004737 023362          JSR      PC,CMPBUF ;DO COMPARISON
5680 023350 023356          10$      ;ERROR RETURN
5681
5682          SS:      TST      (R0)+ ;ADJUST R0 FOR RETURN
5683 023352 005720          RTS      R0 ;RETURN
5684 023354 000200
5685          10$:      MOV      (R0),R0 ;LOAD ERROR RETURN
5686 023356 011000          RTS      R0 ;RETURN
5687 023360 000200

```

.SBTTL COMPARE DATA BUFFER

```

*****
THE REGISTERS WILL BE USED AS FOLLOWS:
REGISTER          USE
-----          ---
R5                ADDRESS OF PARAMETER BLOCK
R4                BUFFER ADDRESS
R3                PATTERN ADDRESS
R2                SECTOR WORD COUNT
R1                WORDS ON SECTOR WRITTEN
R0                PATTERN COUNT
*****
*CALL JSR PC, CBUF
*      (ADDRESS OF COMPARISON ERROR RETURN)
*      RETURN
*      (SP) NUMBER OF SOFT DATA COMPARES
*      (SP+2) WORD COUNT
*****

```

```

5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5710
5711
5712 023362 010446
5713 023364 010346
5714 023366 010246
5715 023370 010146
5716 023372 010046
5717 023374 016504 000214
5718 023400 016546 000222
5719 023404 016546 000220
5720 023410 001002
5721 023412 000137 023646
5722
5723 023416 010465 000214
5724 023422 005002
5725 023424 012400
5726 023426 010003
5727 023430 042700 177760
5728 023434 042703 177417
5729 023440 006203
5730 023442 006203
5731 023444 006203
5732 023446 006203
5733 023450 020003
5734 023452 001066
5735 023454 005202
5736 023456 005366 000002
5737
5738 023462 006300
5739 023464 016003 004602
5740 023470 126414 000001
5741 023474 001055
5742 023476 012401

```

```

CMPBUF: MOV R4, -(SP) ;STORE R4
        MOV R3, -(SP) ;STORE R3
        MOV R2, -(SP) ;STORE R2
        MOV R1, -(SP) ;STORE R1
        MOV R0, -(SP) ;STORE R0
        MOV P.ERHD(R5), R4 ;GET START ADDRESS
        MOV P.BFCP(R5), -(SP) ;GET COMPARE COUNT
        MOV P.CMLG(R5), -(SP) ;LOAD NUMBER OF COMPARISON
        BNE 10$ ;COMPARE IF COMPARE COUNT NOT ZERO
        JMP 40$ ;ELSE RETURN

10$: MOV R4, P.ERHD(R5) ;LOAD SECTOR HEAD IN CASE OF ERROR
     CLR R2 ;CLEAR OFFSET FROM HEAD
     MOV (R4)+, R0 ;STORE 1ST WORD (PATTERN INDEX)
     MOV R0, R3
     BIC #177760, R0 ;KEEP LOW 4 BITS
     BIC #177417, R3 ;MASK CPU ID
     ASR R3 ;SHIFT 4 BITS RIGHT
     ASR R3
     ASR R3
     ASR R3
     CMP R0, R3 ;CHECK FOR PATTERN MATCH
     BNE 30$ ;NO REPORT ERROR
     INC R2 ;INCREMENT OFFSET
     DEC 2(SP) ;DECREMENT COMPARISON COUNT

13$: ASL R0 ;MULTIPLY INDEX BY 2
     MOV PATTAB(R0), R3 ;STORE PATTERN ADDRESS
     CMPB 1(R4), (R4) ;CHECK WORD COUNT
     BNE 30$ ;NO REPORT ERROR
     MOV (R4)+, R1 ;STORE WORDS IN SECTOR WRITTEN

```

5743	023500	042701	177400		BIC	#177400,R1	;KEEP LOW BYTE
5744	023504	005202			INC	R2	;INCREMENT OFFSET VALUE
5745	023506	005366	000002		DEC	2(SP)	;DECREMENT WORD COUNT
5746	023512	005765	000116		TST	P.SMPL(R5)	;CHECK IF SAMPLED COMPARE
5747	023516	001467			BEQ	50\$;YES, START SAMPLED COMPARE SEQUENCE
5748	023520	005701			TST	R1	;CHECK IF SECTOR ALL ZEROES
5749	023522	001426			BEQ	20\$;YES GO TO ZERO FILL LOOP
5750	023524	012700	000040		MOV	#32.,R0	;LOAD PATTERN COUNT
5751	023530	022324		15\$:	CMP	(R3)+,(R4)+	;CHECK PATTERN
5752	023532	001036			BNE	30\$;MISCOMPARE ERROR
5753	023534	005366	000002		DEC	2(SP)	;DECREMENT WORD COUNT
5754	023540	001442			BEQ	40\$;NO MORE COMPARISONS
5755	023542	005316			DEC	(SP)	;DECREMENT NUMBER OF SOFT
5756							DATA COMPARE
5757	023544	001440			BEQ	40\$;NO MORE COMPARISONS
5758	023546	005202			INC	R2	;INCREMENT OFFSET
5759	023550	022702	000400		CMP	#256.,R2	;CHECK IF COMPLETE SECTOR COMPARES
5760	023554	001720			BEQ	10\$;YES, START NEXT SECTOR
5761	023556	005301			DEC	R1	;DECREMENT NUMBER OF WORDS WRITTEN
5762	023560	001407			BEQ	20\$;IF ZERO, GO TO ZERO FILL LOOP
5763	023562	005300			DEC	R0	;DECREMENT PATTERN COUNT
5764	023564	001361			BNE	15\$;NEXT WORD
5765	023566	012700	000040		MOV	#32.,R0	;REINITIALIZE PATTERN COUNT
5766	023572	162703	000100		SUB	#64.,R3	;REINITIALIZE PATTERN ADDRESS
5767	023576	000754			BR	15\$;CHECK NEXT WORD
5768							
5769	023600	005724		20\$:	TST	(R4)+	;CHECK ZERO FILL OF PATTERN
5770	023602	001012			BNE	30\$;MISCOMPARE ERROR
5771	023604	005366	000002		DEC	2(SP)	;DECREMENT WORD COUNT
5772	023610	001416			BEQ	40\$;NO MORE COMPARISONS
5773	023612	005316			DEC	(SP)	;DECREMENT NUMBER OF SOFT
5774							DATA COMPARES
5775	023614	001414			BEQ	40\$;NO MORE COMPARISONS
5776	023616	005202			INC	R2	;INCREMENT OFFSET VALUE
5777	023620	022702	000400		CMP	#256.,R2	;CHECK IF COMPLETE SECTOR COMPARED
5778	023624	001674			BEQ	10\$;YES, START NEXT SECTOR
5779	023626	000764			BR	20\$;COMPARE NEXT WORD ON
5780							
5781	023630	110265	000217	30\$:	MOVB	R2,P.ERAD(R5)	;STORE OFFSET ADDRESS
5782	023634	022626			CMP	(SP)+,(SP)+	;ADJUST STACK
5783	023636	017666	000012 000012		MOV	#12(SP),12(SP)	;LOAD ERROR RETURN
5784	023644	000404			BR	45\$;RETURN
5785							
5786	023646	022626		40\$:	CMP	(SP)+,(SP)+	;ADJUST STACK
5787	023650	062766	000002 000012		ADD	#2,12(SP)	;ADJUST RETURN
5788							
5789	023656	105065	000117	45\$:	CLRB	P.ECMP(R5)	;CLEAR ECC COMPARE FLAG
5790	023662	012600			MOV	(SP)+,R0	;RESTORE R0
5791	023664	012601			MOV	(SP)+,R1	;RESTORE R1
5792	023666	012602			MOV	(SP)+,R2	;RESTORE R2
5793	023670	012603			MOV	(SP)+,R3	;RESTORE R3
5794	023672	012604			MOV	(SP)+,R4	;RESTORE R4
5795	023674	000207			RTS	PC	;RETURN
5796							
5797	023676	005701		50\$:	TST	R1	;CHECK IF ALL ZEROES
5798	023700	001432			BEQ	52\$;YES, TEST FOR ZEROES

5855	024134	101644				BLOS	40\$;CHECK IF FINISHED
5856	024136	062704	000022			ADD	#22,R4		;CALCULATE START OF NEXT SECTOR
5857	024142	000137	023416			JMP	10\$;GO SERVICE NEXT SECTOR
5858									
5859	024146	162766	000043	000002	60\$:	SUB	#35.,2(SP)		;SUBTRACT 35 FROM WORD COUNT
5860	024154	10.334				BLOS	40\$;CHECK IF FINISHED
5861	024156	062702	000043			ADD	#35.,R2		;DETERMINE OFFSET
5862	024162	062704	000106			ADD	#106,R4		;DETERMINE ADDRESS OF COMPARE
5863	024166	005714				TST	(R4)		;CHECK IF DATA CORRECT
5864	024170	001217				BNE	30\$;NO, REPORT ERROR
5865	024172	005316				DEC	(SP)		;DECREMENT NUMBER OF COMPARES
5866	024174	001624				BEQ	40\$;BRANCH IF FINISHED
5867	024176	005300				DEC	R0		;CHECK IF SAMPLE COUNT EXHAUSTED
5868	024200	001362				BNE	60\$;NO, CHECK NEXT SAMPLE
5869	024202	162766	000011	000002		SUB	#11,2(SP)		;ADJUST WORD COUNT
5870	024210	101616				BLOS	40\$;CHECK IF FINISHED
5871	024212	062704	000022			ADD	#22,R4		;CALCULATE START OF NEXT SECTOR
5872	024216	000137	023416			JMP	10\$;GO SERVICE NEXT SECTOR

.SBTTL NORMAL COMMAND RETURN

```

*****
* THIS ROUTINE WILL HANDLE ALL NORMAL COMMAND TERMINATIONS
*
* CALLED ROUTINE          USE
* -----
*
* ASNORM                  ASSIGNMENT NORMAL TERNIMATION
* CNTRL1                  CONTROLLER ERROR
* ABNORM                  DRIVE ERROR
* EYCV 1                  NORMAL ERROR RECOVERY
* DATE 3R                 INCORRECT DATA BUFFER
*
* ASSUMED REGISTERS
*
* REGISTER                CONTENTS
* -----
*
* R5                      ADDRESS OF PARMETER BLOCK
* R2                      ADDRESS OF RK06 REGISTERS
*
*****

```

```

5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898 024222 004737 047364
5899 024226 032765 024000 000016
5900 024234 001010
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911 024236 032765 177400 000020
5912 024244 001004
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925 024246 032765 131761 000034
5926 024254 001405
5927 024256 052737 000001 001564
5928 024264 000137 027164

```

```

NORMAL: JSR PC I,CSTS :GATHER CONTROLLER STATUS
        BIT #SPAR:CTO,P.CS1(R5) ;CHECK IF DRIVE BUS PARITY OR CONTROLLER TIME OUT
        BNE 25 ;YES, INDICATE CONTROLLER FAULT

        CHECK IF ANY OF THE FOLLOWING ERRORS ARE SET IN CS2
        UNIT FIELD ERROR
        MULTIPLE DRIVE SELECT
        PROGRAMMING ERROR
        NON-EXISTENT MEMORY
        NON-EXISTENT DRIVE
        UNIBUS PARITY ERROR
        WRITE CHECK ERROR
        DATA LATE
        BIT #UFE:MDS:PGE:NEM:NED'UPE:WCE:DLT,P.CS2(R5)
        BNE 25 ;YES, INDICATE CONTROLLER ERROR

        CHECK IF ANY OF THE FOLLOWING ERRORS ARE SET IN RYER
        ILLEGAL FUNTION CODE
        FORMAT ERROR
        DRIVE TYPE ERROR
        ECC HARD
        BAD SECTOR ERROR
        HEADER VRC ERROR
        CYLINDER ADDRESS OVERFLOW
        DRIVE TIMING ERROR
        OPERATION INCOMPLETE
        DATA CHECK
        BIT #ILC:FMTE:DTYE:ECH:BSE!HVRC!COE:DTE!OPI:DCK,P.ER(R5)
        BEQ 35 ;NO CHECK FOR UNREPORT DRIVE ERRORS
        BIS #BIT0,ERCONT ;SET CONTROLLER ERROR NOT FLAGGED
        JMP CNTRL1 ;REPORT CONTROLLER ERROR

```

```

5929
5930      :      CHECK THE FOLLOWING ERROR BITS IN RKER(UNREPORTED DRIVE ERROR,
5931      :      SEEK INCOMPLETE
5932      :      NON-EXECUTABLE DRIVE FUNCTION
5933      :      DRIVE DETECTED DRIVE BUS PARITY ERROR
5934      :      INVALID DISK ADDRESS
5935      :      WRITE LOCK ERROR
5936      :      UNSAFE
5937 024270 032765 046016 000034 3$: BIT   #SKI!NXF!DRPAR!IDAE!WLE!UNS,P.ER(R5)
5938 024276 001004      :      BNE   4$      ;YES,SET UNREPORTED DRIVE ERROR
5939
5940      :      CHECK THE FOLLOWING ERROR BITS IN RKDS
5941      :      AC LOW
5942      :      SPEED LOSS
5943      :      DRIVE OFF TRACK
5944 024300 032765 000070 000036      BIT   #ACLO!SPDLSS!DROT,P.DS(R5)
5945 024306 001405      BEQ   5$      ;NO,CHECK IF SPECIAL SEQUENCE
5946 024310 052765 000010 000212 4$: BIS   #BIT3,P.ERR(R5) ;SET DRIVE ERROR NOT REPORTED
5947 024316 000137 025312      JMP   ABNORM ;PROCESS ERROR
5948
5949 024322 032765 000001 000014 5$: BIT   #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
5950 024330 001005      BNE   6$      ;YES,CHECK IF SPECIAL SEQUENCE
5951 024332 052765 000200 000212      BIS   #BIT7,P.ERR(R5) ;SET UNSOLICITED ATTENTION
5952 024340 000137 025312      JMP   ABNORM ;PROCESS ERROR
5953
5954 024344 005737 001564      6$: TST   ERCONT ;CHECK IF CONTROLLER ERROR HAS OCCURRED
5955 024350 100017      BPL   11$     ;NO,CONTINUE NORMAL PROCESSING
5956 024352 122765 000121 000001      CMPB  #RODATA,P.CMND(R5) ;CHECK IF DATA TRANSFER COMMAND
5957 024360 101007      BHI   7$      ;NO,REISSUE COMMAND
5958 024362 132765 000040 000001      BITB  #BITS,P.CMND(R5) ;CHECK IF SPECIAL COMMAND
5959 024370 001003      BNE   7$      ;YES,REISSUE COMMAND
5960 024372 004037 025032      JSR   RO,CHKADD ;CHECK BUS AND DISK ADDRESS
5961 024376 024406      JSR   10$     ;ERROR RETURN
5962 024400 004037 051406      7$: JSR   RO,Q.PUSH ;ENQUEUE PARAMETER BLOCK IN
5963 024404 001324      CINITQ ;COMMAND INITIATION QUEUE
5964 024406 000207      10$: RTS   PC ;RETURN
5965
5966 024410 005765 000150      11$: TST   P.DSTT(R5) ;CHECK IF RETRY SEQUENCE
5967 024414 001417      BEQ   32$     ;NO,CHECK IF DRIVE IS BEING ASSIGNED
5968 024416 004737 040262      JSR   PC,ERCVY1 ;GO THROUGH ERROR RECOVERY
5969 024422 005737 001566      20$: TST   ERAPRO ;CHECK IF ERROR PROCESSING COMPLETE
5970 024426 001367      BNE   10$     ;NO,RETURN
5971 024430 004037 051466      JSR   RO,Q.POP ;GET NEXT ERROR TO BE PROCESSED
5972 024434 001330      ERRPRQ
5973 024436 012605      MOV   (SP)+,R5 ;LOAD ADDRESS OF PARAMETER BLOCK FOR
5974      :      NEXT ERROR
5975 024440 001762      BEQ   10$     ;IF NO MORE ERRORS,RETURN
5976 024442 010537 001566      MOV   R5,ERRPRO ;INDICATE THAT ERROR IS BEING PROCESSED
5977 024446 004737 032002      JSR   PC,ERCVY ;GO THROUGH ERROR RECOVERY
5978 024452 000763      BR   20$     ;CHECK IF ERROR PROCESSING COMPLETE
5979
5980 024454 105765 000210      32$: TSTB  P.ASSN(R5) ;CHECK IF DRIVE IS BEING ASS'GNED
5981 024460 001402      BEQ   33$     ;NO,CHECK IF SEEK OR RELEASE
5982 024462 000137 013504      JMP   ASNORM ;GO CONTINUE ASSIGNMENT
5983
5984 024466 122765 000117 000001 33$: CMPB  #SEEK,P.CMND(R5) ;CHECK IF SEEK TO PREVIOUS POSITION

```

```

5985                                     ; (SWITCH 14 OPTION)
5986 024474 001416 BEQ 34$ ; YES, RELEASE DRIVE
5987 024476 122765 000140 000001 CMPB #RELEAS,P.CMND(R5) ; CHECK IF RELEASE COMMAND
5988 024504 001021 BNE 35$ ; NO, CHECK ADDRESS AND WORD COUNT
5989 024506 112737 177777 001506 MOVB #-1,I.ISRL ; INDICATE THAT AN INTERRUPT HAS BEEN ISSUED
5990 024514 012762 000300 000000 MOV #INTR,RKCS1(R2) ; FORCE AN INTERRUPT SCAN DRIVE ATTENTIONS
5991 024522 004037 051406 JSR RO,G.PUSH ; PUT PARAMTER BLOCK ADDRESS ON
5992 024526 001314 AVAILQ ; DRIVE AVAILIABLE QUEUE
5993 024530 000207 RTS PC ; RETURN
5994
5995 024532 112765 000140 000001 34$: MOVB #RELEAS,P.CMND(R5) ; PUT RELEASE COMMAND IN PARAMETER BLOCK
5996 024540 004037 051406 JSR RO,G.PUSH ; ENQUEUE PARAMETER BLOCK IN
5997 024544 001324 CINITQ ; COMMAND INITIATION QUEUE
5998 024546 000207 RTS PC ; RETURN
5999
6000 024550 004037 025032 35$: JSR RO,CHKADD ; CHECK WORD COUNT, BUS ADDRESS,
6001 ; SECTOR TRACK AND CYLINDER
6002 024554 024406 10$ ; ERROR RETURN ADDRESS
6003 024556 112737 177777 001506 MOVB #-1,I.ISRL ; INDICATE THAT AN INTERRUPT HAS BEEN ISSUED
6004 024564 012762 000300 000000 MOV #INTR,RKCS1(R2) ; GO SCAN FOR DRIVE INTERRUPTS
6005 024572 005037 177776 CLR PS ; ALLOW RK06 INTERRUPTS
6006 024576 122765 000121 000001 CMPB #RDDATA,P.CMND(R5) ; CHECK IF READ DATA
6007 024604 001004 BNE NORM1 ; NO, DO NOT COMPARE BUFFER
6008 024606 004037 023304 JSR RO,BUFER1 ; CHECK DATA READ
6009 024612 025424 DATERR ; ERROR RETURN
6010 024614 000417 BR NORM2 ; RELEASE BUFFER
6011
6012 024616 122765 000131 000001 NORM1: CMPB #WRCHK,P.CMND(R5) ; CHECK IF WRITE COMMAND
6013 024624 001013 BNE NORM2 ; NO, RELEASE BUFFER
6014 024626 032765 000200 000014 BIT #W.WCK,P.PRST(R5) ; CHECK IF WRITE CHECK ISSUED
6015 024634 001407 BEQ NORM2 ; YES, RELEASE BUFFER
6016 024636 042765 000200 000014 BIC #W.WCK,P.PRST(R5) ; CLEAR WRITE FLAG
6017 024644 004037 051406 JSR RO,G.PUSH ; PUT PARAMETER BLOCK IN
6018 024650 001324 CINITQ ; COMMAND INITIATION QUEUE
6019 024652 000207 RTS PC ; RETURN
6020
6021 024654 013737 001452 177776 NORM2: MOV RKPRI,PS ; LOCK OUT RK06 INTERRUPTS
6022 024662 004737 022642 JSR PC,BUFREL ; RELEASE BUFFER
6023 024666 005037 177776 CLR PS ; ALLOW RK06 INTERRUPTS
6024 024672 032777 040040 154240 BIT #SWS!SW14,DSWR ; CHECK IF INHIBIT AUTOMATIC DESELECT
6025 024700 001043 BNE 33$ ; YES, PUT PARAMETER BLOCK IN
6026 ; AVAILIABLE QUEUE
6027 024702 026565 000110 000154 CMP P.MXCD+2(R5),P.NODR+2(R5) ; CHECK IF MAXIMUM NUMBER
6028 ; OF COMMAND HAVE BEEN ISSUED
6029 024710 101010 BHI 29$ ; NO, CHECK IF MAXIMUM NUMBER OF
6030 ; OF WORDS HAVE BEEN TRANSFERRED
6031 024712 103404 BLO 29$ ; YES, DROP DRIVE
6032 024714 026565 000106 000152 CMP P.MXCD(R5),P.NODR(R5) ; CHECK IF MAXIMUM NUMBER
6033 ; OF COMMANDS HAVE BEEN ISSUED
6034 024722 103003 BHIS 29$ ; NO, CHECK IF MAXIMUM NUMBER OF
6035 ; HAVE BEEN TRANSFERRED
6036 024724 004737 013004 28$: JSR PC,DROP ; DROP DRIVE
6037 024730 000435 BR 35$ ; ALLOCATE BUFFER
6038
6039
6040 024732 016546 000166 29$: MOV P.NRD+2(R5),-(SP) ; STORE NUMBER OF WORDS READ ON STACK

```

6041	024736	016546	000170			MOV	P.WRT+4(R5),-(SP)	
6042	024742	066566	000160	000002		ADD	P.WRT+2(R5),2(SP) ;ADD NUMBER OF WORDS WRITTEN	
6043	024750	005516				ADC	(SP)	
6044	024752	066516	000162			ADD	P.WRT+4(R5),(SP)	
6045	024756	103410				BCS	3C\$;DROP DRIVE IF OVERFLOW
6046	024760	026526	000114			CMP	P.MXWT+2(R5),(SP)+	;CHECK IF MAXIMUM NUMBER OF WORDS HAVE BEEN TRANSFERRED
6047								;NO, PUT PARAMETER BLOCK IN AVAILABLE QUEUE
6048	024764	101010				BHI	32\$	
6049								
6050	024766	103405				BLO	31\$;YES, DROP DRIVE
6051	024770	026526	000112			CMP	P.MXWT(R5),(SP)+	;CHECK IF MAXIMUM NUMBER OF WORDS HAVE BEEN TRANSFERRED
6052								;YES, DROP DRIVE
6053	024774	103753				BLO	28\$;NO, PUT PARAMETER BLOCK IN AVAILABLE QUEUE
6054	024776	000404				BR	33\$	
6055								
6056								
6057	025000	005726			30\$:	TST	(SP)+	;ADJUST STACK
6058	025002	005726			31\$:	TST	(SP)+	;ADJUST STACK
6059	025004	000747				BR	28\$;DROP DRIVE
6060								
6061	025006	005726			32\$:	TST	(SP)+	;ADJUST STACK
6062	025010	112765	000140	000001	33\$:	MOVB	#RELEAS,P.CMND(R5)	;PUT RELEASE COMMAND IN PARAMETER BLOCK
6063	025016	004037	051406			JSR	RO,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN COMMAND INITIATION QUEUE
6064	025022	001324				CINITQ		
6065								
6066	025024	004737	022342		35\$:	JSR	PC,GETBUF	;GET NEW BUFFER
6067	025030	000207				RTS	PC	;RETURN

```

        .SBTTL CHECK BUS ADDRESS, WORD COUNT, SECTOR, TRACK, AND CYLINDER
6068
6069
6070 025032 005765 000022      CHKADD: TST      P.WCR(R5)      ;CHECK IF WORD COUNT ZERO
6071 025036 001404              BEQ          6$      ;YES, CHECK BUS ADDRESS
6072 025040 052737 000002 001564      BIS          #BIT1,ERCONT ;SET WORD COUNT NOT EQUAL ZERO
6073 025046 000512              BR           24$      ;REPORT ERROR
6074
6075 025050 032765 001400 000016 6$:      BIT          #BA16!BA17,P.CS1(R5) ;CHECK IF HIGH BUS ADDRESS BITS SET
6076 025056 001011              BNE          10$      ;YES, REPORT ERROR
6077 025060 016546 000012      MOV          P.WC(R5),-(SP) ;STORE WORD COUNT
6079 025064 005416              NEG          (SP)      ;MAKE IT POSITIVE
6079 025066 006316              ASL         (SP)      ;DETERMINE NUMBER OF BITS
6080 025070 066516 000010      ADD          P.BALO(R5),(SP) ;DETERMINE EXPECTED BUS ADDRESS
6081 025074 026526 000024      CMP          P.BAR(R5),(SP)+ ;CHECK BUS ADDRESS
6082 025100 001404              BEQ          17$      ;BUS ADDRESS CORRECT--CHECK CYLINDER,
6083                                     ;TRACK, AND SECTOR
6084 025102 052737 000004 001564 10$:     BIS          #BIT2,ERCONT ;SET BUS ADDRESS INCORRECT
6085 025110 000471              BR           24$      ;REPORT ERROR
6086
6087 025112 016546 000012      17$:      MOV          P.WC(R5),-(SP) ;STORE WORD COUNT
6088 025116 005416              NEG          (SP)      ;MAKE IT POSITIVE
6089 025120 105716              TSTB        (SP)      ;CHECK IF EVEN NUMBER OF SECTORS
6090 025122 001402              BEQ          18$      ;YES, DO COMPARISON
6091 025124 105266 000001      INCB        1(SP)      ;INCREMENT SECTOR COUNT
6092 025130 005046              CLR         -(SP)      ;MAKE ROOM ON STACK
6093 025132 116616 000003      18$:      MOVB        3(SP),(SP) ;STORE NUMBER OF SECTORS TRANSFERRED
6094 025136 005066 000002      CLR         2(SP)      ;CLEAR LOCATION ON STACK
6095 025142 116566 000004 000002      MOVB        P.SECT(R5),2(SP) ;STORE STARTING SECTOR
6096 025150 066616 000002      ADD         2(SP),(SP) ;DETERMINE FINAL SECTOR ADDRESS
6097 025154 005066 000002      CLR         2(SP)      ;CLEAR NUMBER OF TRACKS TRANSFERRED
6098
6099 025160 022716 000026      19$:      CMP          #22,(SP) ;CHECK FOR SECTOR OVERFLOW
6100 025164 101005              BHI          20$      ;NO, CHECK IF SECTOR CORRECT
6101 025166 162716 000026      SUB         #22,(SP) ;DECREMENT SECTOR COUNT BY 22 AND
6102 025172 005266 000002      INC         2(SP)      ;INCREMENT TRACKS TRANSFERRED
6103 025176 000770              BR           19$      ;CHECK FOR SECTOR OVERFLOW
6104
6105 025200 126526 000026      20$:      CMPB        P.DTS(R5),(SP)+ ;CHECK IF FINAL SECTOR CORRECT
6106 025204 001027              BNE          23$      ;NO, REPORT ERROR
6107 025206 005046              CLR         -(SP)      ;MAKE ROOM FOR TRACKS TRANSFERRED
6108 025210 116516 000005      MOVB        P.TRACK(R5),(SP) ;STORE STARTING TRACK
6109 025214 066616 000002      ADD         2(SP),(SP) ;DETERMINE FINAL TRACK ADDRESS
6110 025220 005066 000002      CLR         2(SP)      ;CLEAR FINAL CYLINDER
6111
6112 025224 122716 000003      21$:      CMPB        #3,(SP) ;CHECK FOR TRACK OVERFLOW
6113 025230 101005              BHI          22$      ;NO, CHECK FINAL TRACK
6114 025232 162716 000003      SUB         #3,(SP) ;DECREMENT TRACK COUNT BY 3 AND
6115 025236 005266 000002      INC         2(SP)      ;INCREMENT CYLINDER COUNT
6116 025242 000770              BR           21$      ;CHECK FOR TRACK OVERFLOW
6117
6118 025244 126526 000027      22$:      CMPB        P.DTS+1(R5),(SP)+ ;CHECK IF FINAL TRACK CORRECT
6119 025250 001005              BNE          23$      ;TRACK INCORRECT ERROR
6120 025252 066516 000002      ADD         P.CYLN(R5),(SP) ;CALCULATE FINAL CYLINDER
6121 025256 026516 000030      CMP         P.DCYL(R5),(SP) ;CHECK IF CYLINDER CORRECT
6122 025262 001410              BEQ          25$      ;CORRECT GO TO NORMAL RETURN
6123 025264 005726              23$:      TST         (SP)+ ;ADJUST STACK

```

E11

CZR6PC0 RK611/06 PERF EXEC
CZR6PC P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 136
CHECK BUS ADDRESS, WORD COUNT, SECTOR, TRACK, AND CYLINDER

SEQ 0134

6124	025266	052737	000010	001564		BIS	#BIT3,ERCONT	;SET INCORRECT SECTOR, TRACK, OR CYLINDER
6125	025274	004737	027164		24\$:	JSR	PC,CNTRL1	;REPORT ERROR
6126	025300	011000				MOV	(R0),R0	;LOAD R0 FOR ERROR RETURN
6127	025302	000200				RTS	R0	;RETURN
6128								
6129	025304	005726			25\$:	TST	(SP)+	;ADJUST STACK
6130	025306	005720				TST	(R0)+	;ADJUST R0
6131	025310	000200				RTS	R0	;NORMAL RETURN


```

        .SBTTL  ABNORMAL TERMINATION
6132
6133
6134 025312 004737 051626  ABNORM: JSR      PC,Q.RMCV      ;REMOVE PARAMETER BLOCK FROM COMMAND
6135                                     ;INITIATION QUEUE
6136 025316 032765 000100 000014  BIT      #CMDTO,P.PRST(R5) ;CHECK IF COMMAND TIME OUT
6137 025324 001402                                     BEQ      3$      ;NO, PROCESS ABNORMAL TERMINATION
6138 025326 000137 025516  JMP      TIMOUT   ;JUMP TO TIME OUT ROUTINE
6139
6140 025332 005737 001564  3$:   TST      ERCONT   ;CHECK IF CONTROLLER ERROR HAS OCCURRED
6141 025336 100010                                     BPL      10$    ;NO, CONTINUE ERROR PROCESSING
6142 025340 032765 000001 000014  BIT      #DRVUSE,P.PRST(R5) ;CHECK FOR UNSOLICATED INTERRUPT
6143 025346 001403                                     BEQ      5$      ;YES, DO NOT ENQUEUE PARAMETER BLOCK
6144 025350 004037 051406  JSR      RD,Q.PUSH   ;ENQUEUE PARAMETER BLOCK IN
6145 025354 001324                                     CINITQ    ;COMMAND INITIATION QUEUE
6146 025356 000207  5$:   RTS      PC      ;RETURN
6147
6148 025360 032765 000001 000014  10$:  BIT      #DRVUSE,P.PRST(R5) ;CHECK IF DRIVE IN USE
6149 025366 001004                                     BNE      ABN000 ;YES, CHECK IF DRIVE BEING ASSIGNED
6150 025370 052765 000200 000212  BIS      #BIT7,P.ERR(R5) ;INDICATE UNSOLICATED INTERRUPT
6151 025376 000420                                     BR       ABN001  ;CHECK IF ERROR IS BEING PROCESSED
6152
6153 025400 005765 000150  ABN000: TST      P.DSTT(R5) ;CHECK IF CURRENTLY UNDERGOING
6154                                     ;ERROR PROCESSING
6155 025404 001415                                     BEQ      ABN001  ;NO, CHECK IF ERROR PRESENTLY BEING PROCESSED
6156 025406 100402                                     BMI      13$    ;ERROR ENQUEUED
6157 025410 000137 043060  JMP      ERCVY2    ;ERROR RECOVERY ROUTINE
6158
6159 025414 052765 000100 000212  13$:  BIS      #BIT6,P.ERR(R5) ;SET ERROR WHILE ENQUEUED IN ERROR QUEUE
6160 025422 000207                                     RTS      PC      ;RETURN
6161
6162 025424 013737 001452 177776  DATERR: MOV      RKPRI,PS ;LOCK RK06 INTERRUPTS
6163 025432 052765 000004 000212  BIS      #BIT2,P.ERR(R5) ;SET SOFTWARE DETECTED DATA ERROR
6164
6165 025440 012765 100200 000150  ABN001: MOV      #BIT15!BIT7,P.DSTT(R5) ;SET ERROR ENQUEUED AND FIRST ERROR
6166 025446 005737 001566                                     TST      ERRPRO  ;CHECK IF ERROR IS BEING PROCESSED
6167 025452 001015                                     BNE      5$      ;YES, GO ENQUEUE ERROR
6168 025454 010537 001566  2$:   MOV      R5,ERRPRO ;LOAD ERROR BEING PROCESSED
6169 025460 004737 032002  JSR      PC,ERCVY  ;START ERROR RECOVERY
6170 025464 005737 001566  TST      ERAPRO   ;CHECK IF ERROR PROCESSING THROUGH
6171 025470 001005                                     BNE      3$      ;NO, RETURN
6172 025472 004037 051466  JSR      RD,Q.POP  ;GET NEXT ERROR ENQUEUED
6173 025476 001330  ERRPRQ
6174 025500 012605  MOV      (SP)+,R5  ;LOAD R5 FOR ERROR PROCESSING
6175 025502 001364                                     BNE      2$      ;IF NOT ZERO PROCESS NEXT ERROR
6176 025504 000207  3$:   RTS      PC      ;RETURN
6177
6178 025506 004037 051406  5$:   JSR      RD,Q.PUSH ;PUT PARAMETER BLOCK ON
6179 025512 001330                                     ERRPRQ    ;ERROR PROCESSING QUEUE
6180 025514 000207  RTS      PC      ;RETURN

```

```

        .SBTTL  COMMAND TIME OUT ROUTINE
TIMOUT:  MOV      RKCS2(R2), -(SP) ;GET CS2 FOR DRIVE NUMBER
        BIC      #1<DRVM^K>, (SP) ;KEEP DRIVE NUMBER
        CMPB     (SP)+, P.DRVN(R5) ;CHECK IF TIME OUT IF FOR
        ;        CURRENTLY SELECTED DRIVE
        BNE      5$ ;NO, TEST IF LOOPING ON CONTROLLER ERROR
        MOV      RKCS1(R2), T.CS1 ;GET COMMAND AND STATUS REGISTER 1
        BIT      #GO, T.CS1 ;CHECK IF GO IS STILL SET
        BEQ      5$ ;NO, TEST IF LOOPING ON CONTROLLER ERROR
        BIS      #E.CMTO, E.CONT ;SET CONTROLLER TIMED OUT ON COMMAND
        JMP      CONTRL ;GO REPORT CONTROLLER ERROR

        5$:     TST      ERCONT ;CHECK IF CYCLING ON CONTROLLER ERROR
        BPL      10$ ;NO, CHECK IF DRIVE IS CURRENTLY
        ;        SEIZED BY OTHER PORT
        JSR      RO, Q.PUSH ;REQUEUE COMMAND IN COMMAND
        ;        INITIATION QUEUE
        RTS      PC ;RETURN

        10$:    BIT      #DRVSZD, P.PRST(R5) ;CHECK IF DRIVE WAS SEIZED
        BEQ      12$ ;NO, CONTINUE
        BIS      #BIT4, P.ERR(R5) ;SET TIME OUT BECAUSE DRIVE SEIZED
        BR       ABN000 ;GO INITIATE ERROR PROCESSING

        12$:    BIT      #E.UNLD, P.PRST(R5) ;CHECK IF WAITING FOR HEADS TO RELOAD
        BEQ      15$ ;NO, SET TIME OUT FOR POSITIONING
        BIS      #BIT8, P.ERR(R5) ;SET TIME OUT IN RELOADING HEADS
        BR       ABN000 ;GO INITIATE ERROR PROCESSING

        15$:    BIS      #BIT5, P.ERR(R5) ;SET TIME OUT DURING POSITIONING
        BR       ABN000 ;GO INITIATE ERROR PROCESSING

```

6213
6214
6215 025652 010346
6216 025654 010146
6217 025656 012703 001410
6218 025662 012701 001430
6219 025666 104401 064410
6220 025672 004737 026132
6221 025676 012701 001444
6222 025702 104401 064507
6223 025706 004737 026132
6224 025712 012601
6225 025714 012603
6226 025716 000207
6227
6228
6229
6230 025720 010346
6231 025722 104401 064567
6232 025726 016546 000036
6233 025732 004737 052774
6234 025736 104401 064576
6235 025742 010503
6236 025744 062703 000040
6237 025750 012346
6238 025752 004737 052774
6239 025756 104401 060530
6240 025762 012346
6241 025764 004737 052774
6242 025770 104401 064637
6243 025774 012346
6244 025776 004737 052774
6245 026002 104401 060530
6246 026006 012346
6247 026010 004737 052774
6248 026014 104401 064652
6249 026020 012346
6250 026022 004737 052774
6251 026026 104401 060530
6252 026032 012346
6253 026034 004737 052774
6254 026040 104401 064665
6255 026044 012346
6256 026046 004737 052774
6257 026052 104401 060530
6258 026056 012346
6259 026060 004737 052774
6260 026064 104401 001165
6261 026070 012603
6262 026072 000207

.SBTTL PRINT RK06 UNIBUS REGISTERS

PRIREG: MOV R3,-(SP) ;STORE R3 ON STACK
MOV R1,-(SP) ;STORE R1 ON STACK
MOV #T.CS1,R3 ;LOAD BEGINNING OF REGISTER STORAGE
MOV #T.OS,R1 ;LOAD END OF REGISTER STORAGE
TYPE ERR300 ;PRINT REGISTER NAMES
JSR PC,PRSTAT ;PRINT REGISTER CONTENTS
MOV #T.DB,R1 ;LOAD END OF REGISTER STORAGE
TYPE ERR301 ;PRINT REGISTER NAMES
JSR PC,PRSTAT ;PRINT REGISTER CONTENTS
MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R3 ;RESTORE R3
RTS PC ;RETURN

.SBTTL PRINT DRIVE STATUS

PRDSTT: MOV R3,-(SP) ;STORE R3 ON STACK
TYPE ERR302 ;PRINT REGISTER NAMES
MOV #P.DS(R5),-(SP) ;CONVERT DRIVE STATUS REG TO OCTAL
JSR PC,BINOC
TYPE ERR303 ;PRINT HEADER
MOV R5,R3 ;DETERMINE START OF PRINT OUT
ADD #P.A00,R3
MOV (R3)+,-(SP) ;PRINT MESSAGE A
JSR PC,BINOC
TYPE BLANKS
MOV (R3)+,-(SP) ;PRINT MESSAGE B
JSR PC,BINOC
TYPE ERR304 ;PRINT MESSAGE A
MOV (R3)+,-(SP) ;PRINT MESSAGE B
JSR PC,BINOC
TYPE BLANKS
MOV (R3)+,-(SP) ;PRINT MESSAGE A
JSR PC,BINOC
TYPE ERR305 ;PRINT MESSAGE B
MOV (R3)+,-(SP) ;PRINT MESSAGE A
JSR PC,BINOC
TYPE BLANKS
MOV (R3)+,-(SP) ;PRINT MESSAGE B
JSR PC,BINOC
TYPE ERR306 ;PRINT MESSAGE A
MOV (R3)+,-(SP) ;PRINT MESSAGE B
JSR PC,BINOC
TYPE BLANKS
MOV (R3)+,-(SP) ;PRINT MESSAGE B
JSR PC,BINOC
TYPE SCRLF
MOV (SP)+,R3 ;RESTORE R3
RTS PC ;RETURN

6263
6264
6265 026074 010346
6266 026076 010146
6267 026100 010503
6268 026102 010501
6269 026104 062703 000016
6270 026110 062701 000036
6271 026114 104401 064410
6272 026120 004737 026132
6273 026124 012601
6274 026126 012603
6275 026130 000207
6276
6277
6278
6279 026132 012346
6280 026134 004737 052774
6281 026140 020103
6282 026142 001403
6283 026144 104401 060530
6284 026150 000770
6285
6286 026152 104401 001165
6287 026156 000207

.SBTTL PRINT CONTROLLER STATUS

PRISTT: MOV R3,-(SP) ;STORE R3 ON STACK
MOV R1,-(SP) ;STORE R1 ON STACK
MOV R5,R3 ;STORE PARAMETER BLOCK ADDRESS
MOV R5,R1 ;FOR INDEX CALCULATIONS
ADD #P.CS1,R3 ;CALCULATE BEGINNING OF OUTPUT OCTAL
ADD #P.DS,R1 ;CALCULATE END OF OUTPUT OCTAL
TYPE ERR300 ;PRINT REGISTER NAMES
JSR PC,PRSTAT ;PRINT REGISTER CONTENTS
MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R3 ;RESTORE R3
RTS PC ;RETURN

.SBTTL PRINT REGISTER CONTENTS

PRSTAT: MOV (R3)+,-(SP) ;SAVE REGISTER VALUE FOR TYPE OUT
JSR PC,BINOC ;CONVERT TO OCTAL
CMP R1,R3 ;CHECK IF FINISHED
BEQ 10\$;YES RETURN
TYPE BLANKS ;TYPE 2 BLANKS
BR PRSTAT ;TYPE NEXT REGISTER

10\$: TYPE \$CRLF ;TYPE <CR><LF>
RTS PC ;RETURN

```
.SBTTL ECC CORRECTION ROUTINE
ECCCOR: MOV R4,-(SP) ;STORE R4 ON STACK
        MOV R3,-(SP) ;STORE R3 ON STACK
        MOV R1,-(SP) ;STORE R1 ON STACK
        MOV R0,-(SP) ;STORE R0 ON STACK
        MOV RTY6A,R3 ;LOAD BEGINNING OF SECTOR ADDRESS
        MOV RTYWC,R4 ;LOAD WORDS READ FROM SECTOR
        ASL R4 ;MAKE IT BYTES
        MOV R3,-(SP) ;STORE ADDRESS OF START OF SECTOR
        ADD R4,(SP) ;CALCULATE END OF SECTOR
        MOV P.EPOS(R5),R1 ;SAVE POSITION INFORMATION
        DEC R1 ;DECREMENT BIT COUNT BY ONE TO GET
        ; BIT ADDRESS
        MOV R1,R0
        ASL R1 ;DETERMINE WORDS THAT ARE BAD
        ASL R1 ; FOR DATA READ (SHIFT 3 BITS
        ASL R1 ; RIGHT)
        BIC #1,R1 ;CLEAR BYTE INDICATOR
        CMP R1,R4 ;CHECK IF ERROR WITHIN TRANSFER
        BHIS 10$ ;NO RETURN
        ADD R1,R3 ;DETERMINE ADDRESS OF ERROR
        MOV P.EPAT(R5),ECCPAT ;LOAD PATTERN
        CLR ECCPAT+2
        BIC #177760,R0 ;DETERMINE BIT POSITION FOR START OF ECC
        ; CORRECTION
        BEQ 5$ ;CORRECTION STARTS ON
        ; WORD BOUNDARY
        ASL ECCPAT ;SHIFT PATTERN 1 BIT LEFT
        ROL ECCPAT+2
        DEC R0 ;DECREMENT COUNT
        BNE 3$ ;CHECK IF IN POSITION
        MOV (R3),R0 ;CORRECT FIRST WORD WITH EXCLUSIVE
        MOV ECCPAT,R1 ; OF BITS IN ERROR
        BIC ECCPAT,(R3)
        BIC R0,R1
        BIS R1,(R3)+
        CMP (SP),R3 ;CHECK IF SECOND WORD IN TRANSFER
        BEQ 10$ ;NO RETURN
        MOV (R3),R0 ;CORRECT SECOND WORD WITH EXCLUSIVE
        MOV ECCPAT+2,R1 ; OF BITS IN ERROR
        BIC ECCPAT+2,(R3)
        BIC R0,R1
        BIS R1,(R3)
        TST (SP)+ ;ADJUST STACK
        MOV (SP)+,R0 ;RESTORE R0
        MOV (SP)+,R1 ;RESTORE R1
        MOV (SP)+,R3 ;RESTORE R3
        MOV (SP)+,R4 ;RESTORE R4
        RTS PC ;RETURN
```

```

        .SBTTL CONTROLLER ERROR DETECTED BY RK06 DRIVER
        CONTRL: MOV      R4, -(SP)          ; STORE R4 ON STACK
                JSR      PC, I.STOR        ; GET CONTROLLER STATUS
                BIT      #SW10, DSWR       ; CHECK IF BELL ON ERROR
                BEQ      1$                ; NO, CONTINUE
                TYPE     #BELL              ; RING BELL
                1$: MOV      E.CONT, R4     ; STORE CONTROLLER INDICATORS
                BIT      #E.MDS, R4        ; CHECK IF DUPLICATE DRIVE SELECT
                BEQ      2$                ; NO, CHECK FOR OTHER ERRORS
                TYPE     ERR006            ; TYPE "MULTIPLE DRIVE SELECT"
                BR       10$              ; DISPLAY REGISTERS

        2$: BIT      #E.CCLR, R4           ; CHECK IF "CLEAR CONTROLLER DID NOT
                ; CLEAR ERROR"
                BEQ      3$                ; NO, CHECK OTHER CONTROLLER ERRORS
                TYPE     ERR001            ; DISPLAY REGISTERS
                BR       10$

        3$: BIT      #E.NOAT, R4          ; CHECK IF "NO ATTENTION IN RKASOF"
                BEQ      4$                ; NO, CHECK THE OTHER CONTROLLER ERRORS
                TYPE     ERR002            ; DISPLAY REGISTERS
                BR       10$

        4$: BIT      #E.UDAT, R4          ; CHECK IF "UNEXPECTED DATA TYPE ERROR"
                BEQ      5$                ; NO, CHECK OTHER CONTROLLER ERRORS
                TYPE     ERR003            ; DISPLAY REGISTERS
                BR       10$

        5$: BIT      #E.CLAT, R4          ; CHECK IF "ATTENTION DID NOT RESET
                ; WITH DRIVE CLEAR"
                BEQ      6$                ; NO, CHECK OTHER CONTROLLER ERRORS
                TYPE     ERR004            ; DISPLAY REGISTERS
                BR       10$

        6$: BIT      #E.ILLD, R4          ; CHECK IF "ILLEGAL DRIVER COMMAND"
                BEQ      7$                ; NO, CHECK IF DATA LATE WHEN UNLOADING HEADER
                TYPE     ERR005            ; DISPLAY REGISTERS
                BR       10$

        7$: BIT      #E.DLT, R4           ; CHECK IF "DATA LATE WHEN UNLOADING HEADER"
                BEQ      8$                ; NO, CHECK IF "CONTROLLER ERROR WHILE GATERING
                ; STATUS"
                TYPE     ERR200            ; DISPLAY REGISTERS
                BR       10$

        8$: BIT      #E.CERR, R4          ; CHECK IF "CONTROLLER ERROR DURING
                ; DRIVE SERVICING"
                BEQ      9$                ; NO, CHECK IF CONTROLLER TIME OUT
                BIT      #DTYE, T.ER       ; SEE IF DRIVE TYPE ERROR
                BNE      13$              ; BR IF YES
                TYPE     ERR201            ; DISPLAY REGISTERS
                BR       10$

        13$: BITB   #BIT0, P.ASSN(R5)     ; SEE IF IN PROCESS OF ASSIGNING
                BNE      14$              ; BR IF YES
                TYPE     ,ERR201          ; ELSE VALID ERROR
    
```

```

6397 026560 000414 BR 10$ ;DISPLAY REGS
6398
6399 026562 004737 027030 14$: JSR PC,FLPAPM ;FILL PARAM BLOCK WITH CORRECT RK06-07 DATA
6400 026566 012737 000001 027156 MOV #1,NEWFLG ;SET FLAG TO SHOW DTYE ERROR ON ASSIGNING
6401 026574 012604 MOV (SP)+,R4 ;RESTORE R4
6402 026576 000427 BR 12$
6403
6404 026600 032704 040000 9$: BIT #E.CMTO,R4 ;CHECK IF CONTROLLER TIME OUT
6405 026604 001414 BEQ 20$ ;NO, CHECK IF DRIVE DETECTED DRIVE BUS PARITY ERROR
6406 026606 104401 064136 TYPE ,ERR207
6407
6408 026612 104401 001165 10$: TYPE $CRLF ;TYPE <CR><LF>
6409 026616 004737 043670 JSR PC,PRITIM ;PRINT TIME
6410 026622 004737 025652 JSR PC,PRIREG ;PRINT CONTROLLER REGISTERS
6411 026626 104401 060533 TYPE ,ERR000 ;TYPE FATAL ERROR
6412 026632 000000 HALT ;HANG ON FATAL ERROR
6413 026634 000776 BR -.2
6414
6415 026636 032704 002000 20$: BIT #E.DPAR,R4 ;CHECK IF "DRIVE DETECTED DRIVE BUS PARITY ERROR"
6416 026642 001001 BNE 30$ ;YES, SET DRPAR IN ERROR REGISTER
6417 026644 000000 HALT
6418
6419 026646 012604 000010 30$: MOV (SP)+,R4 ;RESTORE R4
6420 026650 052765 000010 000034 BIS #DRPAR.P.ER(R5) ;SET DRIVE DETECTED DRIVE BUS$OTT> PARITY
6421 026656 010346 12$: MOV R3,-(SP) ;STORE R3 ON THE STACK
6422 026660 005037 001464 CLR 0,WAIT ;CLEAR WAITING FOR COMMAND COMPLETION
6423 026664 116503 000000 MOV#B P.DRVN(R5),R3 ;GET DRIVE NUMBER
6424 026670 005062 000026 CLR RKMR1(R2) ;CLEAR MAINTENANCE REGISTER 1
6425 026674 010362 000010 MOV R3,RKCS2(R2) ;LOAD DRIVE NUMBER
6426 026700 052762 000040 000010 BIS #SCLR,RKCS2(R2) ;DO SUBSYS CLR
6427 026706 105762 000000 35$: TSTB RKCS1(R2) ;WAIT FOR READY
6428 026712 001775 BEQ 35$
6429 026714 004737 047236 JSR PC,I.STOR ;STORE REGISTERS
6430 026720 032737 100000 001410 BIT #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR
6431 026726 001413 BEQ 37$ ;NO, CHECK IF ATTENTION CLEARED
6432
6433 ; CHECK FOR CONTROLLER TYPE ERROR
6434 026730 032737 137400 001412 BIT #UFE!MDS!PGE!NEM!NED!UPE!DLT,T.CS2
6435 026736 001004 BNE 36$
6436 026740 032737 130761 001426 BIT #ILC!DTYE!FMTE!ECH!BSE!HVRC!DTE!OPI!DCK,T.ER
6437 026746 001412 BEQ 40$
6438
6439 026750 104401 063651 36$: TYPE ERR201 ;TYPE CONTROLLER ERROR DURING DRIVE SERVICING
6440 026754 000716 BR 10$ ;GO REPORT ERROR
6441
6442 026756 136337 001514 001425 37$: BITB INTMSK(R3),T.ASOF+1 ;CHECK IF ATTENTION CLEARED
6443 026764 001403 BEQ 40$ ;YES, REPORT ERROR
6444 026766 104401 060707 TYPE ERR004 ;TYPE "DRIVE CLEAR DID NOT RESET ATTENTION"
6445 026772 000707 BR 10$ ;GO REPORT ERROR
6446
6447 026774 146337 001514 001512 40$: BICB INTMSK(R3),W.TIME ;CLEAR TIMING CURRENTLY ON THIS DRIVE
6448 027002 006303 ASL R3 ;MULTIPLY BY 2
6449 027004 005063 001544 CLR W.DRV(R3) ;CLEAR WATCH DOG TIME
6450 027010 012603 MOV (SP)+,R3 ;RESTORE R3
6451 027012 005737 027156 TST NEWFLG ;SEE IF DTYE ERROR ON ASSIGNING
6452 027016 001402 BEQ 41$ ;BR IF NO

```

```

6453 027020 000137 024222          JMP      NORMAL          ;ELSE GO NORMAL RET
6454 027024 000137 025312          41$:   JMP      ABNORM    ;GO TO ABNORMAL RETURN
6455
6456 027030 132765 000004 000007  FLPARM: BITB   #B.CDT,P.CS1H(R5)  ;SEE IF RK06 PARAMETER USED LAST
6457 027036 001407          BEQ      1$           ;BR IF YES
6458 027040 142765 000004 000007  BICB   #B.CDT,P.CS1H(R5)  ;ELSE TRY RK06 PARAM BECAUSE OF DTYE
6459 027046 012765 000632 000070  MOV    #3E,P.MXCL(R5)
6460 027054 000207          RTS      PC
6461
6462 027056 152765 000004 000007  1$:   BISB   #B.CDT,P.CS1H(R5)  ;TRY RK07 PARAM BECAUSE OF DTYE
6463 027064 012765 001456 000070  MOV    #1456,P.MXCL(R5)
6464 027072 000207          RTS     PC
6465
6466 027074 132765 000004 000007  GETNUM: BITB   #B.CDT,P.CS1H(R5)  ;SEE IF RK06
6467 027102 001412          BFC     1$           ;BR IF YES
6468
6469 027104 012737 001456 027154  MOV    #814.,LSTCYL    ;ELSE LOAD RK07 VALUES
6470 027112 012737 151010 027160  MOV    #<814.*22.*3.>+<22.*2.>,HOLD
6471 027120 012737 151011 027162  MOV    #<814.*22.*3.>+<22.*2.>+1,HOLD1
6472 027126 000207          RTS     PC
6473
6474 027130 012737 000632 027154  1$:   MOV    #410.,LSTCYL    ;LOAD RK06 VALUES
6475 027136 012737 064740 027160  MOV    #<410.*22.*3.>+<22.*2.>,HOLD
6476 027144 012737 064741 027162  MOV    #<410.*22.*3.>+<22.*2.>+1,HOLD1
6477 027152 000207          RTS     PC
6478
6479 027154 000000          LSTCYL: 0
6480 027156 000000          NEWFLG: 0
6481 027160 000000          HOLD:   0
6482 027162 000000          HOLD1:  0
6483

```



```

        .SBTTL CONTROLLER ERROR DUE TO REIGSTER INCONSISTANCIES
6484      CNTRL1: JSR      PC,Q.RMCV      ;REMOVE PARAMETER BLOCK FROM THE
6485      ;COMMAND INITIATION QUEUE
6486      027164 004737 051626      ;CHECK IF BELL ON ERROR
6487      ;NO, CONTINUE
6488      027170 032777 002000 151742 . BIT      #SW10,@SWR      ;RING BELL
6489      027176 0C1402      ;CHECK IF INHIBIT PRINT OUT
6490      027200 104401 001160      ;NO PRINT ERROR MESSAGE
6491      027204 032777 020000 151726 1$: BIT      #SW13,@SWR      ;CHECK IF CYCLE ON OPERATION
6492      027212 001404      ;YES, CHECK IF LOOP ON OPERATION
6493      027214 032777 040000 151716 . BIT      #SW14,@SWR      ;CHECK IF "CONTROLLER ERROR NOT FLAGGED"
6494      027222 001050      ;CHECK IF "WORD COUNT NOT 0"
6495      027224 032737 000001 001564 2$: BIT      #BIT0,ERCONT
6496      027232 001402      ;CHECK IF "BUS ADDRESS INCORRECT"
6497      027234 104401 061311 001564 3$: BEQ      #BIT1,ERCONT
6498      027240 032737 000002 001564 4$: BIT      #BIT2,ERCONT
6499      027246 001402      ;CHECK IF "CYLINDER, TRACK, SECTOR
6500      027250 104401 061350      ;INCORRECT"
6501      027254 032737 000004 001564 4$: BEQ      #BIT3,ERCONT
6502      027262 001402      ;CHECK IF INHIBIT PRINT OUT
6503      027264 104401 061401      ;YES, CHECK IF HALT ON ERROR
6504      027270 032737 000010 001564 5$: BIT      #BIT0,ERCONT
6505      ;PRINT TIME
6506      027276 001402      ;PRINT RK06 REGISTERS
6507      027300 104401 061425      ;PRINT CURRENT SUPPLIED COMMAND
6508      027304 032777 020000 151626 7$: BEQ      #BIT0,ERCONT
6509      027312 001014      ;CHECK IF CONTROLLER ERROR
6510      027314 004737 043670      ;NOT FLAGGED
6511      027320 004737 026074      ;NO, DO NOT PRINT PREVIOUS COMMAND
6512      027324 004737 030244      ;PRINT PREVIOUS COMMAND
6513      027330 032737 000001 001564 . BIT      #BIT0,ERCONT
6514      ;CHECK IF HALT ON ERROR
6515      027336 001402      ;YES, HALT
6516      027340 004737 030066      ;CHECK IF LOOP OPERATION
6517      027344 032777 100000 151566 10$: BIT      #SW15,@SWR
6518      027352 001010      ;YES, CLEAR ERROR
6519      027354 032777 040000 151556 . BIT      #SW14,@SWR
6520      027362 001005      ;PRINT "FATAL ERROR"
6521      027364 104401 060533      ;PRINT "SET SWITCH 14 TO CYCLE ON ERROR"
6522      027370 104401 061030      ;SET CONTROLLER ERROR PRESENT
6523      027374 000000      ;CLEAR CONTROLLER
6524      027376 012737 100000 001564 11$: HALT
6525      027404 012762 100000 000000 15$: MOV      #BIT15,ERCONT
6526      027412 005062 000026      ;CLEAR MAINTANENCE REGISTER 1
6527      027416 112737 000005 001410 . CLR      #CCLR,RKCS1(R2)
6528      027424 004037 046774      ;LOAD COMMAND INTO TEMPORARY CS1
6529      027430 027440      ;GO ISSUE DRIVE CLEAR
6530      027432 004037 051406      ;ERROR RETURN
6531      027436 001324      ;PUT PARAMETER BLOCK ON COMMAND
6532      027440 000207      ;INITIATION QUEUE
        ;RETURN
    
```

```

6533          .SBTTL PRINT WHOLE DATA BUFFER
6534
6535 027442 122765 000121 000123 PRIBUF: CMPB  #RDATA,P.RCMD(R5) ;CHECK IF READ DATA
6536 027450 001017          BNE  SS          ;NO, RETURN
6537 027452 032765 000010 000150 BIT  #BIT3,F.DSTT(PS) ;CHECK IF TIME TO PRINT BUFFER
6538 027460 001013          BNE  SS          ;NO, RETURN
6539 027462 052765 000010 000150 BIS  #BIT3,P.DSTT(R5) ;SET BUFFER ATTEMPTED TO BE PRINTED
6540 027470 032777 020000 151442 BIT  #SW13,2SWR      ;CHECK IF INHIBIT PRINT OUT
6541 027476 001004          BNE  SS          ;YES, RETURN
6542 027500 032777 000010 151432 BIT  #SW3,2SWR      ;CHECK IF PRINT SECTOR BEFORE BEGINNING
6543          BNE  SS          ;ERROR RECOVERY
6544 027506 001001          BNE  10$         ;YES, PRINT SECTOR IN ERROR
6545 027510 000207          RTS  PC          ;RETURN
6546
6547 027512 010146          10$: MOV  R1,-(SP)      ;STORE R1 ON THE STACK
6548 027514 010346          MOV  R3,-(SP)      ;STORE R3 ON THE STACK
6549 027516 010446          MOV  R4,-(SP)      ;STORE R4 ON THE STACK
6550 027520 104401 064321 TYPE  ,EARR213     ;TYPE ADDRESS OF BEGINNING OF
6551          BNE  SS          ;SECTOR IN ERROR
6552 027524 013703          MOV  RTYBA,R3      ;GET ADDRESS OF START OF SCTOR
6553 027530 013704          MOV  RTYWC,R4      ;GET WORD COUNT
6554 027534 010346          MOV  R3,-(SP)      ;STORE STARTING ADDRESS FOR PRINT OUT
6555 027536 004737 052774 JSR  PC,BINOCT     ;CONVERT OF OCTAL
6556 027542 104401 064372 TYPE  ,EARR214     ;TYPE DATA READ
6557 027546 012701 000010 15$: MOV  #8,R1        ;SET 8 COLUMNS ON PAGE
6558 027552 104401 001165 TYPE  $CRLF        ;TYPE <CR><LF>
6559 027556 012346          20$: MOV  (R3)+,-(SP) ;GET NEXT WORD OF SECTOR
6560 027560 004737 052774 JSR  PC,BINOCT     ;CONVERT TO OCTAL
6561 027564 005304          DEC  R4          ;CHECK IF FINISHED
6562 027566 001405          BEQ  30$        ;YES, RESTORE REGISTERS
6563 027570 005301          DEC  R1          ;CHECK IF END OF LINE
6564 027572 001765          BEQ  15$        ;YES, INITIALIZE COLUMN COUNT
6565 027574 104401 060530 TYPE  ,BLANKS     ;TYPE 2 BLANKS
6566 027600 000766          BR   20$        ;GET NEXT WORD
6567
6568 027602 104401 001165          30$: TYPE  $CRLF        ;TYPE <CR><LF>
6569 027606 012604          MOV  (SP)+,R4     ;RESTORE R4
6570 027610 012603          MOV  (SP)+,R3     ;RESTORE R3
6571 027612 012601          MOV  (SP)+,R1     ;RESTORE R1
6572 027614 000207          RTS  PC          ;RETURN

```

.SBTTL PRINT DRIVE AND PACK SERIAL NUMBERS

6573							
6574							
6575	027616	010146					
6576	027620	010346					
6577	027622	010446					
6578	027624	012701	056600				
6579	027630	016503	000224				
6580	027634	006103					
6581	027636	006103					
6582	027640	006103					
6583	027642	006103					
6584	027644	012704	000003				
6585	027650	105011		2\$:			
6586	027652	006103					
6587	027654	106111					
6588	027656	006103					
6589	027660	106111					
6590	027662	006103					
6591	027664	106111					
6592	027666	006103					
6593	027670	106111					
6594	027672	152721	000060				
6595	027675	005304					
6596	027700	001363					
6597	027702	104401	056555				
6598	027706	016503	000226				
6599							
6600	027712	016504	000230				
6601							
6602	027716	012701	057143				
6603	027722	105011					
6604	027724	000404					
6605							
6606	027726	105011		5\$:			
6607	027730	006103					
6608	027732	006104					
6609	027734	106111					
6610	027736	006103		7\$:			
6611	027740	006104					
6612	027742	106111					
6613	027744	006103					
6614	027746	006104					
6615	027750	106111					
6616	027752	152721	000060				
6617	027756	022701	057156				
6618	027762	101361					
6619	027764	104401	057116				
6620	027770	012604					
6621	027772	012603					
6622	027774	012601					
6623	027776	000207					

```

PRISER: MOV R1,-(SP) ;STORE R1 ON STACK
        MOV R3,-(SP) ;STORE R3 ON STACK
        MOV R4,-(SP) ;STORE R4 ON STACK
        MOV #STT100,R1 ;GET ADDRESS OF ASCII STRING
        MOV P.SERL(R5),R3 ;GET DRIVE SERIAL NUMBER
        ROL R3 ;SHIFT DRIVE SERIAL NUMBER 4 BITS LEFT
        ROL R3
        ROL R3
        ROL R3
        MOV #3,R4 ;LOAD NUMBER OF DIGITS OF SERIAL NUMBER
2$: CLR B (R1) ;CLEAR NEXT BYTE FOR PRINT OUT
    ROL R3 ;SHIFT IN NEXT 4 BITS
    ROL B (R1)
    ROL R3
    ROL B (R1)
    ROL R3
    ROL B (R1)
    BIS B #60,(R1)+ ;CHANGE BCD TO DECIMAL
    DEC R4 ;CHECK IF FINISHED
    2$: CONVERT NEXT DIGIT
    TYPE "DRIVE SERIAL NUMBER" ;TYPE "DRIVE SERIAL NUMBER"
    MOV P.PKSR(R5),R3 ;GET LEAST SIGNIFICANT BITS OF
        PACK SERIAL NUMBER
    MOV P.PKSR+2(R5),R4 ;GET MOST SIGNIFICANT BITS OF
        PACK SERIAL NUMBER
    MOV #STT012,R1 ;GET ADDRESS OF ASCII STRING
    CLR B (R1) ;CLEAR FIRST BYTE OF PRINT OUT
    BR 7$ ;START CONVERSION TO OCTAL
5$: CLR B (R1) ;CLEAR NEXT BYTE OF PRINT OUT
    ROL R3 ;SHIFT IN NEXT THREE BITS
    ROL R4
    ROL B (R1)
7$: ROL R3
    ROL R4
    ROL B (R1)
    ROL R3
    ROL R4
    ROL B (R1)
    BIS B #60,(R1)+ ;MAKE IT ASCII
    CMP #STT013,R1 ;CHECK IF FINISHED
    BHI 5$ ;NO CONTINUE CONVERSION
    TYPE PACK SERIAL NUMBER ;TYPE PACK SERIAL NUMBER
    MOV (SP)+,R4 ;RESTORE R4
    MOV (SP)+,R3 ;RESTORE R3
    MOV (SP)+,R1 ;RESTORE R1
    RTS PC ;RETURN
    
```

```

        .SBTTL PRINT BEGINNING OF ERROR HEADER
6624
6625
6626 030000 032777 002000 151132 PRI000: BIT      @SW10,@SWR      ;CHECK IF BELL ON ERROR
6627 030006 001402                BEQ      1$              ;NO, CHECK IF INHIBIT PRINT OUT
6628 030010 104401 001160                TYPE     $BELL          ;RING BELL
6629 030014 032777 020000 151116 1$: BIT      @SW13,@SWR      ;CHECK IF INHIBIT PRINT OUT
6630 030022 001020                BNE     20$            ;YES, INCREMENT CONTROLLER ERROR
6631 030024 116537 000000 056357        MOVB    P.DRVN(R5),OPR011 ;STORE DRIVE NUMBER
6632 030032 152737 000060 056357        BISB    @60,OPR011      ;MAKE IT ASCII
6633 030040 104401 056340                TYPE     OPR010        ;TYPE "DRIVE NUMBER N"
6634 030044 132765 000001 000210        BITB    @BIT0,P.ASSN(R5) ;SEE IF SIZING FOR DRIVE
6635 030052 001004                BNE     20$            ;BR IF YES, TO RETURN
6636 030054 004737 027616        JSR    PC,PRISER       ;PRINT DRIVE AND PACK SERIAL NUMBERS
6637 030060 004737 043670        JSR    PC,PRITIM       ;PRINT TIME
6638 030064 000207                RTS     PC              ;RETURN

```

```

6639 .SBTTL PRINT PREVIOUS COMMAND
6640
6641 030066 104401 063462 000135 PRI001: TYPE ERR101 ;TYPE "PREVIOUS CMND CYL TRK SEC WRD CT"
6642 030072 122765 000121 000135 CMPB BRDDATA,P.LCMD(R5) ;CHECK IF READ DATA
6643 030100 001402 BEQ 5$ ;YES, DO NOT PRINT PATTERN NUM
6644 030102 104401 063544 TYPE ,ERR102 ;PRINT "PAT"
6645 030106 104401 001165 5$: TYPE ,SCRLF ;PRINT <CR><LF>
6646 030112 005046 CLR -(SP) ;MAKE ROOM ON STACK
6647 030114 116516 000135 MOVB P.LCMD(R5),(SP) ;GET LAST COMMAND
6648 030120 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6649 030124 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6650 030130 016546 000136 MOV P.LCYL(R5),-(SP) ;GET LAST CYLINDER USED
6651 030134 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6652 030140 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6653 030144 005046 CLR -(SP) ;MAKE ROOM ON STACK
6654 030146 116516 000141 MOVB P.LTRK(R5),(SP) ;GET LAST TRACK USED
6655 030152 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6656 030156 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6657 030162 005046 CLR -(SP) ;MAKE ROOM ON STACK
6658 030164 116516 000140 MOVB P.LSEC(R5),(SP) ;GET LAST SECTOR USED
6659 030170 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6660 030174 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6661 030200 016546 000142 MOV P.LWC(R5),-(SP) ;STORE LAST WORD COUNT
6662 030204 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6663 030210 122765 000121 000135 CMPB BRDDATA,P.LCMD(R5) ;CHECK IF PAT # VALID
6664 030216 001407 BEQ 10$ ;NO, DO NOT TYPE
6665 030220 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6666 030224 005046 CLR -(SP) ;MAKE ROOM ON STACK
6667 030226 116516 000144 MOVB P.LDPT(R5),(SP) ;GET LAST DATA PATTERN
6668 030232 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6669 030236 104401 001165 10$: TYPE ,SCRLF ;TYPE <CR><LF>
6670 030242 000207 RTS PC ;RETURN

```

```

6671 .SBTTL PRINT CURRENT GENERATED COMMAND
6672
6673 030244 104401 063351 PRI002: TYPE ,ERR100 ;PRINT "CURRENT SUPPLIED COMMAND"
6674 ; CYL TRK SEC OFFSET
6675 ; BUS AD WRD CT"
6676 030250 122765 000121 000123 CMPB #PDDATA,P.RCMD(R5) ;CHECK IF PAT # IS TO BE PRINTED
6677 030256 001402 BEQ 5$ ;NO, DO NOT PRINT HEADER
6678 030260 104401 063544 TYPE ,ERR102 ;PRINT "PAT"
6679 030264 104401 001165 5$: TYPE ,$CRLF ;PRINT <CR><LF>
6680 030270 005046 CLR -(SP) ;MAKE ROOM ON STACK
6681 030272 116516 000001 MOVB P.CMND(R5),(SP) ;STORE COMMAND FOR PRINT OUT
6682 030276 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6683 030302 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6684 030306 016546 000002 MOV P.CYLN(R5),-(SP) ;STORE CYLINDER
6685 030312 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6686 030316 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6687 030322 005046 CLR -(SP) ;MAKE ROOM ON STACK
6688 030324 116516 000005 MOVB P.TRCK(R5),(SP) ;STORE TRACK
6689 030330 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6690 030334 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6691 030340 005046 CLR -(SP) ;MAKE ROOM ON STACK
6692 030342 116516 000004 MOVB P.SECT(R5),(SP) ;STORE SECTOR
6693 030346 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6694 030352 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6695 030356 005046 CLR -(SP) ;MAKE ROOM ON THE STACK
6696 030360 122765 000021 000147 CMPB #21,P.RERD(R5) ;CHECK IF OFFSET
6697 030366 101002 BHI 10$ ;NO, INDICATE ZERO OFFSET
6698 030370 116516 000006 MOVB P.OFST(R5),(SP) ;STORE OFFSET FOR PRINT OUT
6699 030374 004737 052774 10$: JSR PC,BINOC ;CONVERT TO OCTAL
6700 030400 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6701 030404 016546 000010 MOV P.BALO(R5),-(SP) ;STORE BUS ADDRESS
6702 030410 004737 052774 JSR PC,BINOC ;COVERT TO OCTAL
6703 030414 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6704 030420 016546 000012 MOV P.WC(R5),-(SP) ;STORE WORD COUNT
6705 030424 004737 052774 JSR PC,BINOC ;COVERT TO OCTAL
6706 030430 122765 000121 000123 CMPB #R0DATA,P.RCMD(R5) ;CHECK IF PAT # IS TO BE PRINTED
6707 030436 001402 BEQ 15$ ;NO, RETURN
6708 030440 104401 060530 TYPE ,BLANKS ;TYPE 2 BLANKS
6709 030444 005046 CLR -(SP) ;MAKE ROOM ON THE STACK
6710 030446 116516 000121 MOVB P.DPAT(R5),(SP) ;STORE DATA PATTERN FOR PRINT OUT
6711 030452 004737 052774 JSR PC,BINOC ;CONVERT TO OCTAL
6712 030456 104401 001165 15$: TYPE ,$CRLF ;TYPE <CR><LF>
6713 030462 000207 RTS ;RETURN

```

6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728 030464 010446
6729 030466 012004
6730 030470 005204
6731 030472 010437 030504
6732 030476 004737 030000
6733 030502 104401
6734 030504 000000
6735 030506 132765 000001 000210
6736 030514 001045
6737 030516 104401 001165
6738 030522 132777 020000 150410
6739 030530 001033
6740 030532 105765 000150
6741 030536 100402
6742 030540 104401 063721
6743 030544 004737 026074
6744 030550 132744 000002
6745 030554 001407
6746 030556 132714 000040
6747 030562 001402
6748 030564 104401 064102
6749 030570 004737 025720
6750 030574 132714 000100
6751 030600 001402
6752 030602 004737 030244
6753 030606 132714 000004
6754 030612 001402
6755 030614 004737 030066
6756 030620 004737 031250
6757 030624 005265 000206
6758 030630 004737 022642
6759 030634 004737 013004
6760 030640 004737 022342
6761 030644 005037 001566
6762 030650 012604
6763 030652 000200

.SBTTL PRINT ERROR MESSAGE AND DROP DRIVE

```

*****
*
*   EXPECTED PRINT CODES
*   -----
*
*   BIT 1  PRINT DRIVE STATUS
*   BIT 2  PRINT PREVIOUS POSITION
*   BIT 5  QUESTIONABLE DRIVE STATUS
*   BIT 6  CURRENT GENERATED COMMAND
*
*****

```

```

PRI100: MOV     R4, -(SP)      ; STORE R4 ON STACK
        MOV     (R0)+, R4   ; STORE ADDRESS OF PRINT CODE
        INC     R4          ; CALCULATE START OF MESSAGE
        MOV     R4, 2$     ; STORE ERROR MESSAGE ADDRESS
        JSR     PC, PRI000  ; PRINT HEADER
        TYPE    WORD       ; TYPE ERROR MESSAGE
        .WORD   0
2$:     BITB    #BIT0, P.ASSN(R5) ; SEE IF SIZING FOR DRIVE
        BNE    12$         ; BR IF YES
        TYPE    $CRLF      ; TYPE <CR><LF>
        BIT    #SW13, $SWR ; CHECK IF INHIBIT PRINTOUT
        BNE    10$         ; YES, RETURN
        TSTB   P.DSTT(R5) ; CHECK IF FIRST ERROR
        BMI    3$          ; YES, GO PRINT CONTROLLER REGISTERS
        TYPE    ERR202     ; TYPE ERROR DURING ERROR RECOVERY
        JSR     PC, PRISTT  ; PRINT CONTROLLER REGISTERS
        BITB   #BIT1, -(R4) ; CHECK IF PRINT DRIVE STATUS
        BEQ    5$          ; NO, CONTINUE
        BITB   #BIT5, (R4) ; CHECK IF QUESTIONABLE STATUS
        BEQ    4$          ; NO, DO NOT PRINT MESSAGE
        TYPE    ERR206     ; PRINT "QUESTIONABLE DRIVE STATUS"
        JSR     PC, PRDSTT ; PRINT DRIVE STATUS
        BITB   #BIT6, (R4) ; CHECK IF PRINT CURRENT GENERATED COMMAND
        BEQ    8$          ; NO, DO NOT PRINT CURRENT COMMAND
        JSR     PC, PRI002  ; PRINT CURRENT GENERATED COMMAND
        BITB   #BIT2, (R4) ; CHECK IF PRINT PREVIOUS COMMAND
        BEQ    10$         ; NO, CHECK IF HALT ON ERROR
        JSR     PC, PRI001  ; PRINT PREVIOUS COMMAND
        JSR     PC, HLT000  ; TEST IF HALT ON ERROR
        INC     P.NER(R5)  ; INCREMENT OTHER ERROR COUNT
        JSR     PC, BUFREL ; RELEASE BUFFER
        JSR     PC, DROP    ; DROP DRIVE FROM TEST SEQUENCE
        JSR     PC, GETBUF  ; GET NEW DRIVE, COMMAND, AND BUFFER
        CLR     ERAPRO     ; CLEAR ERROR CURRENTLY BEING PROCESSED
        MOV     (SP)+, R4  ; RESTORE R4
        RTS     R0         ; RETURN

```

6764
6765
6766
6767
6768
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6799
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6810
6811
6812
6813
6814
6815
6816
6817
6818
6819

030654 010446
030656 012004
030660 132724 000001
030664 001416
030666 032765 000002 000150
030674 001402
030676 012604
030700 000200
030702 052765 000002 000150 15:
030710 100015
030712 052765 000001 000150
030720 000411
030722 032765 000004 000150 55:
030730 001402
030732 012604
030734 000200
030736 052765 000004 000150 65:
030744 004737 030000 105:
030750 032777 020000 150162
030756 001115
030760 010437 031000
030764 105765 030150
030770 100402
030772 104401 063721
030776 104401 115:
031000 000000 125:
031002 104401 001165
031006 004737 026074
031012 132744 000002
031016 001407
031020 132714 000040
031024 001402
031026 104401 064102
031032 004737 025720 205:

.SBTTL PRINT ERROR MESSAGE

```
*****
*
*   EXPECTED PRINT CODES
*   -----
*
*   BIT 0   DATA TYPE ERROR = 1
*           NON-DATA TYPE ERROR = 0
*
*   BIT 1   PRINT DRIVE STATUS
*
*   BIT 2   PRINT PREVIOUS POSITION
*
*   BIT 3   PRINT SECTOR IN ERROR
*
*   BIT 4   PRINT HEADER
*
*   BIT 5   QUESTIONABLE DRIVE STATUS
*
*   BIT 6   CURRENT GENERATED COMMAND
*
*****
```

```
PR1200: MOV R4, -(SP) ; STORE R4 ON STACK
        MOV (R0)+, R4 ; GET PRINT CODE ERROR FLAGS
        BITB #BIT0, (R4)+ ; CHECK IF DATA TYPE ERROR
        BEQ 55 ; NO, NON-DATA TYPE ERROR
        BIT #BIT1, P.DSTT(R5) ; CHECK IF DATA TYPE ERROR
        ; BEING PROCESSED
        BEQ 15 ; NO, PRINT MESSAGE
        MOV (SP)+, R4 ; RESTORE R4
        RTS R0 ; RETURN

15:     BIS #BIT1, P.DSTT(R5) ; SET DATA TRANSFER ERROR OCCURRED
        BPL 105 ; CHECK IF START OF ERROR RECOVERY
        BIS #BIT0, P.DSTT(R5) ; YES, SET SERVICING DATA ERROR
        BR 105 ; GO REPORT ERROR

55:     BIT #BIT2, P.DSTT(R5) ; CHECK IF NON-DATA TYPE ERROR OCCURRED
        BEQ 65 ; NO, REPORT ERROR
        MOV (SP)+, R4 ; RESTORE R4
        RTS R0 ; RETURN

65:     BIS #BIT2, P.DSTT(R5) ; SET NON-DATA ERROR OCCURRED
        JSR PC, PR1J00 ; TYPE ERROR HEADER
        BIT #SW13, 2SWR ; CHECK IF INHIBIT PRINT OUT
        BNE 305 ; YES, INHIBIT PRINT OUT
        MOV R4, 125 ; STORE ERROR MESSAGE
        TSTB P.DSTT(R5) ; CHECK IF ERROR DURING RECOVERY
        BMI 115 ; NO, PRINT ERROR
        TYPE ,ERR20 ; PRINT "ERROR DURING ERROR RECOVERY"
        TYPE ; PRINT ERROR MESSAGE

115:    TYPE
        125:    .WORD 0
        TYPE $CRLF ; TYPE <CR><LF>
        JSR PC, PR1STT ; PRINT CONTROLLER REGISTERS
        BITB #BIT1, -(R4) ; CHECK IF PRINT DRIVE STATUS
        BEQ 225 ; NO, CONTINUE
        BITB #BIT5, (R4) ; CHECK IF QUESTIONABLE DRIVE STATUS
        BEQ 205 ; NO, DO NOT PRINT MESSAGE
        TYPE ERR206 ; TYPE "QUESTIONABLE DRIVE STATUS"
        JSR PC, PRDSTT ; PRINT DRIVE STATUS
```


6820	031036	132714	000100		22\$:	BITB	#BIT6,(R4)	;CHECK IF PRINT CURRENT COMMAND
6821	031042	001402				BEQ	23\$;NO CONTINUE
6822	031044	004737	030244			JSR	PC,PRIO02	;PRINT CURRENTLY ISSUED COMMAND
6823	031050	132714	000004		23\$:	BITB	#BIT2,(R4)	;CHECK IF PRINT PREVIOUS COMMAND
6824	031054	001402				BEQ	25\$;NO CHECK IF PRINT HEADER
6825	031056	004737	030066			JSR	PC,PRIO01	;PRINT PREVIOUS COMMAND
6826	031062	132714	000001		25\$:	BITB	#BIT0,(R4)	;CHECK IF DATA TYPE ERROR
6827	031066	001422				BEQ	27\$;NO DO NOT PRINT ECC
6828	031070	032765	100000	000034		BIT	#DCK,P.ER(R5)	;CHECK IF DATA CHECK
6829	031076	001416				BEQ	27\$;NO DO NOT PRINT ECC
6830	031100	104401	063150			TYPE	ERR067	;TYPE ECC PAT, ECC POS
6831	031104	016546	000062			MOV	P.EPAT(R5),-(SP)	;GET PATTERN
6832	031110	004737	052774			JSR	PC,BIN0CT	;CONVERT TO OCTAL
6833	031114	104401	060530			TYPE	BLANKS	
6834	031120	016546	000060			MOV	P.EPOS(R5),-(SP)	;GET POSITION
6835	031124	004737	052774			JSR	PC,BIN0CT	;CONVERT TO OCTAL
6836	031130	104401	001165			TYPE	\$CRLF	
6837	031134	132724	000020		27\$:	BITB	#BIT4,(R4)+	;CHECK IF PRINT HEADER
6838	031140	001424				BEQ	30\$;NO CHECK IF HALT ON ERROR
6839	031142	104401	063563			TYPE	ERR111	;TYPE HEADER DESCRIPTOR
6840	031146	013746	001646			MOV	HEAD1,-(SP)	;STORE FIRST WORD OF THE HEADER
6841	031152	004737	052774			JSR	PC,BIN0CT	;CONVERT TO OCTAL
6842	031156	104401	060530			TYPE	BLANKS	;TYPE 2 BLANKS
6843	031162	013746	001650			MOV	HEAD2,-(SP)	;STORE SECOND WORD OF THE HEADER
6844	031166	004737	052774			JSR	PC,BIN0CT	;CONVERT TO OCTAL
6845	031172	104401	060530			TYPE	BLANKS	;TYPE 2 BLANKS
6846	031176	013746	001652			MOV	HEAD3,-(SP)	;STORE THIRD WORD OF THE HEADER
6847	031202	004737	052774			JSR	PC,BIN0CT	;CONVERT TO OCTAL
6848	031206	104401	001165			TYPE	\$CRLF	;TYPE <CR><LF>
6849	031212	132744	000010		30\$:	BITB	#BIT3,-(R4)	;CHECK IF PRINT WHOLE SECTOR
6850	031216	001402				BEQ	40\$;NO CHECK IF HALT ON ERROR
6851	031220	004737	027442			JSR	PC,PRIBUF	;GO PRINT BUFFER
6852	031224	132714	000001		40\$:	BITB	#BIT0,(R4)	;CHECK IF DATA TYPE ERROR
6853	031230	001403				BEQ	45\$;NO CHECK IF HALT ON ERROR
6854	031232	042765	100200	000150		BIC	#BIT15!BIT7,P.DST(R5)	;CLEAR FIRST ERROR AND ERROR
6855								;ENQUEUED
6856	031240	004737	031250		45\$:	JSR	PC,HLT000	;CHECK IF HALT ON ERROR
6857	031244	012604				MOV	(SP)+,R4	;RESTORE R4
6858	031246	000200				RTS	RO	;RETURN

```

6859          .SBTTL HALT ON ERROR CHECK
6860
6861 031250 032777 100000 147662 HLT000: BIT      #SW15,2SWR    ;CHECK IF HALT ON ERROR
6862 031256 001407          BEQ      10$          ;NO RETURN
6863 031260 005737 001464          TST      0,WAIT      ;CHECK IF WAITING FOR COMMAND
6864          BEQ      5$          ;COMPLETION
6865 031264 001403          1$:      TSTB     RKCS1(R2)    ;NO HALT
6866 031266 105762 000000          5$:      HALT
6867 031272 001775          10$:     RTS      PC      ;WAIT FOR READY
6868 031274 000000
6869 031276 000207

```

```

6870 .SBTTL OTHER ERROR STATISTICS GATHERING
6871
6872 031300 105765 000150  NERSTT: TSTB P.DSTT(R5) ;CHECK IF FIRST ERROR
6873 031304 100057          BPL STT5$ ;NO, CONTINUE ERROR RECOVERY
6874 031306 042765 100200 000150 BIC #BIT15!BIT7,P.DSTT(R5) ;RESET FIRST ERROR AND ERROR ENQUEUED
6875 031314 005265 000206          INC P.NER(R5) ;INCREMENT OTHER ERRORS COUNT
6876 031320 022765 000020 000206 CMP #T,ER,P.NER(R5) ;CHECK IF ERROR THRESHOLD EXCEEDED
6877 031326 103426          BLO STT2$ ;YES, CHECK IF DROP DRIVE
6878 031330 000445          BR STT1$ ;NO, START ERROR RECOVERY SEQUENCE
6879
6880 .SBTTL SEEK INCOMPLETE STATISTICS GATHERING
6881
6882 031332 105765 000150  SKISTT: TSTB P.DSTT(R5) ;CHECK IF FIRST ERROR
6883 031336 100042          BPL STT5$ ;NO, CONTINUE ERROR RECOVERY
6884 031340 005265 000202          INC P.NSKI(R5) ;INCREMENT NUMBER OF SEEK INCOMPLETES
6885 031344 000405          BR STT1$ ;GO CHECK IF ERROR THRESHOLD EXCEEDED
6886
6887 .SBTTL OPERATION INCOMPLETE STATISTICS GATHERING
6888
6889 031346 105765 000150  OPISTT: TSTB P.DSTT(R5) ;CHECK IF FIRST ERROR
6890 031352 100034          BPL STT5$ ;NO, CONTINUE ERROR PROCESSING
6891 031354 005265 000204          INC P.NOPI(R5) ;INCREMENT NUMBER OF OPERATION INCOMPLETES
6892 031360 042765 100200 000150 STT1$: BIC #BIT15!BIT7,P.DSTT(R5) ;RESET FIRST ERROR AND ERROR ENQUEUED
6893 031366 016546 000204          MOV P.NOPI(R5),-(SP) ;STORE NUMBER OF OPERATION INCOMPLETES
6894 031372 066516 000202          ADD P.NSKI(R5),(SP) ;ADD SEEK INCOMPLETES
6895 031376 026526 000104          CMP P.SKET(R5),(SP)+ ;CHECK IF ERROR THRESHOLD EXCEEDED
6896 031402 103020          BHIS STT5$ ;NO, START ERROR RECOVERY
6897 031404 032777 040020 147526 STT2$: BIT #SW4!SW14,2SWR ;CHECK IF INHIBIT DRIVE DESELECTION
6898 031412 001014          BNE STT5$ ;YES, START RECOVERY SEQUENCE
6899 031414 104401 064047          TYPE ERR205 ;PRINT "ERROR THRESHOLD EXCEEDED"
6900 031420 004737 022642          JSR PC,BUFREL ;RELEASE BUFFER
6901 031424 004737 013004          JSR PC,DROP ;DROP DRIVE
6902 031430 004737 022342          JSR PC,GETBUF ;GET NEW COMMAND, DRIVE, AND BUFFER
6903 031434 005037 001566          CLR ERRPRO ;CLEAR ERROR CURRENT BEING PROCESSED
6904 031440 011000          MOV (R0),R0 ;INDICATE NO RECOVERY SEQUENCE IN PROGRESS
6905 031442 000200          RTS RO ;RETURN
6906
6907 031444 105765 000120  STT5$: TSTB P.IERC(R5) ;CHECK IF INHIBIT ERROR RECOVERY
6908 031450 001010          BNE S$ ;YES, INDICATE UNRECOVERABLE RETURN
6909 031452 105265 000146          INCB P.RECT(R5) ;INCREMENT RETRY COUNT
6910 031456 122765 000010 000146 CMPB #8.,P.RECT(R5) ;CHECK IF RETRY FINISHED
6911 031464 103402          BLO S$ ;YES, CLEAN UP AFTER UNSUCCESSFUL RETRY
6912 031466 005720          TST (R0)+ ;INDICATE RECOVERY SEQUENCE IN PROGRESS
6913 031470 000200          RTS RO ;RETURN
6914
6915 031472 004737 031502  S$: JSR PC,UNRECY ;CLEAN UP FOR UNSUCCESSFUL RECOVERY
6916 031476 011000          MOV (R0),R0 ;INDICATE NO RECOVERY SEQUENCE IN PROGRESS
6917 031500 000200          RTS RO ;RETURN
    
```

```
.SBTTL UNSUCCESSFUL RECOVERY CLEANUP
6918
6919
6920 031502 105037 001644 UNRECY: CLR B RTYSCN ;CLEAR RETRY SECTOR COUNT
6921 031506 004737 022642 JSR PC, BUFREL ;RELEASE BUFFER
6922 031512 032765 000001 000150 BIT #BIT0, P.DSTT(R5) ;CHECK IF DATA HARD ERROR
6923 031520 001415 BEQ 1$ ;NO, CHECK IF DRIVE IS BEING ASSIGNED
6924 031522 005265 000200 INC P.HARD(R5) ;INCREMENT HARD ERROR COUNT
6925 031526 026565 000102 000200 CMP P.UERT(R5), P.HARD(R5) ;CHECK IF HARD ERROR THRESHOLD EXCEEDED
6926 031534 103007 BHIS 1$ ;NO, CHECK IF DRIVE IS BEING ASSIGNED
6927 031536 032777 040020 147374 BIT #SW4:SW14, JSWR ;INHIBIT DRIVE DESELECTION
6928 031544 001003 BNE 1$ ;YES, CHECK IF DRIVE IS BEING ASSIGNED
6929 031546 104401 064047 TYPE ERR205 ;PRINT "ERROR THRESHOLD EXCEEDED"
6930 031552 000413 BR 2$ ;GO DROP DRIVE
6931
6932 031554 105765 000210 1$: TSTB P.ASSN(R5) ;CHECK IF DRIVE IS BEING ASSIGNED
6933 031560 001421 BEQ 5$ ;NO, CLEAN UP ERROR
6934 031562 132765 000010 000210 BITB #BIT3, P.ASSN(R5) ;CHECK IF READ PACK SERIAL NUMBER
6935 031570 001404 BEQ 2$ ;NO, DROP DRIVE
6936 031572 132765 000200 000210 BITB #BIT7, P.ASSN(R5) ;CHECK IF RECAL FOR READ OF
6937 ; PACK SERIAL NUMBER
6938 031600 001441 BEQ 10$ ;NO, ISSUE RECALIBRATE AND TRY AGAIN
6939 031602 104401 064012 2$: TYPE ERR204 ;TYPE "ERROR RECOVERY UNSUCCESSFUL"
6940 031606 004737 013004 JSR PC, DROP ;DROP DRIVE FROM TEST SEQUENCE
6941 031612 004737 022342 JSR PC, GETBUF ;GET NEW DRIVE, COMMAND, AND BUFFER
6942 031616 005037 001566 CLR ERRPRO ;CLEAR ERROR CURRENT BEING PROCESSED
6943 031622 000207 RTS PC ;RETURN
6944
6945 031624 105765 000120 5$: TSTB P.IERC(R5) ;CHECK IF INHIBIT RETRY
6946 031630 001006 BNE 6$ ;YES, DO NOT PRINT MESSAGE
6947 031632 032777 020000 147300 BIT #SW13, JSWR ;CHECK IF INHIBIT PRINT OUT
6948 031640 001002 BNE 6$ ;YES, CONTINUE
6949 031642 104401 064012 TYPE ERR204 ;TYPE "ERROR RECOVERY UNSUCCESSFUL"
6950 031646 010446 6$: MOV R4, -(SP) ;STORE R4 ON STACK
6951 031650 116504 000000 MOV B P.DRVN(R5), R4 ;GET DRIVE NUMBER
6952 031654 156437 001514 001513 BISB INTMSK(R4), 0.OVER ;INDICATE IMPLIED SEEKS ARE ALLOWED
6953 031662 012604 MOV (SP)+, R4 ;RESTORE R4
6954 031664 004037 051406 JSR RO, Q.PUSH ;PUT PARAMETER BLOCK IN DRIVE
6955 031670 001314 AVAILQ ;AVAILABLE QUEUE
6956 031672 004737 022342 JSR PC, GETBUF ;GET NEW DRIVE, COMMAND, AND BUFFER
6957 031676 005037 001566 CLR ERRPRO ;CLEAR ERROR CURRENT BEING PROCESSED
6958 031702 000207 RTS PC ;RETURN
6959
6960 031704 122765 000010 000126 10$: CMPB #8, P.RSEC(R5) ;CHECK IF ALL 16-BIT MODE SECTORS HAVE BEEN TRIED
6961 031712 101733 BLOS 2$ ;YES, DROP DRIVE
6962 031714 010446 MOV R4, -(SP) ;STORE R4 ON STACK
6963 031716 116504 000000 MOV B P.DRVN(R5), R4 ;GET DRIVE NUMBER FOR INDEX
6964 031722 156437 001514 001513 BISB INTMSK(R4), 0.OVER ;INDICATE THAT IMPLIED SEEKS ARE ALLOWED
6965 031730 012604 MOV (SP)+, R4 ;RESTORE R4
6966 031732 152765 000200 000210 BISB #BIT7, P.ASSN(R5) ;INDICATE THAT RECALIBRATE HAS BEEN ISSUED
6967 031740 005065 000146 CLR P.RECT(R5) ;CLEAR ERROR FLAGS
6968 031744 005065 000150 CLR P.DSTT(R5)
6969 031750 005065 000212 CLR P.ERR(R5)
6970 031754 112765 000113 000001 MOV B #RECAL, P.CMND(R5) ;LOAD RECALIBRATE COMMAND
6971 031762 004037 051406 JSR RO, Q.PUSH ;PUT PARAMETER BLOCK IN COMMAND
6972 031766 001324 CINITQ ;INITIATION QUEUE
6973 031770 004737 022342 JSR PC, GETBUF ;GO ALLOCATE MEMORY BUFFER
```

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 157
UNSUCCESSFUL RECOVERY CLEANUP

SEQ 0155

6974 031774 005037 001566
6975 032000 000207

CLR ERRPRO
RTS PC

:CLEAR ERROR PROCESSING IN PROGRESS
:RETURN

```

        .SBTTL  ERROR RECOVERY SEQUENCE
6976
6977
6978 032002 032765 000200 000212 ERCVY: BIT      #BIT7,P.ERR(R5) ;CHECK IF UNSOLICITED INTERRUPT
6979 032010 001441          BEQ      3$      ;NO, CHECK IF ERROR INFORMATION STILL VALID
6980 032012 032777 002000 147120 BIT      #SW10,2SWR ;CHECK IF BELL ON ERROR
6981 032020 001402          BEQ      1$      ;NO, CONTINUE PROCESSING
6982 032022 104401 001160          TYPE    $BELL   ;RING BELL
6983 032026 032777 020000 147104 1$: BIT      #SW13,2SWR ;CHECK IF INHIBIT PRINT OUT
6984 032034 001020          BNE      2$      ;YES, CHECK IF HALT ON ERROR
6985 032036 116537 000000 056357 MOVB    P.DRVN(R5),OPR011 ;LOAD DRIVE NUMBER FOR PRINT OUT
6986 032044 152737 000060 056357 BISB    #60,OPR011 ;MAKE IT ASCII
6987 032052 104401 056340          TYPE    OPR010 ;TYPE "DRIVE N"
6988 032056 004737 043670          JSR     PC,PRITIM ;PRINT TIME
6989 032062 104401 061663          TYPE    ERRO26  ;TYPE "INT FORM DRV NOT UNDER TEST"
6990 032066 004737 026074          JSR     PC,PRISTT ;PRINT CONTROLLER REGISTERS
6991 032072 004737 025720          JSR     PC,PRDSTT ;PRINT DRIVE STATUS
6992 032076 004737 031250          JSR     PC,HLT000 ;CHECK IF HALT ON ERROR
6993 032102 005065 000212          CLR     P.ERR(R5) ;CLEAR ERROR FLAGS
6994 032106 005037 001566          CLR     ERRPRO  ;CLEAR ERROR BEING PROCESSED
6995 032112 000207          RTS      PC      ;RETURN
6996
6997 032114 032765 000020 000212 3$: BIT      #BIT4,P.ERR(R5) ;CHECK IF SERVICING DRIVE SEIZED TIME OUT
6998 032122 001427          BEQ      5$      ;NO, CONTINUE
6999 032124 004737 030000          JSR     PC,PR1000 ;PRINT HEADER
7000 032130 104401 062753          TYPE    ERRO59  ;PRINT "TIME OUT BECAUSE DRIVE SEIZED"
7001 032134 032777 020000 146776 BIT      #SW13,2SWR ;CHECK IF INHIBIT PRINT OUT
7002 032142 001002          BNE      4$      ;YES, DO NOT PRINT CURRENT GENERATED COMMAND
7003 032144 004737 030244          JSR     PC,PR1002 ;PRINT CURRENT GENERATED COMMAND
7004 032150 005265 000206          INC     P.NER(R5) ;INCREMENT OTHER ERRORS
7005 032154 004737 031250          JSR     PC,HLT000 ;CHECK IF HALT ON ERROR
7006 032160 004737 022642          JSR     PC,BUFREL ;RELEASE BUFFER
7007 032164 004737 013004          JSR     PC,DROP  ;DROP DRIVE
7008 032170 004737 022342          JSR     PC,GETBUF ;GET NEW DRIVE, COMMAND, AND BUFFER
7009 032174 005037 001566          CLR     ERRPRO  ;CLEAR ERROR BEING PROCESSED
7010 032200 000207          RTS      PC      ;RETURN
7011
7012 032202 032765 000400 000212 5$: BIT      #BIT8,P.ERR(R5) ;CHECK IF TIME OUT BECAUSE HEADS DID NOT RELOAD
7013 032210 001404          BEQ      6$      ;NO, CONTINUE
7014 032212 004037 030464          JSR     RO,PR1100 ;PRINT TIME OUT BECAUSE HEADS DID NOT RELOAD
7015 032216 063306          ERR074
7016 032220 000207          RTS      PC      ;RETURN
7017
7018 032222 032765 000040 000212 6$: BIT      #BIT5,P.ERR(R5) ;CHECK IF TIME OUT DURING DRIVE POSITIONING
7019 032230 001425          BEQ      7$      ;NO, CONTINUE
7020 032232 042765 100000 000150 BIC     #BIT15,P.DSTT(R5) ;RESET ERROR ENQUEUED
7021 032240 052765 000100 000150 BIS     #BIT6,P.DSTT(R5) ;SET READ STATUS ISSUED
7022 032246 116537 000001 001666 MOVB    P.CMND(R5),POSCMD ;STORE COMMAND IN ERROR
7023 032254 042765 000002 000014 BIC     #DRVPOS,P.PRST(R5) ;RESET DRIVE POSITIONING
7024 032262 004737 036746          JSR     PC,GETCUR ;STORE PRESENT STATUS FOR PRINT OUT
7025 032266 112765 000141 000001 MOVB    #RDSTAT,P.CMND(R5) ;GET DRIVE STATUS
7026 032274 004037 051406          JSR     RO,Q.PUSH ;ENQUEUE COMMAND IN COMMAND
7027 032300 001324          CINITQ ;INITIATION QUEUE
7028 032302 000LJ7          RTS      PC      ;RETURN
7029
7030 032304 032765 000100 000212 7$: BIT      #BIT6,P.ERR(R5) ;CHECK IF ERROR WHILE ENQUEUED
7031 032312 001404          BEQ      8$      ;NO, CONTINUE
    
```

```

7032 032314 004037 030464          JSR    RO,PRI100      ;PRINT HEADER
7033 032320 061543          ERR022                ;TYPE "ERROR WHILE WAITING TO
7034                                ;REPORT ERROR"
7035 032322 000207          RTS    PC            ;RETURN
7036
7037 032324 032765 004000 000016 8$:  BIT    #CTO,P.CS1(R5) ;CHECK IF CONTROLLER TIME OUT
7038 032332 001404          BEQ    10$           ;NO, CONTINUE
7039 032334 004037 037322          JSR    RO,CONERR     ;GO REPORT ERROR
7040 032340 061455          ERR017                ;
7041 032342 000207          RTS    PC            ;RETURN
7042
7043 032344 032765 002000 000020 9$:  BIT    #PGE,P.CS2(R5) ;CHECK IF PROGRAMMING EPROR
7044 032352 001404          BEQ    10$           ;NO, CONTINUE
7045 032354 004037 037322          JSR    RO,CONERR     ;PRINT HEADER
7046 032360 061511          ERR020                ;TYPE "PROG ERROR"
7047 032362 000207          RTS    PC            ;RETURN
7048
7049 032364 032765 000001 000034 10$: BIT    #ILC,P.ER(R5)  ;CHECK IF ILLEGAL FUNCTION CODE
7050 032372 001404          BEQ    11$           ;NO, CONTINUE
7051 032374 004037 037322          JSR    RO,CONERR     ;PRINT HEADER
7052 032400 061524          ERR021                ;TYPE "ILLEGAL FUNCTION CODE"
7053 032402 000207          RTS    PC            ;RETURN
7054
7055 032404 032765 010000 000014 11$: BIT    #DRVSZD,P.PRST(R5);CHECK FOR DRIVE AVAIL LOST
7056 032412 001404          BEQ    12$           ;BR IF NOT
7057 032414 004037 030464          JSR    RO,PRI100     ;PRINT HEADER
7058 032420 062002          ERR029                ;TYPE "DRIVE BECAME NOT AVAIL"
7059 032422 000207          RTS    PC            ;RETURN
7060
7061 032424 032765 010000 000020 12$: BIT    #NED,P.CS2(R5) ;CHECK FOR NON-EXISTENT DRIVE
7062 032432 001404          BEQ    15$           ;NO, CONTINUE
7063 032434 004037 030464          JSR    RO,PRI100     ;PRINT HEADER
7064 032440 061610          ERR023                ;TYPE "NON-EXISTENT DRIVE"
7065 032442 000207          RTS    PC            ;RETURN
7066
7067 032444 032765 000400 000020 15$: BIT    #UFE,P.CS2(R5) ;CHECK IF UNIT FIELD ERROR
7068 032452 001404          BEQ    18$           ;NO, CONTINUE
7069 032454 004037 030464          JSR    RO,PRI100     ;PRINT HEADER
7070 032460 061643          ERR025                ;TYPE "UNIT FIELD ERROR"
7071 032462 000207          RTS    PC            ;RETURN
7072
7073 032464 032765 020000 000016 18$: BIT    #SPAR,P.CS1(R5) ;CHECK IF CONTROLLER DETECTED BAD PARITY
7074 032472 001004          BNE    19$           ;YES, PRINT MESSAGE
7075 032474 032765 000010 000034 19$: BIT    #DRPAR,P.ER(R5) ;CHECK IF DRIVE DETECTED BAD PARITY
7076 032502 001437          BEQ    25$           ;NO, CONTINUE
7077 032504 004037 030654          JSR    RO,PRI200     ;PRINT HEADER
7078 032510 061627          ERR024                ;PRINT "SERCON PARITY"
7079 032512 004037 031300          JSR    RO,NERSTT     ;TAKE STATISTICS FOR OTHER ERRORS
7080 032516 032764          60$                 ;ERROR RECOVERY SEQUENCE FINISHED RETURN
7081 032520 032765 020000 000016 22$: BIT    #SPAR,P.CS1(R5) ;CHECK IF CONTROLLER DETECTD SERCON PARITY ERROR
7082 032526 001021          BNE    22$           ;YES, GO RETRY COMMAND
7083 032530 032765 003400 000150 22$: BIT    #BIT8!BIT9!BIT10,P.DST(R5) ;CHECK IF NO POSITIONING
7084 032536 001015          BNE    22$           ;YES, RETRY COMMAND
7085 032540 132765 000007 000210 22$: BIT    #BIT0!BIT1!BIT2,P.ASSN(R5) ;CHECK IF NO POSITION
7086                                ;DURING DRIVE ASSIGNMENT
7087 032546 001011          BNE    22$           ;YES, RETRY COMMAND

```

7088	032550	042765	030000	700150		BIC	#BIT12!BIT13,P.DSTT(R5)	:RESET RE-SEEK AND RE-OFFSET
7089	032556	052765	004000	000150		BIS	#BIT11,P.DSTT(R5)	:SET RECALIBRATE DRIVE
7090	032564	112765	000113	000001		MOVB	#RECAL,P.CMND(R5)	:LOAD RECALIBRATE COMMAND
7091	032572	004037	051406		22\$:	JSR	RO,Q.PUSH	:PUT PARAMETER BLOCK IN COMMAND
7092	032576	001324				CINITQ		:INITIATION QUEUE
7093	032600	000207				RTS	PC	:RETURN
7094								
7095	032602	032765	004000	000020	25\$:	BIT	#NEM,P.CS2(R5)	:CHECK IF NON-EXISTENT MEMORY
7096	032610	001404				BEQ	30\$:NO CONTINUE
7097	032612	004037	037322			JSR	RO,CONERR	:PRINT HEADER
7098	032616	062277				ERR044		:PRINT "NON-EXISTENT MEMORY"
7099	032620	000207				RTS	PC	:RETURN
7100								
7101	032622	032765	020000	000020	30\$:	BIT	#UPE,P.CS2(R5)	:CHECK IF UNIBUS PARITY
7102	032630	001404				BEQ	35\$:NO CONTINUE
7103	032632	004037	037322			JSR	RO,CONERR	:PRINT HEADER
7104	032636	062315				ERR045		:PRINT "UNIBUS PARITY"
7105	032640	000207				RTS	PC	:RETURN
7106								
7107	032642	004037	037512		35\$:	JSR	RO,TSTDRP	:CHECK IF DROP DRIVE ERROR
7108	032646	032764				60\$:YES, ERROR RETURN
7109	032650	032765	000002	000034		BIT	#SKI,P.ER(R5)	:CHECK IF SEEK INCOMPLETE
7110	032656	001404				BEQ	45\$:NO CONTINUE
7111	032660	004037	030654			JSR	RO,PR1200	:PRINT HEADER
7112	032664	062201				ERR038		:PRINT "SEEK INCOMPLETE"
7113	032666	004037	031332			JSR	RO,SKISTT	:TAKE SEEK INCOMPLETE STATISTICS
7114	032672	032764				60\$:ERROR RECOVERY FINISHED RETURN
7115	032674	000412				BR	48\$:GO ISSUE RECALIBRATE
7116								
7117	032676	032765	000040	000036	45\$:	BIT	#DROT,P.D.(R5)	:CHECK IF DRIVE OFF TRACK
7118	032704	001420				BEQ	50\$:NO CONTINUE
7119	032706	004037	030654			JSR	RO,PR1200	:PRINT HEADER
7120	032712	062216				ERR039		:PRINT "DRIVE OFF TRACK"
7121	032714	004037	031300			JSR	RO,NERSTT	:TAKE STATISTICS ON OTHER ERRORS
7122	032720	032764				60\$:END OF ERROR RECOVERY RETURN
7123	032722	042765	034000	000150	48\$:	BIC	#BIT11!BIT12!BIT13,P.DSTT(R5)	:RESET STATUS
7124	032730	052765	004000	000150		BIS	#BIT11,P.DSTT(R5)	:SET RECALIBRATE ISSUED
7125	032736	112765	000113	000001		MOVB	#RECAL,P.CMND(R5)	:LOAD RECALIBRATE
7126	032744	000712				BR	22\$:ENQUEUE COMMAND
7127								
7128	032746	032765	001000	000034	50\$:	BIT	#COE,P.ER(R5)	:CHECK IF CYLINDER ADDRESS OVERFLOW
7129	032754	001404				BEQ	EO\$:NO CONTINUE
7130	032756	004037	037322			JSR	RO,CONERR	:PRINT HEADER
7131	032762	062415				ERR051		:PRINT "CYLINDER ADDRESS OVERFLOW"
7132	032764	000207			60\$:	RTS	PC	:RETURN
7133								
7134	032766	005765	000022		EO\$:	TS-	P.WCR(R5)	:CHECK IF FINAL WORD COUNT IS ZERO
7135	032772	001410				BEQ	5\$:YES, CONTINUE
7136	032774	026565	000022	000012		CMP	P.WCR(R5),P.WC(R5)	:CHECK THAT FINAL WORD COUNT IS LESS
7137								:THAN OR EQUAL TO INITIAL WORD COUNT
7138	033002	103004				BHIS	5\$:YES, CONTINUE
7139	033004	004037	037322			JSR	RO,CONERR	:TYPE "WORD COUNT INVALID"
7140	033010	063171				ERR070		
7141	033012	000207				RTS	PC	:RETURN
7142								
7143	033014	032765	001400	000016	5\$:	BIT	#BA16!BA17,P.CS1(R5)	:CHECK IF BUS ADDRESS BITS 16 OR 17 SET


```

7144 033022 001015      BNE      8$      ;YES, REPORT ERROR
7145 033024 026565 000024 000010  CMP      P.BAR(R5),P.BALO(R5) ;CHECK THAT FINAL BUS ADDRESS IS GREATER
7146                                     ; THAN OR EQUAL TO INITIAL BUS ADDRESS
7147 033032 103411      BLO      8$      ;NO REPORT ERROR
7148 033034 016546 000012  MOV      P.WC(R5),-(SP) ;PUT INITIAL WORD COUNT ON STACK
7149 033040 005416      NEG      (SP)      ;MAKE IT POSITIVE
7150 033042 006316      ASL      (SP)      ;MULTIPLY BY 2
7151 033044 066516 000010  ADD      P.BALO(R5),(SP) ;CALCULATE MAXIMUM FINAL ADDRESS
7152 033050 026526 000024  CMP      P.BAR(R5),(SP)+ ;MAKE SURE IT IS NOT TOO LARGE
7153 033054 101404      BLOS    10$      ;GOOD ADDRESS CONTINUE
7154 033056 004037 037322  8$:     JSR      RO,CONERR ;TYPE "BUS ADDRESS INVALID"
7155 033062 063214      ERR071
7156 033064 000207      RTS      PC      ;RETURN
7157
7158 033066 026565 000030 000002 10$:    CMP      P.DCYL(R5),P.CYLN(R5) ;CHECK THAT FINAL CYLINDER IS GREATER
7159                                     ; THAN OR EQUAL TO INITIAL CYLINDER
7160 033074 001405      BEQ     15$      ;IF EQUAL CHECK TRACK/SECTOR
7161 033076 101014      BHI     E1$      ;IF GREATER THAN DETERMINE ERROR
7162 033100 004037 037322  JSR      RO,CONERR ;TYPE "CYLIN ADDRESS INVALID"
7163 033104 063240      ERR072
7164 033106 000207      RTS      PC      ;RETURN
7165
7166 033110 026565 000026 000004 15$:    CMP      P.DTS(R5),P.SECT(R5) ;CHECK TRACK/SECTOR GREATER THAN OR
7167                                     ; EQUAL TO INITIAL VALUES
7168 033116 103004      BHIS   E1$      ;YES, GO INTO ERROR RECOVERY
7169 033120 004037 037322  JSR      RO,CONERR ;TYPE "SECTOR/TRACK INVALID"
7170 033124 063261      ERR073
7171 033126 000207      RTS      PC
7172
7173 033130 032765 010000 000034 E1$:    BIT      #0TE,P.ER(R5) ;CHECK FOR DRIVE TIMING ERROR
7174 033136 001002      BNE     2$      ;YES, PROCESS DRIVE TIMING ERROR
7175 033140 000137 033632  JMP      E2$      ;NO, CONTINUE
7176
7177 033144 105765 000147      2$:     TSTB   P.RERD(R5) ;CHECK IF FIRST DATA ERROR IN
7178                                     ; RETRY SEQUENCE
7179 033150 001117      BNE     30$      ;NO REPORT ERROR
7180 033152 016537 000010 001634  MOV      P.BALO(R5),RTYBA ;STORE SUPPLIED BUS ADDRESS
7181 033160 016537 000012 001636  MOV      P.WC(R5),RTYWC ;STORE SUPPLIED WORD COUNT
7182 033166 005437 001636      NEG      RTYWC      ;MAKE IT POSITIVE
7183 033172 022737 000400 001636  CMP      #256.,RTYWC ;CHECK IF LESS THAN OR EQUAL TO A SECTOR
7184 033200 103013      BHIS   21$      ;YES, WORK WITH ONE SECTOR
7185 033202 026565 000026 000004  CMP      P.DTS(R5),P.SECT(R5) ;CHECK IF STARTING TRACK AND SECTOR
7186 033210 001021      BNE     25$      ;NOT RETRY 2 SECTORS
7187 033212 026565 000030 000002  CMP      P.DCYL(R5),P.CYLN(R5) ;CHECK IF STARTING CYLINDER
7188 033220 001015      BNE     25$      ;NO, RETRY 2 SECTORS
7189 033222 012737 000400 001636  MOV      #256.,RTYWC ;SET WORD COUNT = 1 SECTOR
7190 033230 112737 000001 001644  21$:    MOVB   #1,RTYSCN ;SET SECTOR COUNT = 1
7191 033236 016537 000004 001640  MOV      P.SECT(R5),RTYSEC ;LOAD SECTOR AND TRACK
7192 033244 016537 000002 001642  MOV      P.CYLN(R5),RTYCYL ;LOAD CYLINDER
7193 033252 000456      BR      30$      ;GO REPORT ERROR
7194
7195 033254 016537 000026 001640  25$:    MOV      P.DTS(R5),RTYSEC ;SET SECTOR AND TRACK
7196 033262 016537 000030 001642  MOV      P.DCYL(R5),RTYCYL ;GET CYLINDER
7197 033270 105737 001640      TSTB   RTYSEC      ;CHECK IF SECTOR = 0
7198 033274 001403      BEQ     26$      ;YES, GO DECREMENT TRACK
7199 033276 105337 001640      DECB   RTYSEC      ;DECREMENT SECTOR

```

```

7200 033302 000416          BR      28$          ; DETERMINE WORD COUNT
7201                                ;
7202 033304 112737 000025 001640 26$:  MOVB   #21,RTYSEC  ; LOAD SECTOR = 21
7203 033312 105737 001641          TSTB   RTYTRK   ; CHECK IF TRACK 0
7204 033316 001403          BEQ    27$          ; YES, GO DECREMENT CYLINDER
7205 033320 105337 001641          DECB   RTYTRK   ; DECREMENT TRACK
7206 033324 000405          BR      28$          ; DETERMINE WORD COUNT
7207                                ;
7208 033326 112737 000002 001641 27$:  MOVB   #2,RTYTRK  ; LOAD TRACK = 2
7209 033334 05337 001642          DEC    RTYCYL   ; DECREMENT CYLINDER
7210 033340 004737 037210          JSR    PC,CALWC ; CALCULATE RETRY WORD COUNT
7211 033344 112737 000002 001644 28$:  MOVB   #2,RTYSCN ; LOAD SECTOR COUNT = 2
7212 033352 022737 001000 001636  CMP    #512.,RTYWC ; CHECK IF LESS THAN OR EQUAL
7213 033360 101004          BHI    29$          ; TO SECTORS
7214 033362 012737 001000 001636  MOV    #512.,RTYWC ; MAKE IT 512
7215 033370 000407          BR      30$          ; GO REPORT ERROR
7216                                ;
7217 033372 022737 000400 001636 29$:  CMP    #256.,RTYWC ; CHECK IF ONLY ONE SECTOR
7218 033400 103403          BLO    30$          ; NO, GO REPORT ERROR
7219 033402 112737 000001 001644  MOVB   #1,RTYSCN ; LOAD SECTOR COUNT = 1
7220                                ;
7221 033410 004037 030654          JSR    R0,PR1200 ; PRINT HEADER
7222 033414 062330          ERRO46 ; PRINT DRIVE TIMING ERROR
7223 033416 105765 000120          TSTB   P.IERC(R5) ; CHECK IF INHIBIT ERROR RECOVERY
7224 033422 001016          BNE    35$          ; YES, GO CLEAN UP FOR NEXT COMMAND
7225 033424 105265 000147          INCB   P.RERD(R5) ; INCREMENT REREAD COUNT
7226 033430 122765 000043 000147  CMPB   #43,P.RERD(R5) ; CHECK IF RETRY UNSUCCESSFUL
7227 033436 101410          BLOS   35$          ; YES, GO INDICATE UNSUCCESSFUL
7228                                ; RECOVERY
7229 033440 122765 000121 000123  CMPB   #RDDATA,P.RCMD(R5) ; CHECK IF READ DATA
7230 033446 001407          BEQ    37$          ; YES, GO TO RECOVERY SEQUENCE
7231 033450 122765 000021 000147  CMPB   #21,P.RERD(R5) ; CHECK IF RETRY UNSUCCESSFUL
7232 033456 101026          BHI    40$          ; NO, GO TO RETRY SEQUENCE
7233 033460 004737 031502          JSR    PC,UNRECY ; CLEAN UP FOR UNSUCCESSFUL RECOVERY
7234 033464 000207          RTS    PC        ; RETURN
7235                                ;
7236 033466 122765 000021 000147 37$:  CMPB   #21,P.RERD(R5) ; CHECK IF OFFSET +400 MICRO-INCHES
7237 033474 001017          BNE    40$          ; NO, CONTINUE RECOVERY
7238 033476 122737 000001 001644  CMPB   #1,RTYSCN  ; CHECK IF NUMBER OF SECTORS = 1
7239 033504 001404          BEQ    38$          ; YES, DO OFFSET
7240 033506 122737 000025 001640  CMPB   #21.,RTYSEC ; CHECK IF MID-TRANSFER SEEK
7241                                ; IS REQUIRED
7242 033514 001761          BEQ    35$          ; YES, INDICATE UNSUCCESSFUL RETRY
7243 033516 010446          MOV    R4,-(SP)   ; STORE R4 ON THE STACK
7244 033520 116504 000000          MOVB   P.DRVN(R5),R4 ; GET DRIVE NUMBER FOR INDEX
7245 033524 146437 001514 001513  BICB   INTMSK(R4),O.OVER ; DO NOT SEEK BEFORE DATA
7246                                ; TRANSFER (RESETTING OFFSET)
7247 033532 012604          MOV    (SP)+,R4  ; RESTORE R4
7248                                ;
7249 033534 004037 037026          JSR    R0,DETOFF ; DETERMINE IF OFFSET INSTRUCTION
7250 033540 033630          SO$    ; OFFSET ISSUED RETURN
7251 033542 116565 000123 000001  MOVB   P.RCMD(R5),P.CMND(R5) ; LOAD COMMAND
7252 033550 013765 001640 000004  MOV    RTYSEC,P.SECT(R5) ; LOAD SECTOR AND TRACK
7253 033556 013765 001642 000002  MOV    RTYCYL,P.CYLN(R5) ; LOAD CYLINDER
7254 033564 013765 001634 000010  MOV    RTYBA,P.BALO(R5) ; LOAD BUS ADDRESS
7255 033572 013765 001636 000012  MOV    RTYWC,P.WC(R5) ; LOAD WORD COUNT

```

7256	033600	005465	000012			NEG	P.WC(R5)	:MAKE IT NEGATIVE
7257	033604	122765	000131	000001		CMPB	#WRTCHK,P.CMND(R5)	:CHECK IF WRITE CHECK
7258	033612	001003				BNE	45\$:NO, ISSUE COMMAND
7259	033614	052765	000200	000014		BIS	#W.WCK,P.PRST(R5)	:YES, SET ISSUE WRITE COMMAND
7260	033622	004037	051406		45\$:	JSR	RO,Q.PUSH	:ENQUEUE PARAMETER BLOCK IN
7261	033626	001324				CINITQ		:COMMAND INITIATION QUEUE
7262	033630	000207			50\$:	RTS	PC	:RETURN
7263								
7264	033632	032765	020400	000034	E2\$:	BIT	#OPI!HVRC,P.ER(R5)	:CHECK IF HEADER TYPE ERROR
7265	033640	001002				BNE	2\$:YES, PROCESS ERROR
7266	033642	000137	034170			JMP	E3\$:NO, CONTINUE
7267								
7268	033646	004737	036746		2\$:	JSR	PC,GETCUR	:GET CURRENT REGISTER STATUS
7269	033652	112737	000001	001644		MOVB	#1,RTYSCN	:LOAD RETRY SECTOR COUNT TO 1
7270	033660	016537	000026	001640		MOV	P.DTS(R5),RTYSEC	:STORE CURRENT SECTOR AND TRACK
7271	033666	016537	000030	001642		MOV	P.DCYL(R5),RTYCYL	:STORE CURRENT SECTOR
7272	033674	016537	000010	001634		MOV	P.BALO(R5),RTYBA	:GET INITIAL BUS ADDRESS
7273	033702	016537	000012	001636		MOV	P.WC(R5),RTYWC	:GET INITIAL WORD COUNT
7274	033710	005437	001636			NEG	RTYWC	:MAKE IT POSITIVE
7275	033714	022737	000400	001636		CMP	#256.,RTYWC	:CHECK IF LESS THEN OR EQUAL TO ONE SECTOR
7276	033722	103021				BHIS	20\$:RETRY COMMAND
7277	033724	026565	000026	000004		CMP	P.DTS(R5),P.SECT	:CHECK IF CURRENT SECTOR AND TRACK EQUALS
7278								:INITIAL TRACK AND SECTOR
7279	033732	001004				BNE	15\$:NO, CALCULATE DESIRED SECTOR
7280	033734	026565	000030	000002		CMP	P.DCYL(R5),P.CYL	:CHECK IF CURRENT CYLINDER EQUALS
7281								:INITIAL CYLINDER
7282	033742	001406				BEQ	18\$:YES, USE INITIAL CYLINDER, TRACK, AND SECTOR
7283	033744	004737	037210		15\$:	JSR	PC,CALWC	:CALCULATE WORD COUNT
7284	033750	022737	000400	001636		CMP	#256.,RTYWC	:CHECK IF WORD COUNT LESS THAN OR EQUAL
7285								:TO ONE SECTOR
7286	033756	103003				BHIS	20\$:YES, USE CALCULATED WORD COUNT
7287	033760	012737	000400	001636	18\$:	MOV	#256.,RTYWC	:USE 256 WORDS
7288	033766	015546	000052		20\$:	MOV	P.B10(R5),-(SP)	:STORE DRIVE CYLINDER ADDRESS
7289	033772	042716	160017			BIC	#1C<M.CADD>,(SP)	:THROW AWAY PARITY AND MESSAGE TYPE
7290	033776	006216				ASR	(SP)	:SHIFT 4 BITS RIGHT
7291	034000	006216				ASR	(SP)	
7292	034002	006216				ASR	(SP)	
7293	034004	006216				ASR	(SP)	
7294	034006	022637	001642			CMP	(SP)+,RTYCYL	:CHECK IF CYLINDER IS CORRECT
7295	034012	001420				BEQ	30\$:YES, ISSUE READ HEADER
7296	034014	004037	030654			JSR	RO,PR1200	:REPORT ERROR
7297	034020	062233				ERR040		:TYPE "MISPOS"
7298	034022	004037	031346			JSR	RO,OP1STT	:GET OPI/MISPOSITIONING STATISTICS
7299	034026	034166				60\$:ERROR THRESHOLD EXCEEDED RETURN
7300	034030	052765	004000	000150		BIS	#BIT11,P.DSTT(R5)	:SET RECALIBRATE ISSUED
7301	034036	112765	000113	000001		MOVB	#RECAL,P.CMND(R5)	:LOAD RECALIBRATE IN PARAMETER BLOCK
7302	034044	004037	051406			JSR	RO,Q.PUSH	:PUT PARAMETER BLOCK IN COMMAND
7303	034050	001324				CINITQ		:INITIATION QUEUE
7304	034052	000207				RTS	PC	:RETURN
7305								
7306	034054	010446			30\$:	MOV	R4, -(SP)	:STORE R4 ON THE STACK
7307	034056	113704	001641			MOVB	RTYTRK,R4	:GET TRACK FOR INDEX
7308	034062	136465	001507	000057		BITB	H.HEAD(R4),P.B11	:CHECK IF CORRECT HEAD IS SELECTED
7309	034070	001013				BNE	35\$:YES, CONTINUE
7310	034072	012604				MOV	(SP)+,R4	:RESTORE R4
7311	034074	004037	030654			JSR	RO,PR1200	:PRINT HEAD SELECT ERROR

7312	034100	063105				ERR065		
7313	034102	004037	031346			JSR	RO,OPISTT	;RECORD AS AN OPI
7314	034106	054166				60\$;ERROR THRESHOLD EXCEEDED RETURN
7315	034110	004037	051406			JSR	RO,Q.PUSH	;REISSUE COMMAND
7316	034114	001324				CINITQ		
7317	034116	000207				RTS	PC	;RETURN
7318								
7319	034120	012604			35\$:	MOV	(SP)+,R4	;RESTORE R4
7320	034122	052765	000020	000150		BIS	#BIT4,P.DSTT(R5)	;SET READ ALL HEADERS ISSUED
7321	034130	013765	001640	000004		MOV	RTYSEC,P.SECT(R5)	;LOAD SECTOR AND TRACK
7322	034136	013765	001642	000002		MOV	RTYCYL,P.CYLN(R5)	;LOAD CYLINDER
7323	034144	012765	066006	000010		MOV	#HDBUFF,P.BALO(R5)	;LOAD ADDRESS OF HEADER BUFFER
7324	034152	112765	000164	000001		MOVB	#RDALHD,P.CMND(R5)	;LOAD COMMAND
7325	034160	004037	051406			JSR	RO,Q.PUSH	;PUT PARAMETER BLOCK IN COMMAND
7326	034164	001324				CINITQ		;INITIATION QUEUE
7327	034166	000207			60\$:	RTS	PC	;RETURN
7328								
7329	034170	032765	000200	000034	E3\$:	BIT	#BSE,P.ER(R5)	;CHECK FOR BAD SECTOR ERROR
7330	034176	001403				BEQ	5\$;NO CONTINUE
7331	034200	004737	042356			JSR	PC,BDSECT	;SERVICE BAD SECTOR
7332	034204	000207				RTS	PC	;RETURN
7333								
7334	034206	032765	100000	000020	5\$:	BIT	#DLT,P.CS2(R5)	;CHECK IF DATA LATE
7335	034214	001444				BEQ	E4\$;NO CONTINUE
7336	034216	004737	030000			JSR	PC,PRIO00	;TYPE HEADER
7337	034222	032777	020000	144710		BIT	#SW13,SWR	;CHECK IF INHIBIT PRINT OUT
7338	034230	001004				BNE	10\$;YES, CHECK IF HALT ON ERROR
7339	034232	104401	062350			TYPE	ERR047	;TYPE "DATA LATE"
7340	034236	004737	026074			JSR	PC,PRISTT	;PRINT CONTROLLER STATUS
7341	034242	004737	031250		10\$:	JSR	PC,HLT000	;CHECK IF HALT ON ERROR
7342	034246	005237	001626			INC	DLT CNT	;INCREMENT DATA LATE COUNT
7343	034252	022737	001000	001626		CMP	#T.DLT,DLT CNT	;CHECK IF THRESHOLD EXCEEDED
7344	034260	103006				BHS	15\$;NO REISSUE COMMAND
7345	034262	104401	060533			TYPE	ERR000	;TYPE FATAL ERROR
7346	034266	104401	061136			TYPE	ERR009	;TYPE DATA LATE THRESHOLD EXCEEDED
7347	034272	000000				HALT		;HANG
7348	034274	000776				BR	.-2	
7349								
7350	034276	105765	000150		15\$:	TSTB	P.DSTT(R5)	;CHECK IF FIRST ERROR
7351	034302	100005				BPL	20\$;NO, REISSUE COMMAND
7352	034304	042765	100200	000150		BIC	#BIT15!BIT7,P.DSTT(R5)	;CLEAR ERROR ENQUEUED AND FIRST ERROR
7353	034312	005037	001566			CLR	ERRPRO	;CLEAR ERROR RECOVERY IN PROGRESS
7354	034316	004037	051406		20\$:	JSR	RO,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN COMMAND
7355	034322	001324				CINITQ		;INITIATION QUEUE
7356	034324	000207				RTS	PC	;RETURN
7357								
7358	034326	032765	100000	000034	E4\$:	BIT	#DCK,P.ER(R5)	;CHECK IF DATA CHECK
7359	034334	001002				BNE	2\$;YES, START RECOVERY
7360	034336	000137	035136			JMP	E5\$;NO, CONTINUE
7361								
7362	034342	112737	000001	001644	2\$:	MOVB	#1,RTYSCN	;SET SECTOR COUNT = 1
7363	034350	016537	000004	001640		MOV	P.SECT(R5),RTYSEC	;STORE RETRY SECTOR AND TRACK
7364	034356	016537	000002	001642		MOV	P.CYLN(R5),RTYCYL	;STORE RETRY CYLINDER
7365	034364	016537	000010	001634		MOV	P.BALO(R5),RTYBA	;STORE RETRY BUS ADDRESS
7366	034372	016537	000012	001636		MOV	P.WC(R5),RTYWC	;STORE RETRY WORD COUNT
7367	034400	005437	001636			NEG	RTYWC	;MAKE IT POSITIVE

7368	034404	022737	000400	001636		CMP	#256.,RTYWC	;CHECK IF LESS THAN OR EQUAL TO A SECTOR
7369	034412	103043				BHIS	15\$;YES, CHECK IF ECC CORRECTION
7370	034414	016537	000026	001640		MOV	P.DTS(R5),RTYSEC	;STORE PRESENT TRACK AND SECTOR
7371	034422	016537	000030	001642		MOV	P.DCYL(R5),RTYCYL	;STORE PRESENT CYLINDER
7372	034430	105737	001640			TSTB	RTYSEC	;CHECK IF SECTOR = 0
7373	034434	001403				BEQ	11\$;YES, DECREMENT TRACK
7374	034436	105337	001640			DECB	RTYSEC	;DECREMENT SECTOR
7375	034442	000416				BR	13\$;GO CALCULATE WORD COUNT
7376								
7377	034444	112737	000025	001640	11\$:	MOVB	#21,RTYSEC	;SET SECTOR = 21
7378	034452	105737	001641			TSTB	RTYTRK	;CHECK IF TRACK = 0
7379	034456	001403				BEQ	12\$;YES, DECREMENT CYLINDER
7380	034460	105337	001641			DECB	RTYTRK	;DECREMENT TRACK
7381	034464	000405				BR	13\$;GO CALCULATE WORD COUNT
7382								
7383	034466	112737	000002	001641	12\$:	MOVB	#2,RTYTRK	;SET TRACK = 2
7384	034474	005337	001642			DEC	RTYCYL	;DECREMENT CYLINDER
7385	034500	004737	037210		13\$:	JSR	PC,CALWC	;CALCULATE WORD COUNT
7386	034504	022737	000400	001636		CMP	#256.,RTYWC	;CHECK IF LESS THAN OR EQUAL TO ONE SECTOR
7387	034512	103003				BHIS	15\$;YES, PERFORM ECC CORRECTION
7388	034514	012737	000400	001636		MOV	#256.,RTYWC	;MAKE IT ONE SECTOR
7389	034522	004037	030654		15\$:	JSR	RC,PR1200	;PRINT HEADER
7390	034526	062364				ERR048		;PRINT DATA CHECK
7391	034530	105765	000120			TSTB	P.IERC(R5)	;CHECK IF INHIBIT ERROR RECOVERY
7392	034534	001123				BNE	40\$;YES, INDICATE UNSUCCESSFUL RECOVERY
7393	034536	122765	000121	000123		CMPB	#RDDATA,P.RCMD(R5)	;CHECK IF READ DATA
7394	034544	001101				BNE	30\$;NO, DO NOT DO ECC CORRECTION
7395	034546	032765	000100	000034		BIT	#ECH,P.ER(R5)	;CHECK IF CORRECTABLE
7396	034554	001075				BNE	30\$;NO, DO NOT DO ECC CORRECTION
7397	034556	022765	010040	000060		CMP	#10040,P.EPOS(R5)	;CHECK IF ECC POSITION TOO LARGE
7398	034564	101406				BLOS	16\$;YES, REPORT ERROR
7399	034566	005765	000060			TST	P.EPOS(R5)	;CHECK IF ECC POSITION = 0
7400	034572	001403				BEQ	16\$;YES, REPORT ERROR
7401	034574	005765	000062			TST	P.EPAT(R5)	;CHECK IF ECC PATTERN = 0
7402	034600	001004				BNE	17\$;NO, DO ECC CORRECTION
7403	034602	004037	037322		16\$:	JSR	RD,CONERR	;TYPE "ECC LOGIC ERROR"
7404	034606	063130				ERR066		
7405	034610	000207				RTS	PC	;RETURN
7406								
7407	034612	004737	026160		17\$:	JSR	PC,ECCOR	;DO ECC CORRECTION
7408	034616	105765	000210			TSTB	P.ASSN(R5)	;CHECK IF IN ASSIGNMENT SEQUENCE
7409	034622	001017				BNE	18\$;YES, DO NOT CHECK ECC CORRECTION
7410	034624	013765	001634	000214		MOV	RTYBA,P.ERHD(R5)	;LOAD START OF SECTOR FOR
7411								;DATA COMPARISON
7412	034632	013765	001636	000222		MOV	RTYWC,P.BFCP(R5)	;LOAD NUMBER OF WORDS OF
7413	034640	013765	001636	000220		MOV	RTYWC,P.CMLG(R5)	;SECTOR PRESENT
7414	034646	112765	177777	000117		MOVB	#-1,P.ECMP(R5)	;SET ECC COMPARE FLAG
7415	034654	004737	023362			JSR	PC,CMPBUF	;CHECK ECC CORRECTION
7416	034660	034670				ZOS		;INDICATE ECC MISCORRECTION
7417	034662	004737	036276		18\$:	JSR	PC,SURECY	;INDICATE SUCCESSFUL RECOVERY
7418	034666	000207				RTS	PC	;RETURN
7419								
7420	034670	032777	020000	144242	20\$:	BIT	#SW13,2SWR	;CHECK IF INHIBIT ERROR PRINT OUT
7421	034676	001024				BNE	30\$;YES, GO TO RECOVERY SEQUENCE
7422	034700	104401	063024			TYPE	.ERR060	;TYPE "ECC MISCORRECTION"
7423	034704	104401	064303			TYPE	.ERR211	;TYPE "RKPOS"

7424	034710	016546	000060			MOV	P.EPOS(R5),-(SP)	:STORE RKPOS FOR PRINT OUT
7425	034714	004737	052774			JSR	PC,BINOC	:CONVERT TO OCTAL
7426	034720	104401	001165			TYPE	,\$CRLF	:TYPE <CR><LF>
7427	034724	104401	064312			TYPE	,ERR212	:TYPE "RKPAT"
7428	034730	016546	000062			MOV	P.EPAT(R5),-(SP)	:STORE RKPAT FOR PRINT OUT
7429	034734	004737	052774			JSR	PC,BINOC	:CONVERT TO OCTAL
7430	034740	104401	001165			TYPE	,\$CRLF	:TYPE <CR><LF>
7431	034744	004737	036024			JSR	PC,PRERWD	:PRINT WORD IN ERROR
7432	034750	105265	000147			INCB	P.RERD(R5)	:INCREMENT RETRY COUNT
7433	034754	122765	000043	000147		CMPB	#43,P.RERD(R5)	:CHECK IF UNSUCCESSFUL RETRY
7434	034762	101410				BLOS	40\$:YES, INDICATE UNSUCCESSFUL RETRY
7435	034764	122765	000121	000123		CMPB	#RDDATA,P.RCMD(R5)	:CHECK IF READ DATA
7436	034772	001407				BEQ	45\$:YES, CHECK FOR OFFSET
7437	034774	122765	000021	000147		CMPB	#21,P.RERD(R5)	:CHECK IF ABOUT TO ENTER OFFSET OPERATIONS
7438	035002	101021				BHI	50\$:NO, DO REREAD
7439	035004	004737	031502		40\$:	JSR	PC,UNRECY	:INDICATE UNSUCCESSFUL RECOVERY
7440	035010	000207				RTS	PC	:RETURN
7441								
7442	035012	122765	000021	000147	45\$:	CMPB	#21,P.RERD(R5)	:CHECK IF OFFSET +400 MICRO-INCHES
7443	035020	001007				BNE	48\$:NO, GO DETERMINE IF OFFSET IS TO
7444								:BE ISSUED
7445	035022	010446				MOV	R4, -(SP)	:STORE R4 ON STACK
7446	035024	116504	000000			MOVB	P.DRVN(R5),R4	:GET DRIVE NUMBER AS INDEX
7447	035030	146437	001514	001513		BICB	INTMSK(R4),O.OVER	:RESET ISSUE IMPLIED SEEK
7448	035036	012604				MOV	(SP)+R4	:RESTORE R4
7449	035040	004037	037026		48\$:	JSR	RO,DETOFF	:DETERMINE IF OFFSET IS TO BE ISSUED
7450	035044	035134				BOS		:OFFSET ISSUED RETURN
7451	035046	116565	000123	000001	50\$:	MOVB	P.RCMD(R5),P.CMND(R5)	:REISSUE LAST COMMAND
7452	035054	013765	001640	000004		MOV	RTYSEC,P.SECT(R5)	:LOAD SECTOR AND TRACK
7453	035062	013765	001642	000002		MOV	RTYCYL,P.CYLN(R5)	:LOAD CYLINDER
7454	035070	013765	001634	000010		MOV	RTYBA,P.BALO(R5)	:LOAD BUS ADDRESS
7455	035076	013765	001636	000012		MOV	RTYWC,P.WC(R5)	:LOAD WORD COUNT
7456	035104	005465	000012			NEG	P.WC(R5)	:MAKE IT NEGATIVE
7457	035110	122765	000131	000001		CMPB	#WRTCHK,P.CMND(R5)	:CHECK IF WRITE CHECK
7458	035116	001003				BNE	55\$:NO, GO ISSUE COMMAND
7459	035120	052765	000200	000014		BIS	#W.WCK,P.PRST(R5)	:ISSUE WRITE BEFORE WRITE CHECK
7460	035126	004037	051406		55\$:	JSR	RO,Q.PUSH	:PUT PARAMETER BLOCK ON COMMAND
7461	035132	001324				CINITQ		:COMMAND INITIATION QUEUE
7462	035134	000207			60\$:	RTS	PC	:RETURN
7463								
7464	035136	032765	040000	000020	55\$:	BIT	#WCE,P.CS2(R5)	:CHECK IF WRITE CHECK ERROR
7465	035144	001002				BNE	1\$:YES, PROCESS WRITE CHECK ERROR
7466	035146	000137	035310			JMP	E10\$:NO, CONTINUE
7467								
7468	035152	004037	030654		1\$:	JSR	RO,PR1200	:PRINT WRITE CHECK ERROR
7469	035156	062376				ERR050		
7470	035160	105765	000120			TSTB	P.IERC(R5)	:CHECK IF INHIBIT ERROR CORRECTION AND RETRY
7471	035164	001006				BNE	5\$:YES, INITIATE NEXT COMMAND
7472	035166	105265	000147			INCB	P.RERD(R5)	:INCREMENT RETRY COUNT
7473	035172	122765	000021	000147		CMPB	#21,P.RERD(R5)	:CHECK IF 16 REREADS OCCURRED
7474	035200	101003				BHI	10\$:NO, START RECOVERY SEQUENCE
7475	035202	004737	031502		5\$:	JSR	PC,UNRECY	:INDICATE UNSUCCESSFUL RECOVERY
7476	035206	000207				RTS	PC	:RETURN
7477								
7478	035210	016537	000024	001634	10\$:	MOV	P.BAR(R5),RTYBA	:GET ADDRESS OF WORD IN ERROR
7479	035216	162737	000002	001634		SUB	#2,RTYBA	:ADJUST ADDRESS

7480	035224	004737	035612			JSR	PC,CALSEC	;CALCULATE SECTOR, TRACK AND CYLINDER
7481	035230	116565	000123	000001		MOV	P.RCMD(R5),P.CMND(R5)	;GET PREVIOUS COMMAND
7482	035236	013765	001640	000004		MOV	RTYSEC,P.SECT(R5)	;GET SECTOR AND TRACK
7483	035244	013765	001642	000002		MOV	RTYCYL,P.CYLN(R5)	;GET CYLINDER
7484	035252	013765	001634	000010		MOV	RTYBA,P.BALO(R5)	;GET BUS ADDRESS
7485	035260	013765	001636	000012		MOV	RTYWC,P.WC(R5)	;GET WORD COUNT
7486	035266	005465	000012			NEG	P.WC(R5)	;MAKE IT NEGATIVE
7487	035272	052765	000200	000014		BIS	#W.WCK,P.PRST(R5)	;SET ISSUE WRITE BEFORE WRITE CHECK
7488	035300	004037	051406			JSR	RD,Q.PUSH	;ENQUEUE PARAMETER IN
7489	035304	001324				CINITQ		;COMMAND INITIATION QUEUE
7490	035306	000207				RTS	PC	;RETURN
7491								
7492								
7493	035310	032765	000004	000212	E10\$:	BIT	#BIT2,P.ERR(R5)	;CHECK IF DATA COMPARE ERROR
7494	035316	001001				BNE	2\$;YES, CONTINUE PROCESSING ERROR
7495	035320	000000				HALT		;*** UNKNOWN ENTRY
7496	035322	005046			2\$:	CLR	-(SP)	;MAKE ROOM ON STACK
7497	035324	116516	000217			MOV	P.ERAD(R5),(SP)	;GET WORD IN ERROR
7498	035330	006316				ASL	(SP)	;MULTIPLY BY 2
7499	035332	066516	000214			ADD	P.ERAD(R5),(SP)	;CALCULATE ADDRESS OF MISCOMPARED WORD
7500	035336	012637	001634			MOV	(SP)+,RTYBA	;LOAD ADDRESS OF BAD WORD
7501	035342	004737	035612			JSR	PC,CALSEC	;CALCULATE CYLINDER, TRACK, AND SECTOR
7502	035346	032765	000002	000150		BIT	#BIT1,P.DSTT(R5)	;CHECK IF DATA TYPE ERROR HAS ALREADY OCCURRED
7503	035354	001037				BNE	10\$;YES, GO TO RETRY SEQUENCE
7504	035356	052765	000002	000150		BIS	#BIT1,P.DSTT(R5)	;SET DATA TYPE ERROR OCCURRED
7505	035364	100003				BPL	3\$;CHECK IF FIRST ERROR
7506	035366	052765	000001	000150		BIS	#BIT0,P.DSTT(R5)	;SET INITIAL DATA ERROR
7507	035374	004737	030000		3\$:	JSR	PC,PR1000	;PRINT HEADER
7508	035400	032777	020000	143532		BIT	#SW13,JSWR	;CHECK IF INHIBIT PRINT OUT
7509	035406	001013				BNE	5\$;YES, CHECK IF HALT ON ERROR
7510	035410	104401	063050			TYPE	ERR062	;TYPE DATA COMPARE ERROR
7511	035414	005765	000150			TST	P.DSTT(R5)	;CHECK IF FIRST ERROR
7512	035420	100402				BMI	4\$;YES, GO PRINT ERROR WORD
7513	035422	104401	063721			TYPE	ERR202	;TYPE ERROR DURING ERROR RECOVERY
7514	035426	004737	036024		4\$:	JSR	PC,PRERWD	;PRINT WORD IN ERROR
7515	035432	004737	030244			JSR	PC,PR1002	;PRINT CURRENTLY SUPPLIED COMMAND
7516	035436	042765	100200	000150	5\$:	BIC	#BIT15:BIT7,P.DSTT(R5)	;CLEAR FIRST ERROR AND ERROR ENQUEUED
7517	035444	004737	027442			JSR	PC,PRIBUF	;GO PRINT BUFFER
7518	035450	004737	031250			JSR	PC,HLT000	;CHECK IF HALT ON ERROR
7519	035454	105765	000120		10\$:	TSTB	P.ITERC(R5)	;CHECK IF INHIBIT ERROR CORRECTION AND RECOVERY
7520	035460	001006				BNE	15\$;YES, INDICATE UNSUCCESSFUL RECOVERY
7521	035462	105265	000147			INCB	P.RERD(R5)	;INCREMENT RETRY COUNT
7522	035466	122765	000043	000147		CMPB	#43,P.RERD(R5)	;CHECK IF ERROR RECOVERY UNSUCCESSFUL
7523	035474	101003				BHI	20\$;NO, GO TO RECOVERY SEQUENCE
7524	035476	004737	031502		15\$:	JSR	PC,UNRECY	;INDICATE UNSUCCESSFUL RECOVERY
7525	035502	000207				RTS	PC	;RETURN
7526								
7527	035504	122765	000021	000147	20\$:	CMPB	#21,P.RERD(R5)	;CHECK IF ENTERING OFFSET
7528	035512	001007				BNE	25\$;NO, DETERMINE IF TIME TO CHANGE OFFSET
7529	035514	010446				MOV	R4, -(SP)	;STORE R4 ON STACK
7530	035516	116504	000000			MOV	P.DRVN(R5),R4	;GET DRIVE NUMBER FOR INDEX
7531	035522	146437	001514	001513		BICB	INTMSK(R4),0.OVER	;RESET SEEK BEFORE DATA TRANSFER
7532	035530	012604				MOV	(SP)+,R4	;RESTORE R4
7533	035532	004037	037026		25\$:	JSR	RD,DETOFF	;DETERMINE IF OFFSET TO BE ISSUED
7534	035536	035610				50\$;OFFSET HAS BEEN ISSUED
7535	035540	116565	000123	000001		MOV	P.RCMD(R5),P.CMND(R5)	;GET LAST ISSUED COMMAND

7536	035546	013765	001640	000004
7537	035554	013765	001642	000002
7538	035562	013765	001634	000010
7539	035570	013765	001636	000012
7540	035576	005465	000012	
7541	035602	004037	051406	
7542	035606	001324		
7543	035610	000207		

```

MOV RTYSEC,P.SECT(R5) ;LOAD SECTOR AND TRACK ADDRESS
MOV RTYCYL,P.CYLN(R5) ;LOAD CYLINDER ADDRESS
MOV RTYBA,P.BALO(R5) ;LOAD BUS ADDRESS
MOV RTYWC,P.WC(R5) ;GET WORD COUNT
NEG P.WC(R5) ;MAKE IT NEGATIVE
JSR RC,Q.PUSH ;ENQUEUE PARAMETER BLOCK IN COMMAND
CINITQ ; INITIATION QUEUE
RTS PC ;RETURN
SOS:

```



```

7544 .SBTTL CALCULATE CYLINDER, TRACK, AND SECTOR FOR RETRY
7545
7546 035612 010146
7547 035614 010346
7548 035616 013703 001634
7549 035622 166503 000010
7550 035626 042703 000777
7551 035632 010337 001634
7552 035636 066537 000010 001634
7553 035644 016501 000012
7554 035650 005401
7555 035652 006203
7556 035654 160301
7557 035656 022701 000400
7558 035662 103002
7559 035664 012701 000400
7560 035670 010137 001636
7561 035674 116537 000005 001641
7562 035702 016537 000002 001642
7563 035710 010346
7564 035712 116616 000001
7565 035716 105066 000001
7566 035722 005046
7567 035724 116516 000004
7568 035730 062616
7569 035732 022716 000025
7570 035736 103005
7571 035740 162716 000026
7572 035744 105237 001641
7573 035750 000770
7574
7575 035752 112637 001640
7576 035756 005046
7577 035760 113716 001641
7578 035764 122716 000002
7579 035770 103005
7580 035772 162716 000003
7581 035776 005237 001642
7582 036002 000770
7583
7584 036004 112637 001641
7585 036010 112737 000001 001644
7586 036016 012603
7587 036020 012601
7588 036022 000207

CALSEC: MOV R1,-(SP) ; STORE R1 ON STACK
MOV R3,-(SP) ; STORE R3 ON STACK
MOV RTYBA,R3 ; GET ADDRESS OF ERROR WORD
SUB P.BALO(R5),R3 ; CALCULATE BYTES CORRECTLY TRANSFERRED
BIC #777,R3 ; MAKE IT ON SECTOR BOUNDARY
MOV R3,RTYBA ; STORE CALCULATED BUS ADDRESS
ADD P.BALO(R5),RTYBA ; CALCULATE START OF SECTOR WITH ERROR
MOV P.WC(R5),R1 ; GET INITIAL WORD COUNT
NEG R1 ; MAKE IT POSITIVE
ASR R3 ; DETERMINE WORDS BEFORE ERROR
SUB R3,R1 ; CALCULATE NUMBER OF WORDS LEFT
CMP #256.,R1 ; CHECK IF LESS THAN OR EQUAL TO ONE SECTOR
BHS 5$ ; LOAD RETRY WORD COUNT
MOV #256.,R1 ; NO, MAKE IT ONE SECTOR
MOV R1,RTYWC ; LOAD RETY WORD COUNT
MOV P.TRCK(R5),RTYTRK ; STORE RETRY TRACK
MOV P.CYLN(R5),RTYCYL ; STORE RETRY CYLINDER
MOV R3,-(SP) ; STORE WORDS TRANSFERRED
MOV8 1(SP),(SP) ; STORE NUMBER OF SECTORS
CLRB 1(SP) ; CLEAR HIGH ORDER BYTE
CLR -(SP) ; MAKE ROOM ON STACK
MOV8 P.SECT(R5),(SP) ; GET INITIAL SECTOR
ADD (SP)+,(SP) ; ADD SECTORS TRANSFERRED
CMP #21.,(SP) ; CHECK IF SECTOR COUNT GREATER THAN 21
BHS 10$ ; NO, CHECK TRACK
SUB #22.,(SP) ; SUBTRACT 22 SECTORS/TRACK
INCB RTYTRK ; INCREMENT TRACK
BR 10$ ; CHECK SECTOR COUNT

15$: MOV8 (SP)+,RTYSEC ; STORE RETRY SECTOR
CLR -(SP) ; MAKE ROOM ON THE STACK
MOV8 RTYTRK,(SP) ; STORE RETRY TRACK
CMPB #2,(SP) ; CHECK TRACK GREATER THAN 2
BHS 17$ ; NO, RETURN
SUB #3,(SP) ; SUBTRACT 3 TRACKS/CYLINDER
INC RTYCYL ; INCREMENT CYLINDER
BR 17$ ; CHECK TRACK COUNT

20$: MOV8 (SP)+,RTYTRK ; LOAD RETRY COUNT
MOV8 #1,RTYSCN ; INDICATE ONE SECTOR IN RETRY COUNT
MOV (SP)+,R3 ; RESTORE R3
MOV (SP)+,R1 ; RESTORE R1
RTS PC ; RETURN

```

M13

```

7584
7590
7591 036024 010046
7592 036026 010146
7593 036030 010316
7594 036032 010446
7595 036034 104401 061173
7596 036040 016503 000214
7597 036044 005046
7598 036046 116516 000217
7599 036052 006316
7600 036054 011601
7601 036056 062603
7602 036060 010346
7603 036062 004737 052774
7604 036066 122765 000001 000217
7605 036074 103403
7606 036076 104401 060515
7607 036102 000436
7608
7609 036104 016504 000214
7610 036110 016400 000002
7611 036114 042700 177400
7612 036120 005200
7613
7614 036122 126500 000217
7615 036126 101402
7616
7617 036130 005046
7618 036132 000414
7619
7620 036134 011404
7621 036136 042704 177760
7622 036142 006304
7623 036144 016404 004602
7624 036150 162701 000004
7625 036154 042701 177700
7626 036160 060104
7627 036162 011446
7628 036164 104401 060530
7629 036170 004737 052774
7630 036174 104401 060530
7631 036200 011346
7632 036202 004737 052774
7633 036206 104401 060530
7634 036212 005046
7635 036214 116516 000217
7636 036220 004737 052774
7637 036224 105765 000217
7638 036230 001003
7639 036232 104401 060515
7640 036236 000410
7641
7642 036240 104401 060530
7643 036244 017546 000214
7644 036250 042716 177760

.SBTTL PRINT FIRST ERROR WORD IN BUFFER
PRERWD: MOV R0,-(SP) ;STORE R0 ON STACK
MOV R1,-(SP) ;STORE R1 ON STACK
MOV R3,-(SP) ;STORE R3 ON STACK
MOV R4,-(SP) ;STORE R4 ON STACK
TYPE ERAD12 ;TYPE ERROR HEADER
MOV P.ERHD(R5),R3 ;GET ADDRESS OF START OF SECTOR
CLR -(SP) ;MAKE ROOM ON STACK
MOVB P.ERAD(R5),(SP) ;GET WORD IN ERROR OF SECTOR
ASL (SP) ;MULTIPLY IT BY 2
MOV (SP),R1 ;STORE IT IN R1
MOV (SP),R3 ;GET ADDRESS OF BAD WORD
MOV R3,-(SP) ;STORE ADDRESS FOR PRINT OUT
JSR PC,BINOC ;CONVERT TO OCTAL
CMPB #,P.ERAD(R5) ;CHECK IF IN DESCRIPTOR AREA
BLO 5$ ;NO, CALCULATE WORD
TYPE QUESMK ;TYPE QUESTION MARKS
BR 10$ ;GO TYPE WORD READ

5$: MOV P.ERHD(R5),R4 ;GET HEADER FOR PATTERN INDEX
MOV 2(R4),R0 ;GET NUMBER OF WORDS WRITTEN IN SECTOR
BIC #177400,R0 ;KEEP LOW BYTE
INC R0 ;INCREMENT R0 TO MAKE OFFSET FROM START
OF SECTOR
CMPB P.ERAD(R5),R0 ;CHECK IF IN ZERO FILL AREA
BLOS 7$ ;NO, CALCULATE ADDRESS OF
WORD IN DATA PATTERN
CLR -(SP) ;LOAD ZERO FOR PRINT OUT
BR 9$ ;GO PRINT EXPECTED CONTENTS

7$: MOV (R4),R4 ;GET DATA PATTERN NUMBER FOR INDEX
BIC #177760,R4 ;MASK OFF INSIGNIFICANT BITS
ASL R4 ;MULTIPLY BY 2
MOV PATTAB(R4),R4 ;GET PATTERN BASE
SUB #4,R1 ;SUBTRACT 4 FROM SECTOR OFFSET
BIC #177700,R1 ;MAKE IT MOD 32 WORDS
ADD R1,R4 ;CALCULATE PATTERN ADDRESS
MOV (R4),-(SP) ;STORE EXPECTED CONTENTS FOR PRINT OUT

9$: TYPE BLANKS ;TYPE BLANKS
JSR PC,BINOC ;CONVERT TO OCTAL
TYPE BLANKS ;TYPE BLANKS

10$: MOV (R3),-(SP) ;STORE WORD READ
JSR PC,BINOC ;CONVERT TO OCTAL
TYPE BLANKS ;TYPE BLANKS
CLR -(SP) ;MAKE ROOM ON STACK
MOVB P.ERAD(R5),(SP) ;STORE SECTOR WORD NUMBER FOR PRINT OUT
JSR PC,BINOC ;CONVERT TO OCTAL
TSTB P.ERAD(R5) ;CHECK IF ERROR IS IN PATTERN WORD
BNE 15$ ;NO, PRINT PATTERN NUMBER
TYPE QUESMK ;TYPE QUESTION MARKS
BR 20$ ;GO RESTORE REGISTERS

15$: TYPE BLANKS ;TYPE BLANKS
MOV P.ERHD(R5),-(SP) ;GET FIRST WORD OF SECTOR READ
BIC #177760,(SP) ;CLEAR INSIGNIFICANT BITS

```

7645	036254	004737	052774
7646	036260	104401	001165
7647	036264	012604	
7648	036266	012603	
7649	036270	012601	
7650	036272	012600	
7651	036274	000207	

20S:	JSR	PC,BINOCT	:CONVERT TO OCTAL
	TYPE	,SCLF	:TYPE <CR><LF>
	MOV	(SP)+,R4	:RESTORE R4
	MOV	(SP)+,R3	:RESTORE R3
	MOV	(SP)+,R1	:RESTORE R1
	MOV	(SP)+,R0	:RESTORE R0
	RTS	PC	:RETURN

```

7652          .SBTTL  SUCCESSFUL RECOVERY CLEAN UP
7653
7654 036276 032765 000001 000150 SURECY: BIT      #BIT0,P.DSTT(R5) ;CHECK IF DATA TYPE ERROR
7655 036304 001466          BEQ      10$          ;NO CLEAN UP FOR NEXT COMMAND
7656 036306 105765 000147          TSTB   P.RERD(R5)    ;CHECK IF FIRST ERROR
7657 036312 001003          BNE     $          ;NO CHECK IF OFFSET
7658 036314 005265 000176          INC     P.SECC(R5)  ;INCREMENT NO. OF ECC RECOVERABLE ERRORS
7659 036320 000426          BR      9$          ;GO CLEAN UP FOR NEXT COMMAND
7660
7661 036322 122765 000020 000147 5$:  CMPB   #20,P.RERD(R5) ;CHECK IF OFFSET
7662 036330 103403          BLO     8$          ;YES INCREMENT OFFSET RECOVERABLE
7663 036332 005265 000172          INC     P.SRRD(R5)  ;INCREMENT REREAD RECOVERABLE
7664 036336 000417          BR      9$          ;GO CLEAN UP FOR NEXT COMMAND
7665
7666 036340 005265 000174          INC     P.SOFF(R5)  ;INCREMENT OFFSET RECOVERABLE
7667 036344 032777 020000 142566 8$:  BIT     #SW13,2SWR    ;CHECK IF INHIBIT PRINT OUT
7668 036352 001011          BNE     9$          ;YES CLEAN UP FOR NEXT COMMAND
7669 036354 104401 063552          TYPE   ,ERR10B     ;TYPE "OFFSET"
7670 036360 005046          CLR     -(SP)       ;MAKE ROOM ON STACK
7671 036362 116516 000006          MOVB   P.OFFST(R5),(SP) ;STORE OFFSET FOR PRINT OUT
7672 036366 004737 052774          JSR    PC,BINOC    ;CONVERT TO OCTAL
7673 036372 104401 001165          TYPE   ,SCLF       ;TYPE <CR><LF>
7674 036376 105037 001644          CLRB   ATYSCN      ;CLEAR SECTOR RETRY FLAG
7675 036402 016546 000176          MOV     P.SECC(R5),-(SP) ;STORE NUMBER OF ECC RECOVERABLE ERRORS
7676 036406 066516 000172          ADD    P.SRRD(R5),(SP) ;ADD NUMBER OF REREAD RECOVERABLE ERRORS
7677 036412 066516 000174          ADD    P.SOFF(R5),(SP) ;ADD NUMBER OF OFFSET RECOVERABLE ERRORS
7678 036416 022665 000100          CMP    (SP)+,P.CERT(R5) ;CHECK IF SOFT ERROR RATE EXCEEDED
7679 036422 101417          BLOS   10$         ;NO GO ISSUE NEXT COMMAND
7680 036424 032777 040020 142506 10$: BIT     #SW4!SW14,2SWR ;CHECK IF INHIBIT AUTOMATIC DRIVE DROPPING
7681 036432 001013          BNE     10$        ;YES GO ISSUE NEXT COMMAND
7682 036434 004737 022642          JSR    PC,BUFREL   ;GO RELEASE BUFFER
7683 036440 104401 064047          TYPE   ,ERR20S    ;TYPE "ERROR THRESHOLD EXCEEDED"
7684 036444 004737 013004          JSR    PC,DROP     ;GO DROP DRIVE
7685 036450 005037 001566          CLR    ERAPRO      ;CLEAR ERROR CURRENTLY BEING PROCESSED
7686 036454 004737 022342          JSR    PC,GETBUF   ;GET BUFFER FOR NEXT COMMAND
7687 036460 000207          RTS              ;RETURN
7688
7689 036462 032777 020000 142450 10$: BIT     #SW13,2SWR    ;CHECK IF INHIBIT PRINT OUT
7690 036470 001002          BNE     15$        ;YES CHECK IF READ PACK SERIAL NUMBER
7691 036472 104401 063757          TYPE   ,ERR203    ;TYPE RECOVERABLE ERROR
7692 036476 105037 001644          CLRB   ATYSCN      ;CLEAR SECTOR RETRY COUNT
7693 036502 005065 000146          CLR    P.RECT(R5)  ;CLEAR ERROR FLAGS AND RETRY COUNTS
7694 036506 005065 000150          CLR    P.DSTT(R5)
7695 036512 005065 000212          CLR    P.ERR(R5)
7696 036516 005037 001566          CLR    ERAPRO      ;CLEAR ERROR CURRENTLY BEING PROCESSED
7697 036522 010446          MOV     R4, -(SP)  ;STORE R4 ON STACK
7698 036524 116504 000000          MOVB   P.DRVN(R5),R4 ;STORE DRIVE NUMBER FOR INDEX
7699 036530 156437 001514 001513 15$: BISB   INTMSK(R4),0,OVER ;ALLOW OVERLAPPED OPERATIONS ON THIS DRIVE
7700 036536 012604          MOV     (SP)+,R4   ;RESTORE R4
7701 036540 105765 000210          TSTB   P.ASSN(R5)  ;CHECK IF IN ASSIGNMENT SEQUENCE
7702 036544 001025          BNE     20$        ;YES TERMINATE SEQUENCE
7703 036546 122765 000117 000001 20$: CMPB   #SEEK,P.CMND(R5) ;CHECK IF SEEK TO PREVIOUS CYLINDER
7704 036554 001410          BEQ    16$        ;YES GO ISSUE RELEASE
7705 036556 122765 000140 000001 16$: CMPB   #RELEAS,P.CMND(R5) ;CHECK IF RELEASE
7706 036564 001013          BNE     17$        ;NO CARRY OUT NORMAL RECOVERY
7707 036566 004037 051406          JSR    RO,Q.PUSH   ;ENQUEUE PARAMETER BLOCK IN DRIVE

```

```

7708 036572 001314 AVAILQ ; AVAILABLE QUEUE
7709 036574 000207 RTS PC ; RETURN
7710
7711 036576 112765 000140 000001 16$: MOVB #RELEAS.P.CMND(R5) ; ISSUE RELEASE COMMAND
7712 036604 004037 051406 JSR RD,Q.PUSH ; ENQUEUE PARAMETER BLOCK IN COMMAND
7713 036610 001324 CINITQ ; INITIATION QUEUE
7714 036612 000207 RTS PC ; RETURN
7715
7716 036614 000137 024616 17$: JMP NORM1 ; CARRY OUT NORMAL DATA TRANSFER TERMINATION
7717
7718 : CHECK IF ANY OF THE FOLLOWING ASSIGNMENT STAGES
7719 :
7720 : READ DRIVE STATUS
7721 : START SPINOLE
7722 : SET VOLUME VALID
7723 : RECAL
7724 036620 132765 000117 000210 20$: BITB #BIT0:BIT1:BIT2:BIT3:BIT6,P.ASSN(R5)
7725 036626 001402 BEQ 25$ ; NO PROCESS WRITE PACK
7726 036630 000137 013504 JMP ASNORM ; PROCESS NORMAL TERMINATION
7727
7728 036634 032765 002000 000014 25$: BIT #DRPDRV,P.PRST(R5) ; CHECK IF DROP DRIVE
7729 036642 001030 BNE 30$ ; YES GO DROP DRIVE
7730 036644 105065 000211 CLR B ; CLEAR SEEK TO PREVIOUS LOCATION
7731 036650 116565 00023 000001 MOVB P.RCMD(R5),P.CMND(R5) ; REISSUE LAST WRITE/WRITE CHECK COMMAND
7732 036656 052765 000200 000014 BIS #W.WCK,P.PRST(R5) ; SET ISSUE WRITE BEFORE WRITE CHECK
7733 036664 016565 000126 000004 MOV P.RSEC(R5),P.SECT(R5) ; LOAD PREVIOUS TRACK AND SECTOR
7734 036672 016565 000124 000002 MOV P.RCYL(R5),P.CYLN(R5) ; LOAD PREVIOUS CYLINDER
7735 036700 016565 000132 000010 MOV P.RBAL(R5),P.BALO(R5) ; LOAD PREVIOUS BUS ADDRESS
7736 036706 016565 000130 000012 MOV P.RWC(R5),P.WC(R5) ; LOAD PREVIOUS WORD COUNT
7737 036714 004037 051406 JSR RD,Q.PUSH ; ENQUEUE PARAMETER BLOCK IN COMMAND
7738 036720 001324 CINITQ ; INITIATION QUEUE
7739 036722 000207 RTS PC ; RETURN
7740
7741 036724 004737 022642 30$: JSR PC,BUFREL ; GO RELEASE BUFFER
7742 036730 104401 056260 TYPE ,OPROB ; TYPE "OPERATOR INITIATED DROP DRIVE
7743 : DURING PACK WRITING"
7744 036734 004737 013004 JSR PC,DROP ; GO DROP DRIVE
7745 036740 004737 022342 JSR PC,GETBUF ; GET BUFFER FOR NEXT COMMAND
7746 036744 000207 RTS PC ; RETURN

```

```

7747
7748
7749 036746 010446
7750 036750 010346
7751 036752 010504
7752 036754 012703 004516
7753
7754 036760 012423
7755 036762 022703 004602
7756 036766 001374
7757 036770 012603
7758 036772 012604
7759 036774 000207
7760
7761
7762
7763 036776 010446
7764 037000 010346
7765 037002 010504
7766 037004 012703 004516
7767
7768 037010 012324
7769 037012 022703 004602
7770 037016 001374
7771 037020 012603
7772 037022 012604
7773 037024 000207

.SBTTL GET CURRENT STATUS INFORMATION
GETCUR: MOV R4, -(SP) ; STORE R4 ON STACK
        MOV R3, -(SP) ; STORE R3 ON STACK
        MOV R5, R4 ; LOAD START OF CURRENT PARRAMETER BLOCK
        MOV #PARM10, R3 ; LOAD START OF RETRY PARAMETER BLOCK

SS:     MOV (R4)+, (R3)+ ; COPY PARAMTER BLOCK
        CMP #PARM10+P.EPAT+2, R3 ; CHECK IF FINISHED
        BNE SS ; NO, GET NEXT WORD
        MOV (SP)+, R3 ; RESTORE R3
        MOV (SP)+, R4 ; RESTORE R4
        RTS PC ; RETURN

.SBTTL RESTORE STATUS
RSTAT:  MOV R4, -(SP) ; STORE R4 ON STACK
        MOV R3, -(SP) ; STORE R3 ON STACK
        MOV R5, R4 ; LOAD START OF CURRENT PARRAMETER BLOCK
        MOV #PARM10, R3 ; LOAD START OF RETRY PARAMETER BLOCK

SS:     MOV (R3)+, (R4)+ ; COPY PARAMTER BLOCK
        CMP #PARM10+P.EPAT+2, R3 ; CHECK IF FINISHED
        BNE SS ; NO, GET NEXT WORD
        MOV (SP)+, R3 ; RESTORE R3
        MOV (SP)+, R4 ; RESTORE R4
        RTS PC ; RETURN

```

```

7774 .SBTTL DETERMINE OFFSET FOR RETRY
7775
7776 037026 122765 000021 000147 DETOFF: CMPB #21,P.RERD(R5) ;CHECK IF OFFSET = +400 MICRO-INCHES
7777 037034 001004 BNE 10$ ;NO, CONTINUE
7778 037036 112765 000020 000006 MOVB #OFFP4,P.OFST(R5) ;LOAD OFFSET VALUE = +400 MICRO-INCHES
7779 037044 000447 BR 15$ ;ISSUE OFFSET
7780
7781 037046 122765 000024 000147 10$: CMPB #24,P.RERD(R5) ;CHECK IF OFFSET = -400 MICRO-INCHES
7782 037054 001004 BNE 11$ ;NO, CONTINUE
7783 037056 112765 000220 000006 MOVB #OFFM4,P.OFST(R5) ;LOAD OFFSET VALUE = -400 MICRO-INCHES
7784 037064 000437 BR 15$ ;ISSUE OFFSET
7785
7786 037066 122765 000027 000147 11$: CMPB #27,P.RERD(R5) ;CHECK IF OFFSET = +800 MICRO-INCHES
7787 037074 001004 BNE 12$ ;NO, CONTINUE
7788 037076 112765 000040 000006 MOVB #OFFP8,P.OFST(R5) ;LOAD OFFSET VALUE = +800 MICRO-INCHES
7789 037104 000427 BR 15$ ;ISSUE OFFSET
7790
7791 037106 122765 000032 000147 12$: CMPB #32,P.RERD(R5) ;CHECK IF OFFSET = -800 MICRO-INCHES
7792 037114 001004 BNE 13$ ;NO, CONTINUE
7793 037116 112765 000240 000006 MOVB #OFFM8,P.OFST(R5) ;LOAD OFFSET VALUE = -800 MICRO-INCHES
7794 037124 000417 BR 15$ ;ISSUE OFFSET
7795
7796 037126 122765 000035 000147 13$: CMPB #35,P.RERD(R5) ;CHECK IF OFFSET = +1200 MICRO-INCHES
7797 037134 001004 BNE 14$ ;NO, CONTINUE
7798 037136 112765 000060 000006 MOVB #OFFP12,P.OFST(R5) ;LOAD OFFSET VALUE = +1200 MICRO-INCHES
7799 037144 000407 BR 15$ ;ISSUE OFFSET
7800
7801 037146 122765 000040 000147 14$: CMPB #40,P.RERD(R5) ;CHECK IF OFFSET = -1200 MICRO-INCHES
7802 037154 001013 BNE 30$ ;NO, DO REREAD
7803 037156 112765 000260 000006 MOVB #OFFM12,P.OFST(R5) ;LOAD OFFSET VALUE = -1200 MICRO-INCHES
7804 037164 112765 000115 000001 15$: MOVB #OFFSET,P.CMND(R5) ;LOAD OFFSET COMMAND
7805 037172 004037 051406 JSR RO,Q.PUSH ;ENQUEUE PARAMETER BLOCK IN COMMAND
7806 037176 001324 CINITQ ; INITIATION QUEUE
7807 037200 011000 MOV (RO),RO ;LOAD OFFSET ISSUED RETURN
7808 037202 000200 RTS RO ;RETURN
7809
7810 037204 005720 30$: TST (RO)+ ;ADJUST RETURN FOR NO OFFSET
7811 037206 000200 RTS RO ;RETURN

```

```

7812
7813
7814 037210 013746 001642
7815 037214 166516 000002
7816 037220 011646
7817 037222 006316
7818 037224 062616
7819 037226 005046
7820 037230 113716 001641
7821 037234 062616
7822 037236 005046
7823 037240 116516 000005
7824 037244 162616
7825 037246 012746 000026
7826 037252 004737 054714
7827 037256 012616
7828 037260 005046
7829 037262 113716 001640
7830 037266 062616
7831 037270 005046
7832 037272 116516 000004
7833 037276 162616
7834 037300 111666 000001
7835 037304 105016
7836 037306 161637 001636
7837 037312 006316
7838 037314 062637 001634
7839 037320 000207
    
```

.SBTTL CALCULATE RETRY WORD COUNT

```

CALWC:  MOV    RTYCYL, -(SP)      ;STORE RETRY CYLINDER
        SUB    P.CYLN(R5), (SP)  ;SUBTRACT INITIAL CYLINDER
        MOV    (SP), -(SP)      ;STORE CYLINDER DIFFERENCE
        ASL    (SP)              ;MULTIPLY DIFFERENCE BY 2
        ADD    (SP)+, (SP)      ;MULTIPLY DIFFERENCE BY 3
        CLR    -(SP)            ;MAKE ROOM ON STACK
        MOVB   RTYTRK, (SP)      ;GET RETRY TRACK
        ADD    (SP)+, (SP)      ;ADD RETRY TRACK
        CLR    -(SP)            ;MAKE ROOM ON THE STACK
        MOVB   P.TRCK(R5), (SP)  ;GET INITIAL TRACK
        SUB    (SP)+, (SP)      ;SUBTRACT INITIAL TRACK
        MOV    #22, -(SP)       ;MULTIPLY TRACK BY 22
        JSR    PC, $MULT        ;
        MOV    (SP)+, (SP)      ;THROW AWAY MOST SIGNIFICANT BITS
        CLR    -(SP)            ;MAKE ROOM ON THE STACK
        MOVB   RTYSEC, (SP)      ;GET RETRY SECTOR
        ADD    (SP)+, (SP)      ;ADD PRESENT SECTOR
        CLR    -(SP)            ;MAKE ROOM ON STACK
        MOVB   P.SECT(R5), (SP)  ;GET INITIAL SECTOR
        SUB    (SP)+, (SP)      ;CALCULATE NUMBER OF SECTORS UNTIL ERROR
        MOVB   (SP), 1(SP)      ;MULTIPLY BY 256 WORDS/SECTOR
        CLRB   (SP)
        SUB    (SP), RTYWC      ;SUBTRACT FROM INITIAL WORD COUNT
        ASL    (SP)              ;DETERMINE NUMBER OF BYTES
        ADD    (SP)+, RTYBA     ;CALCULATE STARTING BUFFER ADDRESS
        RTS    PC              ;RETURN
    
```



```

7840 .SBTTL RECOVERABLE CONTROLLER ERROR TERMINATION
7841
7842 037322 012037 037344 CONERR: MOV (R0)+ 2$ ;STORE ERROR MESSAGE ADDRESS
7843 037326 004737 030000 JSR PC,PR1000 ;PRINTER HEADER
7844 037332 032777 020000 141600 BIT #SW13,2SWR ;CHECK IF INHIBIT PRINT OUT
7845 037340 001032 BNE 10$ ;YES, CHECK IF HALT ON ERROR
7846 037342 104401 TYPE ;TYPE ERROR MESSAGE
7847 037344 000000 2$: .WORD 0
7848 037346 104401 001165 TYPE $CRLF ;TYPE <CR><LF>
7849 037352 004737 026074 JSR PC,PR1000 ;PRINT CONTROLLER REGISTERS
7850 037356 004737 030244 JSR PC,PR1002 ;PRINT CURRENT EXECUTED COMMAND
7851 037362 022737 063130 037344 CMP #ERR066,2$ ;CHECK IF ECC LOGIC ERROR
7852 037370 001016 BNE 10$ ;NO, CHECK IF HALT ON ERROR
7853 037372 104401 063150 TYPE ERR067 ;TYPE "RKECPT RKECPS"
7854 037376 016546 000062 MOV P.EPAT(R5),-(SP) ;STORE ECC PATTERN FOR PRINT OUT
7855 037402 004737 052774 JSR PC,BINOC7 ;CONVERT TO OCTAL
7856 037406 104401 060530 TYPE BLANKS ;TYPE 2 SPACES
7857 037412 016546 000060 MOV P.EPOS(R5),-(SP) ;STORE ECC POSITION FOR PRINT OUT
7858 037416 004737 052774 JSR PC,BINOC7 ;CONVERT TO OCTAL
7859 037422 104401 001165 TYPE $CRLF ;TYPE <CR><LF>
7860 037426 004737 031250 10$: JSR PC,HLT000 ;CHECK IF HALT ON ERROR
7861 037432 005237 001630 INC CNTCNT ;INCREMENT CONTROLLER ERROR COUNT
7862 037436 022737 000012 001630 CMP #T.CONT,CNTCNT ;CHECK IF THRESHOLD EXCEEDED
7863 037444 103006 BHS 15$ ;NO, CONTINUE
7864 037446 104401 060533 TYPE ,ERR000 ;TYPE "FATAL ERROR"
7865 037452 104401 061072 TYPE ,ERR008 ;TYPE "CONTROLLER ERROR THRESHOLD EXCEEDED"
7866 037456 000000 HALT
7867 037460 000776 BR .-2
7868
7869 037462 105765 000150 15$: TSTB P.DSTT(R5) ;CHECK IF FIRST ERROR
7870 037466 100005 BPL 20$ ;NO, REQUEUE COMMAND
7871 037470 042765 100200 000150 BIC #BIT15:BIT7,P.DSTT(R5) ;CLEAR ERROR ENQUEUED AND FIRST ERROR
7872 037476 005037 001566 CLR ERRPRO ;CLEAR ERROR BEING PROCESSED
7873 037502 004037 051406 20$: JSR RO,Q.PUSH ;REQUEUE COMMAND IN COMMAND
7874 037506 001324 CINITQ ;INITIATION QUEUE
7875 037510 000200 RTS RO ;RETURN

```

```

7876 .SBTTL TEST FOR DROP DRIVE ERROR
7877
7878 037512 032765 000030 000036 TSTDRP: BIT #ACLO!SPDLSS,P.DS(R5) ;CHECK IF AC LOW OR SPEED LOSS
7879 037520 001016 BNE 10$ ;YES REPORT ERROR
7880 : CHECK IF ANY OF THE FOLLOWING ERRORS ARE SET
7881 : UNSAFE
7882 : ILLEGAL DISK ADDRESS
7883 : WRITE LOCK ERROR
7884 : DRIVE TYPE ERROR
7885 : FORMAT ERROR
7886 : NON-EXECUTABLE DRIVE FUNCTION
7887 037522 032765 046064 000034 BIT #UNS!IDAE!WLE!DTYE!FMTE!HXF,P.ER(R5)
7888 BNE 12$ ;YES, REPORT ERROR
7889 037530 001026 CHECK IF ANY OF THE FOLLOWING FLAGS ARE SET
7890 : DRIVE ERROR NOT REPORTED
7891 : DRIVE READY NOT SET AFTER START SPINDLE
7892 : VOLUME VALID NOT SET AFTER PACK ACKNOWLEDGE
7893 BIT #BIT3!BIT10!BIT11,P.ERR(R5)
7894 037532 032765 006010 000212
7895 BNE 18$ ;YES, GC REPORT ERROR
7896 037540 001072
7897 : CHECK IF ANY OF THE FOLLOWING FLAGS ARE SET
7898 : DRIVE HARD ERROR
7899 : DRIVE STATUS CHANGE DID NOT CLEAR
7900 : UNEXPECTED ATTN FROM DRIVE COMMAND
7901 : ATTENTION SET BUT DSC AND FAULT RESET
7902 BIT #DRVHRD!DRVDSC!UEXATT!NODSC,P.PRST(R5)
7903 037542 032765 004070 000014
7904 BNE 32$ ;YES REPORT ERROR
7905 037550 001116 TST (R0)+ ;GET NORMAL RETURN ADDRESS
7906 037552 005720 RTS R0 ;RETURN
7907 037554 000200
7908
7909 037556 004737 030000 10$: JSR PC,PRIO00 ;PRINT ERROR HEADER
7910 037562 032765 000010 000036 BIT #ACLO,P.DS(R5) ;CHECK IF AC LOW
7911 037570 001403 BEQ 11$ ;NO, CONTINUE
7912 037572 104401 062103 TYPE ERR033 ;TYPE "AC LOW"
7913 037576 000472 BR 30$ ;GO GET ERROR QUALIFICATION
7914
7915 037600 104401 062112 11$: TYPE ERR034 ;TYPE "SPEED LOSS"
7916 037604 000467 BR 30$ ;GET ERROR QUALIFICATION
7917
7918 037606 004737 030000 12$: JSR PC,PRIO00 ;PRINT ERROR HEADER
7919 037612 032765 040000 000034 BIT #UYS,P.ER(R5) ;CHECK IF DRIVE UNSAFE
7920 037620 001403 BEQ 13$ ;NO, CONTINUE
7921 037622 104401 062622 TYPE ERRO56 ;TYPE "DRIVE UNSAFE"
7922 037626 000456 BR 30$ ;GET ERROR QUALIFICATION
7923
7924 037630 032765 000040 000034 13$: BIT #DTYE,P.ER(R5) ;CHECK IF DRIVE TYPE ERROR
7925 037636 001403 GEQ 14$ ;NO, CONTINUE
7926 037640 104401 062125 TYPE ERRO35 ;TYPE "DRIVE TYPE ERROR"
7927 037644 000447 BR 30$ ;GET ERROR QUALIFIER BITS
7928
7929 037646 032765 000020 000034 14$: BIT #FMTE,P.ER(R5) ;CHECK IF FORMAT ERROR
7930 037654 001403 BEQ 15$ ;NO, CONTINUE
7931 037656 104401 062142 TYPE .ERR036 ;TYPE FORMAT ERROR

```

7932	037662	000440				BR	30\$:GET ERROR QUALIFIER BITS
7933									
7934	037664	032765	000004	000034	15\$:	BIT	#NXF,P.ER(R5)		:CHECK IF NON-EXECUTABLE DRIVE FUNCTION
7935	037672	001403				BEG	16\$:NO CONTINUE
7936	037674	104401	062155			TYPE	ERR037		:TYPE ILLEGAL DRIVE FUNCTION
7937	037700	000431				BR	30\$:GET ERROR QUALIFIER BITS
7938									
7939	037702	032765	002000	000034	16\$:	BIT	#IDAE,P.ER(R5)		:CHECK IF ILLEGAL DISK ADDRESS
7940	037710	001403				BEG	17\$:NO CONTINUE
7941	037712	104401	062052			TYPE	ERR031		:TYPE "ILLEGAL DISK ADDRESS"
7942	037716	000422				BR	30\$:GET ERROR QUALIFIER BITS
7943									
7944	037720	104401	062066		17\$:	TYPE	ERR032		:TYPE "WRITE LOCK ERROR"
7945	037724	000417				BR	30\$:GET ERROR QUALIFIER BITS
7946									
7947	037726	004737	030000		18\$:	JSR	PC,PR1000		:PRINT ERROR HEADER
7948	037732	032765	002000	000212		BIT	#BIT10,P.ERR(R5)		:CHECK IF DRIVE NOT READY AFTER START SPINDLE
7949	037740	001403				BEG	19\$:NO CONTINUE
7950	037742	104401	062514			TYPE	ERR053+1		:PRINT "DRIVE NOT READY AFTER START SPINDLE"
7951	037746	000406				BR	30\$:GET ERROR QUALIFIER BITS
7952									
7953	037750	032765	004000	000212	19\$:	BIT	#BIT11,P.ERR(R5)		:CHECK IF VOLUME VALID DID NOT SET
7954									:AFTER PACK ACKNOWLEDGE
7955	037756	001404				BEG	31\$:NO CONTINUE
7956	037760	104401	062555			TYPE	ERR054+1		:PRINT "VOLUME VAID NOT SET AFTE ?
7957									:PACK ACKNOWLEDGE INDSTRUCTION"
7958	037764	104401	001165		30\$:	TYPE	SCRLF		:TYPE <CR><LF>
7959	037770	032765	000010	000212	31\$:	BIT	#BIT3,P.ERR(R5)		:CHECK IF ERROR NOT INDICATED BY FAULT
7960	037776	001405				BEG	42\$:NO CHECK OTHER QUALIFIER BITS
7961	040000	104401	062635			TYPE	ERR057		:TYPE "ERROR NOT INDICATED BY FAULT"
7962	040004	000402				BR	42\$:CHECK OTHER QUALIFIER BITS
7963									
7964	040006	004737	030000		32\$:	JSR	PC,PR1000		:PRINT ERROR HEADER
7965	040012	032765	000020	000014	42\$:	BIT	#DRVHRD,P.PRST(R5)		:CHECK IF HARD DRIVE ERROR
7966	040020	001402				BEG	43\$:NO CHECK OTHER QUALIFIER BITS
7967	040022	104401	061723			TYPE	ERR027		:TYPE "DRIVE HARD ERROR"
7968	040026	032765	004000	000014	43\$:	BIT	#NODSC,P.PRST(R5)		:CHECK IF ATTENTION BUT
7969									:NO DRIVE STATUS CHANGE OR FAULT
7970	040034	001402				BEG	44\$:NO CHECK OTHER QUALIFIER BITS
7971	040036	104401	062676			TYPE	ERR058		:TYPE "ATTN BUT NO DSC OR FAULT"
7972	040042	032765	000040	000014	44\$:	BIT	#DRVDSC,P.PRST(R5)		:CHECK IF DRIVE STATUS CHANGE DID
7973									:NOT CLEAR
7974	040050	001402				BEG	45\$:NO CHECK OTHER ERROR QUALIFIER BITS
7975	040052	104401	061742			TYPE	ERR028		:TYPE "DRIVE STATUS CHANGE DID NOT CLEAR"
7976	040056	032765	000010	000014	45\$:	BIT	#UEXATT,P.PRST(R5)		:CHECK IF UNEXPECTED ATTENTION FROM
7977									:DRIVE COMMAND
7978	040064	001402				BEG	46\$:NO CHECK OTHER QUALIFIER BITS
7979	040066	104401	062032			TYPE	ERR030		:TYPE "UNEXPECTED ATTN FROM DRIVE COMMAND"
7980	040072	005265	000206		46\$:	INC	P.NER(R5)		:INCREMENT OTHER ERROR COUNT
7981	040076	032777	020000	141034		BIT	#SW13,DSWR		:CHECK IF INHIBIT PRINT OUT
7982	040104	001015				BNE	60\$:YES, CHECK IF HALT ON ERROR
7983	040106	005765	000150			TST	P.DSTT(R5)		:CHECK IF ERROR DURING ERROR RECOVERY
7984	040112	100402				BMI	51\$:NO PRINT STATUS
7985	040114	104401	063721			TYPE	ERR202		:TYPE "ERROR DURING ERROR RECOVERY"
7986	040120	004737	026074		51\$:	JSR	PC,PR1000		:PRINT CONTROLLER REGISTERS
7987	040124	004737	025720			JSR	PC,PRDSTT		:PRINT DRIVE STATUS

7988	040130	004737	030244		JSR	PC,PR1002	;PRINT CURRENT COMMAND	
7989	040134	004737	030066		JSR	PC,PR1001	;PRINT PREVIOUS COMMAND	
7990	040140	004737	031250		JSR	PC,HLTOCO	;CHECK IF HALT ON ERROR	
7991	040144	032777	000100	140766	60\$:	BIT	#SW6,#SWR	;CHECK FOR SWITCH 6
7992	040152	001431			BEQ	63\$;NO DROP DRIVE	
7993	040154	032765	000030	000036	BIT	#ACLO!SPDLSS,P.D5(R5)	;CHECK IF UNSAFE, ACLOW AND SPEED LOSS	
7994	040162	001004			BNE	61\$		
7995	040164	032765	040000	000034	BIT	#UNS,P.ER(R5)		
7996	040172	001421			BEQ	63\$		
7997	040174	105765	000120		61\$:	TSTB	P.IERC(R5)	;CHECK IF INHIBIT ERROR RECOVERY
7998	040200	001012			BNE	62\$		
7999	040202	105265	000146		INCB	P.RECT(R5)	;INCREMENT ERROR RETRY COUNT	
8000	040206	122765	000010	000146	CMPB	#8,P.RECT(R5)	;CHECK IF RETRY THRESHOLD EXCEEDED	
8001	040214	103404			BLO	62\$;YES, INDICATE UNSUCCESSFUL RECOVERY	
8002	040216	004037	051406		JSR	RO,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN COMMAND	
8003	040222	001324			CINITQ		;INITIATION QUEUE	
8004	040224	000402			BR	57\$;RETURN	
8005								
8006	040226	004737	031502		62\$:	JSR	PC,UNRECY	;INDICATE UNSUCCESSFUL RECOVERY
8007	040232	011000			57\$:	MOV	(RO),RO	
8008	040234	000200			RTS	RO		
8009								
8010	040236	004737	022642		63\$:	JSR	PC,BUFREL	;RELEASE BUFFER
8011	040242	004737	013004		JSR	PC,DROP	;DROP DRIVE	
8012	040246	004737	022342		JSR	PC,GETBUF	;GET NEW BUFFER	
8013	040252	005037	001566		CLR	ERRPRO	;CLEAR ERROR BEING PROCESSED	
8014	040256	011000			MOV	(RO),RO	;LOAD ERROR RETURN	
8015	040260	000200			RTS	RO	;RETURN	

```

.SBTTL ERROR RECOVERY SEQUENCE NORMAL RETURN
8016
8017
8018 040262 032765 004000 000150 ERCVY1: BIT #BIT11,P.DSTT(R5) ;CHECK IF RECALIBRATE ISSUED
8019 040270 00415 BEQ EC3$ ;NO, CONTINUE
8020 040272 042765 004000 000150 BIC #BIT11,P.DSTT(R5) ;CLEAR RECALIBRATE ISSUED
8021 040300 052765 010000 000150 15$: BIS #BIT12,P.DSTT(R5) ;SET SEEK ISSUED
8022 040306 112765 000117 000001 MOVB #SEEK,P.CMND(R5) ;ISSUE SEEK INSTRUCTION
8023 040314 004037 051406 JSR RD,Q.PUSH ;PUT PARAMETER BLOCK IN COMMAND
8024 040320 001324 CINITQ ;INITIATION QUEUE
8025 040322 000207 RTS PC ;RETURN
8026
8027 040324 032765 010000 000150 EC3$: BIT #BIT12,P.DSTT(R5) ;CHECK IF SEEK ISSUED
8028 040332 001427 BEQ 10$ ;NO, CONTINUE
8029 040334 042765 010000 000150 BIC #BIT12,P.DSTT(R5) ;CLEAR SEEK ISSUED
8030 040342 105765 000211 TSTB P.SEEK(R5) ;CHECK IF SEEK TO PREVIOUS POSITION
8031 040346 001403 BEQ 5$ ;NO, PROCESS DATA TRANSFER
8032 040350 004737 036276 JSR PC,SURECY ;INDICATE SUCCESSFUL RECOVERY
8033 040354 000207 RTS PC ;RETURN
8034
8035 040356 122765 000021 000147 5$: CMPB #21,P.RERD(R5) ;CHECK IF OFFSET
8036 040364 101053 BHI 20$ ;NO, REISSUE COMMAND
8037 040366 052765 020000 000150 BIS #BIT13,P.DSTT(R5) ;SET OFFSET ISSUED
8038 040374 112765 000115 000001 MOVB #OFFSET,P.CMND(R5) ;ISSUE OFFSET
8039 040402 004037 051406 JSR RD,Q.PUSH ;PUT PARAMETER BLOCK IN COMMAND
8040 040406 001324 CINITQ ;INITIATION QUEUE
8041 040410 000207 RTS PC ;RETURN
8042
8043 040412 032765 020000 000150 10$: BIT #BIT13,P.DSTT(R5) ;CHECK IF DO PREVIOUS OFFSET
8044 040420 001453 BEQ EC4$ ;NO, CONTINUE
8045 040422 042765 020000 000150 BIC #BIT13,P.DSTT(R5) ;CLEAR OFFSET ISSUED
8046 040430 122765 000147 000147 CMPB #21,P.RERD(R5) ;CHECK IF OFFSET +400 MICRO-INCHES
8047 040436 001424 BEQ 15$ ;YES, GO ISSUE READ
8048 040440 122765 000024 000147 CMPB #24,P.RERD(R5) ;CHECK IF OFFSET -400 MICRO-INCHES
8049 040446 001420 BEQ 15$ ;YES, GO ISSUE READ
8050 040450 122765 000027 000147 CMPB #27,P.RERD(R5) ;CHECK IF OFFSET +800 MICRO-INCHES
8051 040456 001414 BEQ 15$ ;YES, GO ISSUE READ
8052 040460 122765 000032 000147 CMPB #32,P.RERD(R5) ;CHECK IF OFFSET -800 MICRO-INCHES
8053 040466 001410 BEQ 15$ ;YES, GO ISSUE READ
8054 040470 122765 000035 000147 CMPB #35,P.RERD(R5) ;CHECK IF OFFSET +1200 MICRO-INCHES
8055 040476 001404 BEQ 15$ ;YES, GO ISSUE READ
8056 040500 122765 000040 000147 CMPB #40,P.RERD(R5) ;CHECK IF OFFSET -1200 MICRO-INCHES
8057 040506 001002 BNE 20$ ;NO, DO NOT INCREMENT REREAD COUNT
8058 040510 105265 000147 15$: INCB P.RERD(R5) ;INCREMENT REREAD COUNT
8059 040514 116565 000123 000001 20$: MOVB P.RCMD(R5),P.CMND(R5) ;LOAD COMMAND
8060 040522 122765 000131 000001 CMPB #WRTCHK,P.CMND(R5) ;CHECK IF WRITE CHECK
8061 040530 001003 BNE 25$ ;NO, ISSUE COMMAND
8062 040532 052765 000200 000014 BIS #W.WCK,P.PRST(R5) ;YES, ISSUE WRITE BEFORE WRITE CHECK
8063 040540 004037 051406 25$: JSR RD,Q.PUSH ;ENQUEUE PARAMETER BLOCK IN COMMAND
8064 040544 001324 CINITQ ;INITIATION QUEUE
8065 040546 000207 RTS PC ;RETURN
8066
8067 040550 032765 000100 000150 EC4$: BIT #BIT6,P.DSTT(R5) ;CHECK IF WAITING FOR READ STATUS
8068 BEQ EC5$ ;FOR POSITIONING TIME OUT
8069 040556 001436 BEQ EC5$ ;NO, CONTINUE
8070 040560 042765 000100 000150 BIC #BIT6,P.DSTT(R5) ;CLEAR WAITING FOR READ STATUS
8071 040566 113765 001666 000001 MOVB POSCMD,P.CMND(R5) ;RESTORE COMMAND OF ERROR

```

8072	040574	032765	000200	000036		BIT	#DRDY,P.DS(R5)	;CHECK IF DRIVE READY
8073	040602	001004				BNE	10\$;YES, REISSUE COMMAND
8074	040604	004037	030464			JSR	RD,PPI100	;PRINT DRIVE TIME OUT DURING POSITIONING
8075	040610	062435				ERR052		;AND DROP DRIVE
8076	040612	000207			5\$:	RTS	PC	;RETURN
8077								
8078	040614	004037	030654		10\$:	JSR	RD,FRI200	;PRINT DRIVE TIME OUT DURING
8079	040620	062435				ERR052		;POSITIONING
8080	040622	004037	031300			JSR	RD,NERSTT	;TAKE OTHER ERROR STATICS
8081	040626	040612				5\$;NO RECOVERY RETURN
8082	040630	052765	004000	000150		BIS	#BIT11,P.DSTT(R5)	;SET RECALIBRATE ISSUED
8083	040636	112765	000113	000001		MOVB	#RECAL,P.CMND(R5)	;ISSUE RECALIBRATE
8084	040644	004037	051406			JSR	RD,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN COMMAND
8085	040650	001324				CINITQ		;INITIATION QUEUE
8086	040652	000207				RTS	PC	;RETURN
8087								
8088	040654	032765	000020	000150	EC5\$:	BIT	#BIT4,P.DSTT(R5)	;CHECK IF HEADER ERROR
8089	040662	001002				BNE	1\$;YES, GO PROCESS ERROR
8090	040664	000137	041400			JMP	EC6\$;NO, CONTINUE
8091								
8092	040670	042765	000020	000150	1\$:	BIC	#BIT4,P.DSTT(R5)	;CLEAR READ ALL HEADERS ISSUED
8093	040676	010346				MOV	R3,-(SP)	;STORE R3 ON THE STACK
8094	040700	016546	000056			MOV	P.B11(R5),-(SP)	;STORE SECTOR READ
8095	040704	042716	177017			BIC	#(C<M.SECT>),(SP)	;KEEP ONLY SECTOR COUNT
8096	040710	006216				ASR	(SP)	;SHIFT 4 BITS RIGHT
8097	040712	006216				ASR	(SP)	
8098	040714	006216				ASR	(SP)	
8099	040716	006216				ASR	(SP)	;GET SECTOR COUNT
8100	040720	005046				CLR	-(SP)	;MAKE ROOM ON THE STACK
8101	040722	116516	000004			MOVB	P.SECT(R5),(SP)	;GET DESIRED SECTOR
8102	040726	162616				SUB	(SP)+,(SP)	;CALCULATE DIFFERENCE
8103	040730	100410				BMI	2\$;BRANCH IF IN LOWER PART OF BUFFER
8104	040732	012703	066204			MOV	#BUFADD-6,R3	;LOAD BASE OF HIGHER PART OF BUFFER
8105	040736	006316				ASL	(SP)	;MULTIPLY SECTOR COUNT BY 6
8106	040740	011646				MOV	(SP),-(SP)	
8107	040742	006316				ASL	(SP)	
8108	040744	062616				ADD	(SP)+,(SP)	
8109	040746	162603				SUB	(SP)+,R3	;CALCULATE HEADER ADDRESS
8110	040750	000407				BR	3\$;CHECK HEADER
8111								
8112	040752	005416			2\$:	NEG	(SP)	;MAKE COUNT POSITIVE
8113	040754	006316				ASL	(SP)	;MULTIPLY DIFFERENCE BY 6
8114	040756	011603				MOV	(SP),R3	
8115	040760	006316				ASL	(SP)	
8116	040762	062603				ADD	(SP)+,R3	
8117	040764	062703	066000			ADD	#HDBUFF-6,R3	;CALCULATE HEADER ADDRESS
8118	040770	012337	001646		3\$:	MOV	(R3)+,HEAD1	;GET FIRST WORD OF HEADER
8119	040774	012337	001650			MOV	(R3)+,HEAD2	;GET SECOND WORD OF HEADER
8120	041000	011337	001652			MOV	(R3),HEAD3	;GET THIRD WORD OF HEADER
8121	041004	012603				MOV	(SP)+,R3	;RESTORE R3
8122	041006	004737	036776			JSR	PC,RSTAT	;RESTORE STATUS
8123	041012	013746	001646			MOV	HEAD1,-(SP)	;STORE FIRST WORD OF THE HEADER
8124	041016	043716	001650			BIC	HEAD2,(SP)	;HD1 & HD2
8125	041022	013746	001650			MOV	HEAD2,-(SP)	;STORE SECOND WORD OF HEADER
8126	041026	043716	001646			BIC	HEAD1,(SP)	;HD1 & HD2
8127	041032	052616				BIS	(SP)+,(SP)	;GENERATE EXPECTED HEADER VRC

8128	041034	023726	001652			CMP	HEAD3,(SP)+	:CHECK IF HEADER VRC IS CORRECT
8129	041040	001501				BEQ	15\$:NO CONTINUE
8130	041042	004037	030654			JSR	RO,PR1200	:PRINT HEADER VRC ERROR
8131	041046	062262				ERR043		
8132	041050	105765	000120			TSTB	P.IERC(R5)	:CHECK IF INHIBIT ERROR RECOVERY
8133	041054	001016				BNE	5\$:YES, INDICATE UNSUCCESSFUL RECOVERY
8134	041056	105265	000147	4\$:		INCB	P.RERD(R5)	:INCREMENT REREAD COUNT
8135	041062	122765	000043	000147		CMPB	#43,P.RERD(R5)	:CHECK IF UNRECOVERABLE
8136	041070	101410				BLOS	5\$:YES, INDICATE UNRECOVERABLE ERROR
8137	041072	122765	000121	000123		CMPB	#RDDATA,P.RCMD(R5)	:CHECK IF READ DATA COMMAND
8138	041100	001407				BEQ	6\$:YES, START ERROR RECOVERY
8139	041102	122765	000021	000147		CMPB	#21,P.RERD(R5)	:CHECK IF UNRECOVERABLE WRITE
8140	041110	101021				BHI	8\$:NO, START ERROR RECOVERY
8141	041112	004737	031502	5\$:		JSR	PC,UNRECY	:INDICATE UNRECOVERABLE ERROR
8142	041116	000207				RTS	PC	:RETURN
8143								
8144	041120	122765	000021	000147	6\$:	CMPB	#21,P.RERD(R5)	:CHECK IF BEGINNING OFFSETTING OPERATION
8145	041126	001007				BNE	7\$:NO, DETERMINE PROPER OFFSET
8146	041130	010446				MOV	R4, -(SP)	:STORE R4 ON THE STACK
8147	041132	116504	000000			MOV	P.DRVN(R5),R4	:GET DRIVE NUMBER FOR INDEX
8148	041136	146437	001514	001513		BICB	INTMSK(R4),0.OVER	:DO NOT DO IMPLIED SEEKS
8149	041144	012604				MOV	(SP)+,R4	:RESTORE R4
8150	041146	004037	037026	7\$:		JSR	RO,DETOFF	:DETERMINE OFFSET VALUE
8151	041152	041242				12\$:OFFSET ISSUED RETURN
8152	041154	116565	000123	000001	8\$:	MOV	P.RCMD(R5),P.CMND(R5)	:STORE COMMAND
8153	041162	013765	001642	000002		MOV	RTYCYL,P.CYLN(R5)	:GET CYLINDER
8154	041170	013765	001640	000004		MOV	RTYSEC,P.SECT(R5)	:GET TRACK AND SECTOR
8155	041176	013765	001634	000010		MOV	RTYBA,P.BALO(R5)	:GET BUS ADDRESS
8156	041204	013765	001636	000012		MOV	RTYWC,P.WC(R5)	:GET WORD COUNT
8157	041212	005465	000012			NEG	P.WC(R5)	:MAKE IT POSITIVE
8158	041216	122765	000131	000001		CMPB	#WRTCHK,P.CMND(R5)	:CHECK IF WRITE CHECK COMMAND
8159	041224	001003				BNE	10\$:NO, GO ISSUE COMMAND
8160	041226	052765	000200	000014		BIS	#W.WCK,P.PRST(R5)	:ISSUE WRITE BEFORE WRITE CHECK
8161	041234	004037	051406	10\$:		JSR	RO,Q.PUSH	:ENQUEUE PARAMETER BLOCK ON COMMAND
8162	041240	001324				CINITQ		:INITIATION QUEUE
8163	041242	000207			12\$:	RTS	PC	:RETURN
8164								
8165	041244	023765	001646	000030	15\$:	CMP	HEAD1,P.DCYL(R5)	:CHECK IF CYLINDER CORRECT
8166	041252	001420				BEQ	20\$:YES, CHECK SECTOR TRACK AND FORMAT
8167	041254	004037	030654			JSR	RO,PR1200	:PRINT MISPOSITIONING
8168	041260	063075				ERR064		
8169	041262	004037	031346			JSR	RO,OP1STT	:GO TAKE STATISTICS
8170	041266	041312				17\$:UNRECOVERABLE ERROR RETURN
8171	041270	052765	004000	000150		BIS	#BIT11,P.DSTT(R5)	:SET RECALIBRATE ISSUED
8172	041276	112765	000113	000001		MOV	#RECAL,P.CMND(R5)	:ISSUE RECALIBRATE
8173	041304	004037	051406			JSR	RO,Q.PUSH	:ENQUEUE PARAMETER BLOCK IN
8174	041310	001324				CINITQ		:COMMAND INITIATION QUEUE
8175	041312	000207			17\$:	RTS	PC	:RETURN
8176								
8177	041314	005046			20\$:	CLR	-(SP)	:MAKE ROOM ON STACK
8178	041316	113766	001641	000001		MOV	RTYTRK,1(SP)	:GET TRACK
8179	041324	006216				ASR	(SP)	:SHIFT RIGHT 3 BITS
8180	041326	006216				ASR	(SP)	
8181	041330	006216				ASR	(SP)	
8182	041332	153716	001640			BISB	RTYSEC,(SP)	:GENERATE EXPECTED SECOND WORD
8183	041336	052716	140000			BIS	#140000,(SP)	:PUT IN GO TRACK BITS

8184	041342	022637	001650			CMP	(SP)+,HEAD2	;CHECK IF SECOND WORD IS CORRECT
8185	041346	001407				BEQ	25\$;CHECK FOR BAD SECTOR
8186	041350	004037	030654			JSR	RO,PRI200	;INDICATE OPERATION INCOMPLETE
8187	041354	062243				ERR042		
8188	041356	004037	031346			JSR	RO,OPISTT	;GO TAKE STATISTICS
8189	041362	041312				17\$;UNRECOVERABLE RETURN
8190	041364	000673				BR	8\$;GO ISSUE LAST COMMAND
8191								
8192	041366	004037	030654		25\$:	JSR	RO,PRI200	;TYPE HEADER ERROR
8193	041372	064232				ERR209		
8194	041374	000137	041056			JMP	4\$;GO TRY TO RETRY COMMAND
8195								
8196	041400	122765	000021	000147	EC6\$:	CMPB	#21,P.RERD(R5)	;CHECK IF REISSUE DATA TRANSFER
8197	041406	101036				BHI	10\$;YES, INDICATE SUCCESSFUL RECOVERY
8198	041410	001424				BEQ	5\$;NO, OFFSET = +400 MICRO-INCHES
8199								ISSUE DATA TRANSFER
8200	041412	122765	000024	000147		CMPB	#24,P.RERD(R5)	;CHECK IF OFFSET -400 MICRO-INCHES
8201	041420	001420				BEQ	5\$;YES, ISSUE DATA TRANSFER
8202	041422	122765	000027	000147		CMPB	#27,P.RERD(R5)	;CHECK IF OFFSET +800 MICRO-INCHES
8203	041430	001414				BEQ	5\$;YES, ISSUE DATA TRANSFER
8204	041432	122765	000032	000147		CMPB	#32,P.RERD(R5)	;CHECK IF OFFSET -800 MICRO-INCHES
8205	041440	001410				BEQ	5\$;YES, ISSUE DATA TRANSFER
8206	041442	122765	000035	000147		CMPB	#35,P.RERD(R5)	;CHECK IF OFFSET +1200 MICRO-INCHES
8207	041450	001404				BEQ	5\$;YES, ISSUE DATA TRANSFER
8208	041452	122765	000040	000147		CMPB	#40,P.RERD(R5)	;CHECK IF OFFSET -1200 MICRO-INCHES
8209	041460	001011				BNE	10\$;NO, INDICATE SUCCESSFUL RECOVERY
8210	041462	105265	000147		5\$:	INCB	P.RERD(R5)	INCREMENT REREAD COUNT
8211	041466	116565	000123	000001		MOVB	P.RCMD(R5),P.CMND(R5)	;REISSUE COMMAND
8212	041474	004037	051406			JSR	RO,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN COMMAND
8213	041500	001324				CINITQ		INITIATION QUEUE
8214	041502	000207				RTS	PC	;RETURN
8215								
8216	041504	032765	040000	000150	10\$:	BIT	#BIT14,P.DSTT(R5)	;CHECK IF SERVICING BAD SECTOR
8217	041512	001403				BEQ	11\$;NO, CONTINUE
8218	041514	005765	000146			TST	P.RECT(R5)	;CHECK IF IN RETRY SEQUENCE
8219	041520	001472				BEQ	EC7\$;NO, GO TO BAD SECTOR SERVICE ROUTINE
8220	041522	105765	000210		11\$:	TSTB	P.ASSN(R5)	;CHECK IF DRIVE IS BEING ASSIGNED
8221	041526	001403				BEQ	12\$;NO, CHECK IF SEEK TO PREVIOUS CYLINDER
8222	041530	004737	036276			JSR	PC,SURECY	;INDICATE SUCCESSFUL RECOVERY
8223	041534	000207				RTS	PC	;RETURN
8224								
8225	041536	122765	000117	000001	12\$:	CMPB	#SEEK,P.CMND(R5)	;CHECK IF SEEK TO PREVIOUS CYLINDER
8226	041544	001404				BEQ	14\$;YES, INDICATE SUCCESSFUL RECOVERY
8227	041546	122765	000140	000001		CMPB	#RELEAS,P.CMND(R5)	;CHECK IF RELEASE
8228	041554	001003				BNE	16\$;NO, PROCESS DATA TRANSFER
8229	041556	004737	036276		14\$:	JSR	PC,SURECY	;INDICATE SUCCESSFUL RECOVERY
8230	041562	000207				RTS	PC	;RETURN
8231								
8232	041564	004037	025032		16\$:	JSR	RO,CHKADD	;CHECK SECTOR, TRACK, CYLINDER, BUS ADDRESS
8233								AND WORD COUNT
8234	041570	041626				20\$		ERROR RETURN
8235	041572	005037	177776			CLR	PS	ALLOW RK06 INTERRUPTS
8236	041576	122765	000121	000001		CMPB	#RDATA,P.CMND(R5)	;CHECK IF READ DATA
8237	041604	001021				BNE	30\$;NO, CHECK IF WRITE CHECK
8238	041606	004037	023304			JSR	RO,BUFER1	;FIND FIRST DATA ERROR
8239	041612	041630				25\$		ERROR RETURN

8240	041614	013737	001352	177776	18\$:	MOV	RKPRI,PS	:LOCK OUT RK06 INTERRUPTS
8241	041622	004737	036276			JSR	PC,SURECY	:INDICATE SUCCESSFUL RECOVERY
8242	041626	000207			20\$:	RTS	PC	:RETURN
8243								
8244	041630	013737	001452	177776	25\$:	MOV	RKPRI,PS	:LOCK OUT RK06 INTERRUPTS
8245	041636	052765	000004	000212		BIS	#BIT2,P.ERR(R5)	:INDICATE DATA COMPARE ERROR
8246	041644	000137	035310			JMP	E10\$:PROCESS DATA COMPARE ERROR
8247								
8248	041650	122765	000131	000001	30\$:	CMPB	#WRTCHK,P.CMND(R5)	:CHECK IF WRITE CHECK
8249	041656	001356				BNE	18\$:NO, INDICATE SUCCESSFUL RECOVERY
8250	041660	032765	000200	000014		BIT	#W.WCK,P.PRST(R5)	:CHECK IF WRITE CHECK ISSUED
8251	041666	001752				BEQ	18\$:YES, INDICATE SUCCESSFUL RECOVERY
8252	041670	042765	000200	000014		BIC	#W.WCK,P.PRST(R5)	:CLEAR WRITE BEFORE WRITE CHECK
8253	041676	004037	051406			JSR	RO,Q.PUSH	:ENQUEUE COMMAND IN COMMAND INITIATION QUEUE
8254	041702	001324				CINITQ		
8255	041704	000207				RTS	PC	:RETURN
8256								
8257	041706	004037	025032		EC7\$:	JSR	RO,CHKADD	:CHECK IF BUS ADD, WORD COUNT, SECTOR, TRACK AND CYLINDER ARE CORRECT
8258								:ERROR RETURN
8259	041712	042000			9\$:			
8260	041714	042765	040000	000150		BIC	#BIT14,P.DSTT(R5)	:CLEAR SERVICING BAD SECTOR
8261	041722	004737	042142			JSR	PC,BSTRAN	:CALCULATE NUMBER OF WORDS TRANSFERRED
8262	041726	013765	001660	000012		MOV	BSWC,P.WC(R5)	:LOAD BUS ADDRESS FOR NEXT DATA TRANSFER
8263	041734	100022				BPL	10\$:CHECK IF DATA TRANSFER FINISHED
8264	041736	013765	001654	000004		MOV	BSSECT,P.SECT(R5)	:LOAD SECTOR AND TRACK FOR NEXT COMMAND
8265	041744	013765	001656	000002		MOV	BSCYLN,P.CYLN(R5)	:LOAD CYLINDER FOR NEXT COMMAND
8266	041752	013765	001662	000010		MOV	BSBA,P.BALO(R5)	:LOAD BUS ADDRESS FOR NEXT DATA TRANSFER
8267	041760	052765	000200	000014		BIS	#W.WCK,P.PRST(R5)	:GO ISSUE WRITE
8268	041766	004037	051406		5\$:	JSR	RO,Q.PUSH	:ENQUEUE COMMAND IN COMMAND INITIATION QUEUE
8269	041772	001324				CINITQ		
8270	041774	005037	001566			CLR	ERRPRO	:CLEAR ERROR PROCESSING FLAG
8271	042000	000207			9\$:	RTS	PC	:RETURN
8272								
8273	042002	105765	000210		10\$:	TSTB	P.ASSN(R5)	:CHECK IF PACK IS BEING WRITTEN
8274	042006	001012				BNE	13\$:YES, CHECK IF SEEK TO NEXT CYLINDER
8275	042010	004737	022642			JSR	PC,BUFREL	:RELEASE BUFFER
8276	042014	004037	051406			JSR	RO,Q.PUSH	:ENQUEUE PARAMETER BLOCK IN DRIVE AVAILABLE QUEUE
8277	042020	001314				AVAILQ		
8278	042022	004737	022342			JSR	PC,GETBUF	:GET NEXT BUFFER FOR DAT TRANSFER
8279	042026	005037	001566			CLR	ERRPRO	:CLEAR ERROR BEING PROCESSING
8280	042032	000207				RTS	PC	:RETURN
8281								
8282	042034	005037	001566		13\$:	CLR	ERRPRO	:CLEAR ERROR BEING PROCESSED
8283	042040	013765	001654	000026		MOV	BSSECT,P.DTS(R5)	:LOAD NEXT SECTOR AND TRACK
8284	042046	013765	001656	000030		MOV	BSCYLN,P.DCYL(R5)	:LOAD NEXT CYLINDER
8285	042054	032777	040000	137056		BIT	#SW14,#SWR	:CHECK IF LOOP ON CURRENT OPERATION
8286	042062	001002				BNE	14\$:YES, ISSUE SEEK TO PREVIOUS CYLINDER
8287	042064	000137	015276			JMP	WV0\$:NO, CONTINUE TO WRITE PACK
8288								
8289	042070	004737	022642		14\$:	JSR	PC,BUFREL	:RELEASE BUFFER
8290	042074	032765	002000	000014		BIT	#DAPDRV,P.PRST(R5)	:CHECK IF DROP DRIVE
8291	042102	001402				BEQ	15\$:NO ISSUE SEEK TO PREVIOUS CYLINDER
8292	042104	000137	016040			JMP	WV1\$:DROP DRIVE
8293								
8294	042110	112765	177777	000211	15\$:	MOVB	#-1,P.SEEK(R5)	:SET SEEK TO PREVIOUS CYLINDER
8295	042116	016565	000136	000002		MOV	P.LCYL(R5),P.CYLN(R5)	:GET LAST CYLINDER

CZR6PCD RK611.06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 186
ERROR RECOVERY SEQUENCE NORMAL RETURN

SEQ 0184

8296	042124	016565	000140	000004
8297	042132	112765	000117	000001
8298	042140	000712		

MOV	P.LSEC(R5),P.SECT(R5)	:GET LAST TRACK AND SECTOR
MOVW	#SEEK,P.CMND(R5)	:LOAD COMMAND
BR	SS	:GO ISSUE SEEK COMMAND

```

      .SBTTL  CALCULATE WORDS TRANSFERRED UNTIL BAD SECTOR
8299
8300
8301 0421142 016537 000026 001654 BSTRAN: MOV      P.DTS(R5),BSSECT ;STORE ADDRESS OF BAD SECTOR
8302 0421150 016537 000030 001656      MOV      P.DCYL(R5),BSCYLN
8303 0421156 105237 001654      INCB     BSSECT ;ADVANCE TO NEXT SECTOR ON PACK
8304 0421162 122737 000025 001654      CMPB    #21.,BSSECT ;CHECK FOR SECTOR OVERFLOW
8305 0421170 103014      BHIS    $$$      NO CONTINUE
8306 0421176 105037 001654      CLRB    BSSECT ;CLEAR SECTOR
8307 0421176 105237 001655      INCB     BSTRACK ;INCREMENT TRACK
8308 0421202 122737 000002 001655      CMPB    #2,BSTRACK ;CHECK FOR TRACK OVERFLOW
8309 0421210 103004      BHIS    $$$      NO CONTINUE
8310 0421210 105037 001655      CLRB    BSTRACK ;CLEAR TRACK
8311 0421216 005237 001656      INC     BSCYLN ;INCREMENT CYLINDER
8312 0421222 013746 001656      MOV     BSCYLN,-(SP) ;STORE CYLINDER ON STACK
8313 0421230 166516 000002      SUB     P.CYLN(R5),(SP) ;SUBTRACT STARTING CYLINDER
8314 0421236 011646      MOV     (SP),-(SP) ;SAVE DIFFERENCE
8315 0421237 006316      ASL     (SP) ;MULTIPLY CYLINDER BY 3
8316 0421236 062616      ADD     (SP)+,(SP)
8317 0421240 005046      CLR     -(SP) ;MAKE ROOM ON STACK
8318 0421242 113716 001655      MOVB    BSTRACK,(SP) ;PLACE TRACK ON STACK
8319 0421246 062616      ADD     (SP)+,(SP) ;ADD TRACK
8320 0421250 005046      CLR     -(SP) ;MAKE ROOM ON THE STACK
8321 0421252 116516 000005      MOVB    P.TRACK(R5),(SP) ;STORE TRACK
8322 0421256 162616      SUB     (SP)+,(SP) ;SUBTRACT INITIAL TRACK
8323 0421260 012746 000026      MOV     #22,-(SP) ;STORE 22 ON STACK FOR MULTIPLICATION
8324 0421264 004737 054714      JSR     PC,$MULT ;MULTIPLY BY 22
8325 0421270 005046      CLR     -(SP) ;MAKE ROOM ON THE STACK
8326 0421272 113716 001654      MOVB    BSSECT,(SP) ;STORE BAD SECTOR
8327 0421276 062616      ADD     (SP)+,(SP) ;ADD BAD SECTOR ADDRESS
8328 0421300 005046      CLR     -(SP) ;MAKE ROOM ON THE STACK
8329 0421302 116516 000004      MOVB    P.SECT(R5),(SP) ;STORE ORIGINAL SECTOR ADDRESS
8330 0421306 162616      SUB     (SP)+,(SP) ;SUBTRACT ORIGINAL SECTOR ADDRESS
8331 0421310 111666 000001      MOVB    (SP),1(SP) ;KEEP NUMBER OF SECTORS TRANSFERRED
8332 0421314 105016      CLRB    (SP)
8333 0421316 011637 001664      MOV     (SP),BSWORD ;LOAD WORD COUNT
8334 0421322 006316      ASL     (SP) ;MULTIPLY BY 2
8335 0421324 066516 000010      ADD     P.BALO(R5),(SP) ;GET STARTING ADDRESS FOR NEW DATA
8336 0421330 011637 001662      MOV     (SP),BSBA ;GET NEW BUS ADDRESS
8337 0421334 166516 000132      SUB     P.RBAL(R5),(SP) ;GET NUMBER OF BYTES TRANSFERRED
8338 0421340 006216      ASR     (SP) ;CONVERT TO WORDS (DIVIDE BY 2)
8339 0421342 066516 000130      ADD     P.RWC(R5),(SP) ;DETERMINE REMAINING WORD COUNT
8340 0421346 012637 001660      MOV     (SP)+,BSWC ;STORE CALCULATED WORD COUNT
8341 0421352 005726      TST     (SP)+ ;THROW AWAY MOST SIGNIFICANT BITS
8342 0421354 000207      RTS     PC ;RETURN

```

```

      .SBTTL  BAD SECTOR HANDLING
8343
8344
8345 042356 032765 040000 000150 BDSECT: BIT      #BIT14,P.DSTT(R5) ;CHECK IF PRESENTLY SERVICING BAD SECTOR
8346 042364 001401 000000 000000 BEQ      5$          ;NO, CONTINUE
8347 042366 000000 000000 000000 HALT
8348 042370 132765 000010 000210 5$: BITB    #BIT3,P.ASSN(R5) ;CHECK IF READING PACK SERIAL NUMBER
8349 042376 001157 000000 000000 BNE     40$          ;YES, DETERMINE NEXT SECTOR
8350 042400 004737 042142 000000 JSR     PC,BSTRAN   ;CALCULATE WORDS TRANSFERRED
8351 042400 122765 000121 000001 CMPB    #RODATA,P.CMND(R5) ;CHECK IF READ COMMAND
8352 042412 001076 000000 000000 BNE     25$          ;NO, DO NO CHECK BUS ADDRESS AND WORD COUNT
8353 042414 013746 001662 000000 MOV     BSBA,-(SP)   ;STORE BUS ADDRESS
8354 042420 162716 001000 000000 SUB     #256,#2(SP)  ;SUBTRACT ADDITIONAL SECTOR
8355 042424 026526 000024 000000 CMP     P.BAP(R5),(SP)+ ;CHECK IF BUS ADDRESS CORRECT
8356 042430 001404 000000 000000 BEQ     10$          ;YES, CONTINUE
8357 042432 004037 037322 000000 JSR     RO,CONERR   ;PRINT "BUS ADDRESS INVALID"
8358 042436 063214 000000 000000 ERR071
8359 042440 000207 000000 000000 RTS     PC          ;RETURN
8360
8361 042442 016546 000022 000000 10$: MOV     P.WCR(R5),-(SP) ;STORE WORD COUNT FOR COMPARISON
8362 042446 062716 000400 000000 ADD     #256,(SP)    ;SUBTRACT OFF BAD SECTOR
8363 042452 023726 001660 000000 CMP     BSWC,(SP)+  ;CHECK IF WORD COUNT CORRECT
8364 042456 001404 000000 000000 BEQ     15$          ;YES, CONTINUE
8365 042460 004037 037322 000000 JSR     RO,CONERR   ;PRINT "WORD COUNT INVALID"
8366 042464 063171 000000 000000 ERR070
8367 042466 000207 000000 000000 RTS     PC          ;RETURN
8368
8369 042470 042765 100200 000150 15$: BIC     #BIT15:BIT7,P.DSTT(R5) ;CLEAR FIRST ERROR AND ERROR ENQUEUED
8370 042476 022737 000400 001664 CMP     #256.,BSWORD ;CHECK IF NO WORDS TRANSFERRED
8371 042504 001413 000000 000000 BEQ     20$          ;YES, DO NOT CHECK DATA
8372 042506 004037 023304 000000 JSR     RO,BUFER1   ;CHECK DATA READ
8373 042512 043046 000000 000000 S0$    ;ERROR RETURN
8374 042514 063765 001664 000012 ADD     BSWORD,P.WC(R5) ;CALCULATE NEW WORD COUNT
8375 042522 100410 000000 000000 BMI     22$          ;CHECK IF DATA TRANSFER NOT COMPLETE
8376 042524 000464 000000 000000 BR     30$          ;DATA TRANSFER COMPLETE
8377
8378 042526 052765 000200 000014 19$: BIS     #W.WCK,P.PRST(R5) ;ISSUE WRITE BEFORE WRITE CHECK
8379 042534 013765 001660 000012 20$: MOV     BSWC,P.WC(R5) ;GET NUMBER OF WORDS TO BE TRANSFERRED
8380 042542 100055 000000 000000 BPL     30$          ;CHECK IF DATA TRANSFER COMPLETE
8381 042544 013765 001656 000002 22$: MOV     BSCYLN,P.CYLN(R5) ;GET NEW CYLINDER
8382 042552 013765 001654 000004 MOV     BSSECT,P.SECT(R5) ;GET NEW SECTOR
8383 042560 013765 001662 000010 MOV     BSBA,P.BALO(R5) ;GET NEW BUS ADDRESS
8384 042566 004037 051406 000000 JSR     RO,Q.PUSH   ;PUT PARAMETER IN COMMAND
8385 042572 001324 000000 000000 CINITQ ;INITIATION QUEUE
8386 042574 005765 000146 000000 TST     P.RECT(R5)  ;CHECK IF ERROR RECOVERY IN PROGRESS
8387 042600 001002 000000 000000 BNE     23$          ;YES, DO NOT RESET ERROR PROCESSING IN PROGRESS
8388 042602 005037 001566 000000 CLR     ERRPRO     ;RESET ERROR PROCESSING IN PROGRESS
8389 042606 000207 000000 000000 23$: RTS     PC          ;RETURN
8390
8391 042610 042765 100200 000150 25$: BIC     #BIT15:BIT7,P.DSTT(R5) ;CLEAR FIRST ERROR AND ERROR ENQUEUED
8392 042616 12765 000131 000001 CMPB    #WRTCHK,P.CMND(R5) ;CHECK IF WRITE CHECK COMMAND
8393 042624 0C1343 000000 000000 BNE     20$          ;NO, CONTINUE
8394 042626 022737 000400 001664 CMP     #256.,BSWORD ;CHECK IF NO WORDS TRANSFERRED
8395 042634 001734 000000 000000 BEQ     19$          ;YES, ISSUE NEXT DATA TRANSFER
8396 042636 012746 000400 000000 MOV     #256,-(SP)   ;STORE EXCESS WORD COUNT ON STACK
8397 042642 163716 001664 000000 SUB     BSWORD,(SP)  ;CALCULATE NEW WORD COUNT
8398 042646 012665 000012 000000 MOV     (SP)+,P.WC(R5) ;STORE NEW WORD COUNT

```

8399	042652	052765	040000	000150		BIS	#BIT14,P.DSTT(R5)	;INDICATE THAT BAD
8400								SECTOR IS BEING PROCESSED
8401	042660	042765	000200	000014		BIC	#W.WCK,P.PRST(R5)	;DO WRITE CHECK PART OF COMMAND
8402	042666	004037	051406			JSR	RO,Q.PUSH	;PUT PARAMETER BLOCK IN COMMAND
8403	042672	001324				CINITQ		;INITIATION QUEUE
8404	042674	000207				RTS	PC	;RETURN
8405								
8406	042676	004737	022642		30\$:	JSR	PC,BUFREL	;RELEASE BUFFER
8407	042702	005765	000146			TST	P.RECT(R5)	;DETERMINE IF CURRENTLY UNDERGOING ERROR
8408								RECOVERY
8409	042706	001403				BEQ	35\$;YES, PUT IN AVAILIABLE QUEUE
8410	042710	004737	036276			JSR	PC,SURECY	;INDICATE SUCCESSFUL RECOVERY
8411	042714	000207				RTS	PC	;RETURN
8412								
8413	042716	004037	051406		35\$:	JSR	RO,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN
8414	042722	001314				AVAILQ		AVAILIABLE QUEUE
8415	042724	004737	022342			JSR	PC,GETBUF	;GET NEXT BUFFER
8416	042730	005037	001566			CLR	ERRPRO	;CLEAR ERROR BEING PROCESSED
8417	042734	000207				RTS	PC	;RETURN
8418								
8419	042736	042765	100200	000150	40\$:	BIC	#BIT15!BIT7,P.DSTT(R5)	;CLEAR FIRST ERROR AND ERROR ENQUEUED
8420	042744	122737	000010	000126		CMPB	#8,P.RSEC	;CHECK IF ALL SECTORS HAVE BEEN READ
8421	042752	101422				BLOS	45\$;YES, DROP DRIVE FROM TEST SEQUENCE
8422	042754	116565	000123	000135		MOVB	P.RCMD(R5),P.LCMD(R5)	;STORE PREVIOUS COMMAND
8423	042762	016565	000124	000136		MOV	P.RCYL(R5),P.LCYL(R5)	
8424	042770	116565	000126	000140		MOV	P.RSEC(R5),P.LSEC(R5)	
8425	042776	016537	000130	000142		MOV	P.RWC(R5),P.LWC	
8426	043004	004037	051406			JSR	RO,Q.PUSH	;ENQUEUE PARAMETER BLOCK IN COMMAND
8427	043010	001324				CINITQ		;INITIATION QUEUE
8428	043012	005037	001566			CLR	ERRPRO	;CLEAR ERROR RECOVERY IN PROGRESS
8429	043016	000207				RTS	PC	;RETURN
8430								
8431	043020	004737	022642		45\$:	JSR	PC,BUFREL	;GO RELEASE BUFFER
8432	043024	104401	064012			TYPE	ERR204	
8433	043030	004737	013004			JSR	PC,DROP	;DROP DRIVE FROM TEST SEQUENCE
8434	043034	004737	022342			JSR	PC,GETBUF	;GET NEW BUFFER
8435	043040	005037	001566			CLR	ERRPRO	;CLEAR ERROR RECOVERY IN PROGRESS
8436	043044	000207				RTS	PC	;RETURN
8437								
8438	043046	052765	000004	000212	50\$:	BIS	#BIT2,P.ERR(R5)	;SET DATA TYPE ERROR
8439	043054	000137	035310			JMP	E10\$;GO PROCESS DATA ERROR

```

      .SBTTL  ERROR RECOVERY SEQUENCE ABNORMAL RETURN
8440
8441
8442 043060 032765 000120 000150 ERVY2: BIT      #BIT4!BIT6,P.DSTT(R5) ;CHECK IF DIAGNOSING SEQUENCE
8443 043066 001524          BEQ      35$          ;NO, GO TO ERROR RECOVERY SEQUENCE
8444 043070 004737 030000          JSR      PC,PRIO00          ;PRINT HEADER
8445 043074 104401 064201          TYPE    ,ERR208          ;TYPE "ERROR WHILE DIAGNOSING ----"
8446 043100 032765 000020 000150 BIT      #BIT4,P.DSTT(R5) ;CHECK IF HEADER TYPE ERROR
8447 043106 001005          BNE      2$          ;YES, PRINT MESSAGE
8448 043110 104401 062436          TYPE    ,ERR052+1        ;TYPE "TIME OUT WHILE DRIVE POSITION"
8449 043114 104401 001165          TYPE    ,SCRFLF          ;TYPE <CR><LF>
8450 043120 000402          BR       3$          ;GO PRINT STATUS
8451
8452 043122 104401 064233          2$: TYPE    ,ERR209+1        ;PRINT HEADER TYPE ERROR
8453 043126 032777 020000 136004 3$: BIT      #SW13,#SWR          ;CHECK IF INHIBIT PRINT OUT
8454 043134 001066          BNE      30$          ;YES, DO NOT PRINT STATUS
8455 043136 032765 000020 000212 BIT      #BIT4,P.ERR(R5) ;CHECK IF DRIVE SEIZED TIME OUT
8456 043144 001402          BEQ      4$          ;
8457 043146 104401 062753          TYPE    ,ERR059          ;
8458 043152 032765 000040 007212 4$: BIT      #BIT5,P.ERR(R5) ;CHECK IF TIME OUT WHILE DRIVE POSITIONING
8459 043160 001402          BEQ      5$          ;
8460 043162 104401 062436          TYPE    ,ERR052+1        ;
8461 043166 032765 000010 000014 5$: BIT      #UEXATT,P.PRST(R5) ;CHECK IF UNEXPECTED ATTENTION
8462 043174 001402          BEQ      6$          ;
8463 043176 104401 062032          TYPE    ,ERR030          ;
8464 043202 032765 000020 000014 6$: BIT      #DRVHRD,P.PRST(R5) ;CHECK IF DRIVE HARD ERROR
8465 043210 001402          BEQ      7$          ;
8466 043212 104401 061723          TYPE    ,ERR027          ;
8467 043216 032765 000040 000014 7$: BIT      #DRVDSC,P.PRST(R5) ;CHECK IF DSC DID NOT CLEAR
8468 043224 001402          BEQ      8$          ;
8469 043226 104401 061742          TYPE    ,ERR028          ;
8470 043232 032765 004000 000014 8$: BIT      #MODSC,P.PRST(R5) ;CHECK IF ATTN BUT NO DSC OR FAULT
8471 043240 001402          BEQ      11$          ;
8472 043242 104401 062676          TYPE    ,ERR058          ;
8473 043246 004737 026074          11$: JSR      PC,PRISTT          ;PRINT CONTROLLER STATUS
8474 043252 004737 025720          JSR      PC,PRDSTT          ;PRINT DRIVE STATUS
8475 043256 004737 030244          JSR      PC,PRIO02          ;PRINT RETRY COMMAND
8476 043262 010546          MOV      R5,-(SP)          ;STORE R5 ON STACK
8477 043264 012705 004516          MOV      #PARM10,R5          ;LOAD R5 FOR PREVIOUS STATUS
8478 043270 104401 064252          TYPE    ,ERR210          ;TYPE "FIRST ERROR INFO"
8479 043274 004737 026074          JSR      PC,PRISTT          ;PRINT CONTROLLER STATUS
8480 043300 004737 025720          JSR      PC,PRDSTT          ;PRINT DRIVE STATUS
8481 043304 004737 030244          JSR      PC,PRIO02          ;PRINT INITIAL COMMAND
8482 043310 012605          MOV      (SP)+,R5          ;RESTORE R5
8483 043312 005265 000206          30$: INC      P,NER(R5)          ;INCREMENT OTHER ERRORS
8484 043316 004737 031250          JSR      PC,HLT000          ;CHECK IF HALT ON ERROR
8485 043322 004737 022642          JSR      PC,BUFREL          ;RELEASE BUFFER
8486 043326 004737 013004          JSR      PC,DROP          ;DROP DRIVE
8487 043332 004737 022342          JSR      PC,GETBUF          ;GO ALLOCATE BUFFERS
8488 043336 000405          BR       40$          ;CHECK IF ERROR QUEUE EMPTY
8489
8490 043340 004737 032002          35$: JSR      PC,ERVY          ;GO DO ERROR RECOVERY
8491 043344 005737 001566          TST     ERAPRO          ;CHECK IF ERROR RECOVERY ON LAST DRIVE
8492
8493 043350 001007          BNE      45$          ;IS FINISHED
8494 043352 004037 051466          40$: JSR      RO,Q.POP          ;NO, RETURN
8495 043356 001330          ERRPRQ          ;GET NEXT ELEMENT FROM
; ERROR PROCESSING QUEUE

```

H15

CZR6PCD RK611 06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 191
ERROR RECOVERY SEQUENCE ABNORMAL RETURN

SEQ 0189

8496 043360 012605
8497 043362 010537 001566
8498 043366 001364
8499
8500 043370 000207

MOV (SP)+,R5
MOV R5,ERAPRO
BNE 35\$
45\$: RTS PC

:LOAD PARAMETER BLOCK
:LOAD CURRENT PROCESSING ERROR
:IF ERROR PROCESSING QUEUE NOT EMPTY GO
:PROCESS NEXT ERROR
:RETURN

.SBTTL DETERMINE IF CLOCK IS PRESENT

8501									
8502									
8503	043372	012737	043422	000004	CHKCLK:	MOV	#55,ERRVEC		;SET UP ERROR VECTOR FOR NON-EXISTENT MEMORY
8504	043400	012737	000340	000006		MOV	#PR7,ERRVEC+2		;LOCK ALL INTERRUPTS
8505	043406	005777	135724			TST	2\$CLKCSB		;CHECK FOR P CLOCK
8506	043412	112737	177777	001374		MOVB	#-1,CLKFLG		;INDICATE KW11-P PRESENT
8507	043420	000413				BR	20\$;LOAD COMMON CLOCK PARAMETERS
8508									
8509	043422	022626			5\$:	CMP	(SP)+,(SP)+		;ADJUST STACK
8510	043424	012737	043446	000004		MOV	#155,ERRVEC		;SET UP ERROR VECTOR FOR NON-EXISTENT MEMORY
8511	043432	005777	135704			TST	2\$CLKS		;CHECK FOR L CLOCK
8512	043436	112737	000001	001374		MOVB	#1,CLKFLG		;SET KW11-L PRESENT
8513	043444	000401				BR	20\$;REINSTATE TRAP CATCHER
8514									
8515	043446	022626			15\$:	CMP	(SP)+,(SP)+		;ADJUST STACK
8516	043450	012737	000006	000004	20\$:	MOV	#ERRVEC+2,ERRVEC		;REINSTATE TRAP CATCHER
8517	043456	005037	00000E			CLR	ERRVEC+2		
8518	043462	000207				RTS	PC		;RETURN


```

8519      .SETTL  SET UP CLOCK INTERRUPT
8520
8521 043464 105737 001374      CLKINT: TSTB  CLKFLG      ;CHECK IF A CLOCK IS ON SYSTEM
8522 043470 001431              BEQ    20$          ;NO RETURN
8523 043472 100011              BPL    $$          ;CHECK IF KW11-L
8524 043474 013701 001340      MOV    $LPVEC,R1   ;LOAD VECTOR ADDRESS
8525 043500 012777 177777 135630  MOV    #-1,$LKCSB ;LOAD COUNT BUFFER WITH 1'S
8526 043506 012777 000135 135620  MOV    #13,$LKCSR ;SET COUNT UP 16 MS. CONT
8527 043514 000405              BR     10$         ;LOAD COMMON CLOCK PARAMETERS
8528
8529 043516 013701 001344      5$:   MOV    $LLVEC,R1 ;LOAD VECTOR ADDRESS
8530 043522 012777 000100 135612  MOV    #100,$LKS   ;SET UP L CLOCK FOR INTERRUPT MODE
8531 043530 012721 043556      10$:  MOV    #CLOCK,(R1)+ ;LOAD ADDRESS OF INTERRUPT ROUTINE
8532 043534 012711 000300      MOV    #PR6,(R1)  ;SET UP FOR PRIORITY 6
8533 043540 113737 001352 001353  MOVB   HZ,CLKFRQ   ;SET UP CLOCK FREQUENCY
8534 043546 113737 001360 001375  MOVB   PERINV,PERIOD ;SET UP COUNTER FOR INITIAL STATISTIC SUMMARY
8535 043554 000207              20$:  RTS    PC        ;RETURN

```

```

      .SBTTL  KW11-L AND KW11-P INTERRUPT HANDLER
8536
8537
8538 043556 105337 001353      CLOCK:  DECB  CLKFRQ      ;CHECK IF ONE SECOND PASTED
8539 043562 001041          BNE  10$      ;NO, RETURN
8540 043564 113737 001352 001353  MOVB  HZ,CLKFRQ  ;REINITIALIZE SYSTEM CLOCK
8541 043572 005237 001372          INC  SECOND      ;INCREMENT SECOND COUNT
8542 043576 022737 000074 001372  CMP   #60.,SECOND ;CHECK IF MINUTE HAS OCCURRED
8543 043604 001030          BNE  10$      ;NO, RETURN
8544 043606 005037 001372          CLR  SECOND      ;CLEAR SECOND
8545 043612 105737 001360          TSTB PERINV     ;CHECK IF INTERVAL STATISTICS
8546 043616 001411          BEQ  $$        ;NO, UPDATE MINUTE COUNT
8547 043620 105337 001375          DECB PERIOD     ;DECREMENT PERIOD COUNT
8548 043624 001006          BNE  $$        ;CHECK IF END OF PERIOD, IF NOT UPDATE MINUTE COUNT
8549 043626 113737 001360 001375  MOVB  PERINV,PERIOD ;INITIALIZE PERIOD
8550 043634 112737 177777 001633  MOVB  #-1,STATIS ;INDICATE TIME FOR STATISTIC TYPE OUT
8551 043642 005237 001370          INC  MINUTE      ;INCREMENT MINUTE COUNTER
8552 043646 022737 000074 001370  CMP   #60.,MINUTE ;CHECK IF HOUR HAS OCCURED
8553 043654 001004          BNE  10$      ;NO, RETURN
8554 043656 005037 001370          CLR  *MINUTE    ;CLEAR MINUTE COUNT
8555 043662 005237 001366          INC  HOUR       ;INCREMENT HOUR COUNT
8556 043666 000002          RTS                ;RETURN

```

```
8557 .SBTTL PRINT TIME ROUTINE
8558
8559 043670 105737 001374 PRITIM: TSTB CLKFLG ;CHECK IF CLOCK ON SYSTEM
8560 043674 001455 SO$ ;NO, RETURN
8561 043676 010046 MOV RO,-(SP) ;SAVE RO
8562 043700 013746 177776 MOV PS,-(SP) ;SAVE PSW
8563 043704 012737 000340 177776 MOV #PR7,PS ;LOCK OUT CLOCK
8564 043712 013737 001366 001376 MOV HOUR,TIMHR ;SAVE TIME FOR PRINT OUT
8565 043720 013737 001370 001400 MOV MINUTE,TIMMIN
8566 043726 013737 001372 001402 MOV SECOND,TIMSEC
8567 043734 012637 177776 MOV (SP)+,PS ;RESTORE PSW
8568 043740 012700 064743 MOV #TIM001,RO ;LOAD RO FOR INDEX
8569 043744 105010 CLRB (RO) ;DETERMINE HUNDREDS OF HOURS
8570 043746 013746 001376 MOV TIMHR,-(SP) ;STORE HOURS FOR CONVERSION
8571 043752 162716 000144 1$: SUB #100,-(SP) ;CONVERT HUNDREDS
8572 043756 100402 BMI 2$ ;CHECK IF TOO SMALL
8573 043760 105210 INCB (RO)
8574 043762 000773 BR 1$
8575
8576 043764 062716 000144 2$: ADD #100,-(SP)
8577 043770 152720 000060 BISB #60,(RO)+ ;MAKE DIGIT ASCII
8578 043774 004737 044032 JSR PC,CONHUN
8579 044000 013716 001400 MOV TIMMIN,(SP) ;LOAD STACK FOR MINUTES
8580 044004 004737 044032 JSR PC,CONHUN
8581 044010 013716 001402 MOV TIMSEC,(SP) ;LOAD STACK FOR SECONDS
8582 044014 004737 044032 JSR PC,CONHUN
8583 044020 005726 TST (SP)+ ;ADJUST STACK
8584 044022 104401 064736 TYPE TIM000 ;TYPE TIME:
8585 044026 012600 MOV (SP)+,RO ;RESTORE RO
8586 044030 000207 SO$: RTS PC ;RETURN
8587
8588 .SBTTL CONVERT DECIMAL LESS THAN 100
8589
8590 044032 105010 CONHUN: CLRB (RO) ;ZERO DIGIT
8591 044034 162766 000012 000002 1$: SUB #10,-2(SP) ;CONVERT TENS
8592 044042 100402 BMI 5$
8593 044044 105210 INCB (RO)
8594 044046 000772 BR 1$
8595
8596 044050 062766 000012 000002 5$: ADD #10,-2(SP)
8597 044056 152720 000060 BISB #60,(RO)+ ;MAKE ASCII
8598 044062 105010 CLRB (RO) ;ZERO DIGIT
8599 044064 005766 000002 TST 2(SP)
8600 044070 001404 BEQ 7$
8601 044072 105210 6$: INCB (RO) ;CONVERT ONES
8602 044074 005366 000002 DEC 2(SP)
8603 044100 001374 BNE 6$
8604 044102 152720 000060 7$: BISB #60,(RO)+ ;MAKE ASCII
8605 044106 105720 TSTB (RO)+ ;SKIP OVER COLON
8606 044110 000207 RTS PC
```

8607
8608
8609
8610
8611
8612
8613
8614
8615
8616
8617
8618
8619
8620
8621
8622
8623
8624
8625
8626
8627
8628
8629
8630
8631
8632
8633
8634
8635
8636
8637
8638
8639
8640
8641
8642
8643
8644
8645
8646
8647
8648
8649
8650
8651
8652
8653
8654
8655
8656
8657
8658
8659
8660
8661
8662

044112 010546
044114 010446
044116 010346
044120 010246
044122 013746 177776
044126 005337 001466
044132 001046
044134 013737 001470 001466
044142 013737 001452 177776
044150 013702 001446
044154 005004
044156 005003
044160 136437 001514 001512 1\$:

.SBTTL RK611/RK06-RK07 UNIBUS DRIVER FOR QUEUED OPERATIONS (REV. 0.11)

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

.SBTTL *WATCH-DOG TIMER

```
*****
*
* THE WATCH-DOG TIMER DOES A PSEUDO-TIMING OF RK06/07 UNIBUS
* SUBSYSTEM COMMAND. SINCE ONE CAN NOT GUARANTEE THAT A
* REAL-TIME CLOCK (KW11-P OR KW11-L) IS ON THE SYSTEM
* THE RK06 DRIVER WILL USE THE LOCATION W.MTIM FOR
* MILLI-SECOND TIMING. WHEN W.MTIM REACHES ZERO THE
* WATCH-DOG TIMER WILL SCAN THE DRIVES IN USE AS
* DETERMINED BY THE LOCATION W.TIME. THE TIMER COUNTS
* (ONE FOR EACH DRIVE) ARE KEPT IN THE TABLE W.DRV.
* IF ANY COUNT IN THE TABLE W.DRV REACHES ZERO A COMMAND
* TIME-OUT WILL BE DESIGNATED IN THE PROGRAM DEVICE STATUS
* REGISTER OF THAT DRIVE'S PARAMETER BLOCK.
*
* THE DRIVER WILL USE THE LOCATION W.MIN AS THE NUMBER
* OF MILLISECONDS FOR AN UNLOAD OR START SPINDLE COMMAND.
* THE DRIVER WILL USE THE LOCATION W.SEC AS THE TIME
* LIMIT FOR ALL OTHER COMMANDS.
*
* FOR QUEUED OPERATIONS THE WATCH-DOG TIMER WILL
* WATCH UP TO 8 OPERATIONS SIMULTANEOUSLY. FOR SEQUENTIAL
* OPERATIONS ONLY ONE OPERATION WILL BE WATCHED.
*
*CALL JSR PC,W.WTCH
*      RETURN IF NO DRIVE ORDER EXCEEDED ITS TIME LIMIT
*
* OTHERWISE AN ABNORMAL RETURN TO THE ROUTINE ADDRESS
* BY LOCATION A.ABNL WILL OCCUR AND THE 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
        MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
        MOV RKBAS,R2 ;LOAD BASE OF RK06 REGISTERS
        CLR R4 ;CLEAR INDEX OF DRIVE MASKS
        CLR R3 ;CLEAR INDEX OF DRIVE TIMES
        ;AND PARAMETER BLOCK ADDRESSES
        BITB INTMSK(R4),W.TIME ;CHECK IF DRIVE IS BEING TIMED
```

8663	044166	001422				BEQ	6\$:	NO, GO TO NEXT DRIVE
8664	044170	005363	001544			DEC	W.DRV(R3)	:	DECREMENT DRIVE COUNT
8665	044174	001017				BNE	6\$:	IF NOT TIME OUT, GO TO
8666								:	NEXT DRIVE
8667	044176	6305	001524			MOV	PBLKT(R3),R5	:	LOAD ADDRESS OF PARAMETER
8668								:	BLOCK TABLE FOR INDEXING
8669	044202	146437	001514	001512		BICB	INT/ISK(R4),W.TIME	:	RESET TIMING INDICATOR FOR DRIVE
8670	044210	052765	000100	000014		BIS	#CMDTO P.PAST(R5)	:	SET COMMAND TIME OUT
8671	044216	020537	001464			CMP	R5,O.WAIT	:	CHECK IF DRIVER IS WAITING FOR
8672								:	COMMAND COMPLETION
8673	044222	001002				BNE	5\$:	NO, DO NOT ALTER WAITING FOR
8674								:	COMMAND COMPLETION
8675	044224	005037	001464			CLR	O.WAIT	:	CLEAR WAIT FOR COMMAND COMPLETION
8676	044230	004777	135222		5\$:	JSR	PC,2A.ABNL	:	BRANCH TO ERROR ROUTINE
8677	044234	062703	000002		6\$:	ADD	#2,R3	:	ADD 2 TO INDEX OF DRIVE TIMES
8678								:	AND PARAMETER BLOCK ADDRESSES
8679	044240	005204				INC	R4	:	INCREMENT INDEX OF DRIVE MASK
8680	044242	022704	000010			CMP	#10,R4	:	CHECK IF ALL DRIVES ARE TESTED
8681	044246	001344				BNE	1\$:	NO, CHECK NEXT DRIVE
8682	044250	012637	177776		20\$:	MOV	(SP)+,PS	:	RESTORE PSW
8683	044254	012602				MOV	(SP)+,R2	:	RESTORE R2
8684	044256	012603				MOV	(SP)+,R3	:	RESTORE R3
8685	044260	012604				MOV	(SP)+,R4	:	RESTORE R4
8686	044262	012605				MOV	(SP)+,R5	:	RESTORE R5
8687	044264	000207				RTS	PC	:	RETURN

.SBTTL *RK06-RK07 INTERRUPT SERVICE ROUTINE

8688
8689
8690
8691
8692
8693
8694
8695
8696
8697
8698
8699
8700
8701
8702
8703
8704
8705
8706
8707
8708
8709
8710
8711
8712
8713
8714
8715
8716
8717
8718
8719
8720
8721
8722
8723
8724
8725
8726
8727
8728
8729
8730
8731
8732
8733
8734
8735
8736
8737
8738
8739
8740
8741
8742
8743

044266 010546
044270 010446
044272 010346
044274 010246
044276 010146
044300 010046
044302 013702 001446
044306 016237 000010 001412
044314 032737 001000 001412
044322 001407
044324 052737 100000 001462
044332 004777 135122

```
*****
*
* THIS ROUTINE WILL SERVICE ALL RK06/07 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 ERR R RETURN
*
* FOR NORMAL AND ABNORMAL RETURNS, THE ADDRESS OF THE APPROPRIATE
* PARAMETER BLOCK WILL BE IN RS.
*
* 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
*
*****
```

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

8744	044336	000137	046674			JMP	I.RTRN	;RETURN
8745								
8746	044342	105737	001506	15:		TSTB	I.ISRL	;CHECK IF INTERRUPT OR RELEASE
8747	044346	001410				BEQ	65	;NO, CHECK IF DRIVE AVAILABLE
8748	044350	100403				BMI	55	;CHECK IF RELEASE COMMAND
8749	044352	105037	001506			CLRB	I.ISRL	;YES, CLEAR FLAG
8750	044356	000425				BR	I.IG0	;CONTINUE PROCESSING INTERRUPT
8751								
8752	044360	105037	001506	55:		CLRB	I.ISRL	;CLEAR FLAG
8753	044364	000137	045420			JMP	I.ATTN	;GO PROCESS DRIVE ATTENTIONS
8754								
8755	044370	032737	010400	001412	65:	BIT	#NED!UFE, T.CS2	;CHECK FOR NON-EXISTENT DRIVE OR
8756								UNIT FIELD ERROR
8757	044376	001415				BEQ	75	;NO, WAIT FOR DUAL ACCESS INTERRUPT
8758	044400	013704	001412			MOV	T.CS2, R4	;LOAD R4 FOR DRIVE NUMBER
8759	044404	042704	177770			BIC	#↑C<DRVMSK>, R4	;KEEP DRIVE BITS
8760	044410	010403				MOV	R4, R3	;STORE DRIVE NUMBER FOR INDEX
8761	044412	006303				ASL	R3	;MULTIPLY BY 2
8762	044414	016305	001524			MOV	PBLKT(R3), R5	;STORE PARAMETER BLOCK ADDRESS
8763	044420	016237	000000	001410		MOV	RKCS1(R2), T.CS1	;LOAD TEMPORARY CS1 FOR STATUS REPORT
8764	044426	000137	044602			JMP	I.ERRC	;REPORT ERROR
8765								
8766	044432			75:				
8767								
8768	044432	013705	001464	I.100:		MOV	0.WAIT, R5	;LOAD PARAMETER BLOCK ADDRESS INTO R5
8769	044436	001002				BNE	25	;IS COMMAND WAITING PROCESSING
8770								;YES, DO PROCESSING
8771	044440	000137	045420			JMP	I.ATTN	;NO, PROCESS ATTENTION
8772								
8773	044444	013704	001412	25:		MOV	T.CS2, R4	;STORE RKCS2 FOR DRIVE NUMBER
8774	044450	042704	177770			BIC	#↑C<DRVMSK>, R4	;MASK OUT UNNECESSARY BITS
8775								
8776								
8777	044454	010403				MOV	R4, R3	;STORE DRIVE NUMBER FOR INDEX
8778	044456	006303				ASL	R3	;MULTIPLY BY 2 FOR INDEX
8779								
8780	044460	126504	000000			CMPB	P.DRVN(R5), R4	;CHECK IF DRIVE NUMBER IS EXPECTED
8781	044464	001401				BEQ	35	;YES, CONTINUE
8782	044466	000000				HALT		;NO, DRIVER ERROR
8783	044470	122765	000164	000001	35:	CMPB	#RDALHD, P.CMND(R5)	;CHECK IF READ ALL HEADERS
8784	044476	001002				BNE	105	;NO, EXECUTE NORMAL DATA TRANSFER
8785	044500	000137	045760			JMP	I.HDAL	;GO EXECUTE SPECIAL HEADER SEQUENCE
8786								
8787	044504	005037	001464	105:		CLR	0.WAIT	;CLEAR WAIT FOR COMMAND COMPLETION
8788	044510	005063	001544			CLR	W.DRV(R3)	;CLEAR WATCH-DOG TIME
8789	044514	146437	001514	001512		BICB	INTMSK(R4), W.TIME	;RESET TIMING ON THIS DRIVE
8790	044522	016237	000000	001410		MOV	RKCS1(R2), T.CS1	;STORE COMMAND AND STATUS REGISTER 1
8791	044530	032737	100000	001410		BIT	#CERR, T.CS1	;CHECK IF CONTROLLER ERROR
8792	044536	001021				BNE	I.ERRC	;YES, PROCESS ERROR
8793	044540	016237	000016	001424		MOV	RKASOF(R2), T.ASOF	;STORE ATTENTION SUMMARY
8794	044546	136437	001514	001425		BITB	INTMSK(R4), T.ASOF+1	;CHECK IF DRIVE ATTENTION SET
8795	044554	001004				BNE	155	;YES, REPORT ERROR
8796	044556	004777	134672			JSR	PC.2A.NORM	;INDICATE NORMAL RETURN
8797	044562	000137	046674			JMP	I.RTRN	;RESTORE REGISTERS
8798								
8799	044566	052765	000010	000014	155:	BIS	#UJEXATT, P.PRST(R5)	;SET UNEXPECTED ATTENTION

```

8800
8801 044574 004737 047364 I.ERRA: JSR PC,I.CSTS ;STORE CONTROLLER STATUS
8802 044600 000405 BR I.ERR ;STORE PATTERN AND POSITION INFORMATION
8803
8804 044602 013765 001410 000016 I.ERRC: MOV T.CS1,P.CS1(R5) ;GET ERROR RKCS1
8805 044610 004737 047406 JSR PC,I.CST1 ;GET REST OF CONTROLLER STATUS
8806 044614 016265 000032 000062 I.ERR: MOV RKECPT(R2),P.EPAT(R5) ;STORE ECC PATTERN
8807 044622 016265 000030 000060 MOV RKECPS(R2),P.EPOS(R5) ;STORE ECC POSITION
8808 044630 004037 046712 JSR RO,I.CCLR ;CLEAR CONTROLLER
8809 044634 046674 I.RTRN ;ERROR RETURN
8810 044636 032765 010400 000020 BIT #NED!UFE,P.CS2(R5) ;CHECK IF IT WAS NON-EXISTENT DRIVE OR
8811 ; UNIT FIELD ERROR
8812 044644 001056 BNE 5$ ;YES, REPORT ERROR
8813 044646 032765 000001 000036 BIT #DRA,P.DS(R5) ;SEE IF DRIVE IS PVAİL
8814 044654 001004 BNE 4$ ;BR IF YES
8815 044656 052765 010000 000014 BIS #DRVSZD,P.PRST(R5) ;SET "DRIVE SIEZED" BIT
8816 044664 000471 BR 10$ ;REPORT ERROR
8817 044666 004037 047470 4$: JSR RO,I.STAT ;GATHER DRIVE STATUS
8818 044672 046674 I.RTRN ;ERROR RETURN
8819 044674 112737 000005 001410 MOV# DR.CLR,T.CS1 ;LOAD COMMAND
8820 044702 004037 046774 JSR RO,I.ISSU ;ISSUE DRIVE CLEAR
8821 044706 046674 I.RTRN ;ERROR RETURN
8822 044710 136437 001514 001425 BIT# INTMSK(R4),T.ASO# +1 ;CHECK IF ATTENTION RESET
8823 044716 001407 BEQ 2$ ;NO, INDICATE DRIVE ERROR
8824 044720 052737 000020 001462 BIS #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
8825 ; WITH CLEAR
8826 044726 004777 134526 JSR PC,JA.CONT ;REPORT CONTROLLER ERROR
8827 044732 000137 046674 JMP I.RTRN ;GO RESTORE REGISTERS
8828
8829 044736 032737 040000 001434 2$: BIT #S.DSC,T.MR2 ;CHECK IF DRIVE STATUS CHANGE CLEARED
8830 044744 001403 BEQ 3$ ;YES, CHECK FAULT
8831 044746 052765 000040 000014 BIS #DRVDSC,P.PRST(R5) ;SET DSC DID NOT CLEAR
8832 044754 032737 001000 001436 3$: BIT #S.PAR,T.MR3 ;CHECK IF DRIVE PARITY ERROR
8833 044762 001407 BEQ 5$ ;NO, INDICATE ABNORMAL TERMINATION
8834 044764 052737 002000 001462 BIS #E.DPAR,E.CONT ;SET DRIVE PARITY ERROR
8835 044772 004777 134462 JSR PC,JA.CONT ;INDICATE CONTROLLER ERROR
8836 044776 000137 046674 JMP I.RTRN ;RETURN
8837
8838 045002 032765 000020 000014 5$: BIT #DRVHRD,P.PRST(R5) ;CHECK IF HARD DRIVE ERROR
8839 045010 001017 BNE 10$ ;YES, GO REPORT ERROR
8840 045012 032737 020000 001434 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS CYCLING DOWN
8841 045020 001413 BEQ 10$ ;NO, REPORT ERROR
8842 045022 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING
8843 045030 156437 001514 001512 BIS# INTMSK(R4),W.TIME ;SET UP 8 SECONDS FOR
8844 045036 013763 001474 001544 MOV W.8SEC,W.DRV(R3) ;DRIVE TO CYCLE UP
8845 045044 000137 046674 JMP I.RTRN ;GO RESTORE REGISTERS
8846
8847 045050 004777 134402 10$: JSR PC,JA.ABNL ;GO REPORT ERROR
8848 045054 000137 046674 JMP I.RTRN ;GO RESTORE REGISTERS
8849
8850 .SBTTL *READ ALL HEADERS INTERRUPT SEQUENCE
8851
8852 045060 016237 000000 001410 I.HDAL: MOV RKCS1(R2),T.CS1 ;STORE CS1 TO CHECK CONTROLLER
8853 ; ERROR
8854 045066 032737 100000 001410 BIT #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR
8855 045074 001423 BEQ 5$ ;NO, CHECK FOR ATTENTION

```



```

8856
8857 045076 005037 001464          CLR      0.WAIT          ;CLEAR WAITING FOR COMMAND COMPLETE
8858 045102 146437 001514 001512  BICB    INTMSK(R4),W.TIME ;RESET TIMING ON DRIVE
8859 045110 005063 001544          CLR      W.DRV(R3)      ;CLEAR TIME OUT COUNT
8860 045114 013765 001410 000016  MOV     T.CS1,P.CS1(R5) ;STORE ERROR RKCS1
8861 045122 004737 047406          JSR     PC,I.CS1        ;STORE CONTROLLER REGISTERS
8862 045126 004037 046712          JSR     RD,I.CCLR       ;CLEAR CONTROLFR
8863 045132 046674          I.RTRN ;ERROR RETURN
8864 045134 004777 134316          JSR     PC,JA.ABNL      ;INDICATE ERROR RETURN
8865 045140 000137 046674          JMP     I.RTRN         ;RESTORE REGISTERS
8866
8867 045144 016237 000016 001424 5$:  MOV     RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
8868 045152 136437 001514 001425  BITB    INTMSK(R4),T.ASOF+1 ;CHECK IF DRIVE ATTENTION IS SET
8869 045160 001411          BEQ     7$            ;NO, CHECK IF READ ALL HEADERS
8870 045162 005037 001464          CLR      0.WAIT          ;CLEAR WAITING FOR COMMAND COMPLETION
8871 045166 146437 001514 001512  BICB    INTMSK(R4),W.TIME ;RESET TIMING ON DRIVE
8872 045174 005063 001544          CLR      W.DRV(R3)      ;CLEAR TIME OUT COUNT
8873 045200 000137 044574          JMP     I.ERRA        ;GO REPORT ERROR
8874
8875 045204 013701 001502          7$:  MOV     HDR.AD,R1        ;GET MAIN MEMORY ADDRESS
8876 045210 016221 000024          MOV     RKDB(R2),(R1)+  ;GET FIRST WORD OF HEADER
8877 045214 016221 000024          MOV     RKDB(R2),(R1)+  ;GET SECOND WORD OF HEADER
8878 045220 016221 000024          MOV     RKDB(R2),(R1)+  ;GET THIRD WORD OF HEADER
8879 045224 010137 001502          MOV     R1,HDR.AD      ;STORE ADDRESS FOR NEXT HEADER
8880 045230 016237 000010 001412  MOV     RKCS2(R2),T.CS2 ;STORE CS2 TO CHECK FOR DATA LATE
8881 045236 032737 100000 001412  BIT     #DLT,T.CS2      ;CHECK FOR DATA LATE
8882 045244 001056          BNE     35$          ;YES, REPORT ERROR
8883 045246 005337 001504          DEC     HDR.CT        ;DECREMENT NUMBER OF HEADER YET TO READ
8884 045252 001027          BNE     25$          ;IF NON-ZERO, GO ISSUE NEXT READ HEADER
8885 045254 005037 001464          CLR      0.WAIT          ;CLEAR DRIVER WAITING FOR COMMAND COMPLETION
8886 045260 005063 001544          CLR      W.DRV(R3)      ;CLEAR TIME OUT COUNT FOR THIS DRIVE
8887 045264 146437 001514 001512  BICB    INTMSK(R4),W.TIME ;CLEAR WATCH DOG TIME ON THIS DRIVE
8888 045272 012762 000003 000026  MOV     #3,RKMRI(R2)   ;LOAD MAINTENANCE REGISTER FOR SECTOR COUNT
8889 045300 112737 000001 001410  MOVB    #DR.SEL,T.CS1  ;LOAD SELECT COMMAND
8890 045306 004037 046774          JSR     RD,I.ISSU      ;GET SECTOR COUNT
8891 045312 046674          I.RTRN ;ERROR RETURN
8892 045314 013765 001436 000056  MOV     T.MR3,P.B11(R5) ;LOAD SECTOR COUNT
8893 045322 004777 134126          JSR     PC,JA.NORM     ;INDICATE NORMAL TERMINATION
8894 045326 000137 046674          JMP     I.RTRN         ;RESTORE REGISTER;
8895
8896 045332 016562 000002 000020 25$:  MOV     P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
8897 045340 016562 000004 000006  MOV     P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
8898 045346 116565 000007 000017  MOVB    P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
8899 045354 042765 165777 000016  BIC     #1C<CDT!CFMT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT AND
8900                                     ;DRIVE TYPE
8901 045362 112765 000125 000016  MOVB    #RDHEAD,P.CS1(R5) ;STORE COMMAND ISSUED
8902 045370 016562 000016 000000  MOV     P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
8903 045376 000137 046674          JMP     I.RTRN         ;RESTORE REGISTERS
8904
8905 045402 052737 000400 001462 35$:  BIS     #E.DLT,E.CONT   ;SET DATA LATE WHILE UNLOADING HEADER
8906 045410 004777 134044          JSR     PC,JA.CONT     ;REPORT ERROR
8907 045414 000137 046674          JMP     I.RTRN         ;RESTORE REGISTERS
8908
8909                                     .SBTTL  *DRIVE ATTENTION SCANNER
8910
8911 045420 016237 000000 001410  I.ATTN: MOV    RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS

```

```

8912                                     ; REGISTER 1 FOR COMPARISON
8913 045426 032737 100000 001410      BIT   #CERR,T.CS1      ;CHECK IF CONTROLLER ERROR OCCURRED
8914 045434 001444                                     BEQ   SS              ;NO, CHECK IF ATTENTION
8915                                     ;CHECK IF ANY DATA TRANSFER TYPE ERROR EXISTS
8916                                     BIT   #DLT!WCE!UPE!NEM,T.CS2
8917 045436 032737 164000 001412
8918
8919 045444 001007      BNE   1$              ;INDICATE ERROR
8920 045446 016237 000014 001426      MOV   RKER(R2),T.ER  ;STORE ERROR REGISTER
8921
8922                                     ; CHECK FOR DATA TRANSFER ERROR TYPE
8923 045454 032737 125700 001426      BIT   #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
8924
8925 045462 001407      BEQ   2$              ;NO DATA TRANSFER ERROR
8926
8927 045464 052737 000010 001462 1$:   BIS   #E.UDAT,E.CONT ;SET UNEXPECTED DATA TYPE ERROR
8928 045472 004777 133762      JSR   PC,QA.CONT    ;REPORT ERROR
8929 045476 000137 046674      JMP   I.ATRN        ;RESTORE REGISTERS
8930
8931 045502 013704 001412      2$:   MOV   T.CS2,R4      ;SAVE CS2 FOR REGISTER NUMBER
8932 045506 042704 177770      BIC   #1C<DRVMSK>,R4 ;STRIP OFF JUNK
8933 045512 010403      MOV   R4,R3         ;STORE DRIVE NUMBER IN R3
8934 045514 006303      ASL   R3            ;MULTIPLY DRIVE NUMBER BY 2
8935 045516 146437 001514 001512      BICB  INTMSK(R4),W.TIME ;CLEAR WATCH-DOG TIMER FOR DRIVE
8936 045524 005063 001544      CLR   W.DRV(R3)     ;RESET WATCH DOG TIME
8937 045530 016305 001524      MOV   PBLKT(R3),R5  ;STORE PARAMETER BLOCK ADDRESS IN R5
8938
8939                                     ;
8940                                     ; CLEAR DRIVE POSITIONING AND DRIVE POSITIONED FOR DATA TRANSFER
8941 045534 042765 000006 000014      BIC   #DRVPOS!DRVPDT,P.PRST(R5)
8942
8943 045542 000137 044602      JMP   I.ERRC        ;GO REPORT ERROR
8944
8945 045546 032737 040000 001410 5$:   BIT   #DI,T.CS1      ;CHECK IF ANY DRIVE ATTENTION
8946 045554 001004      BNE   6$              ;YES, PROCESS INTERRUPT
8947 045556 004737 051740      JSR   PC,C.OPT      ;CALL COMMAND OPTIMIZER
8948 045562 000137 046674      JMP   I.ATRN        ;RESTORE REGISTERS
8949
8950 045566 016237 000016 001424 6$:   MOV   RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
8951 045574 105737 001425      TSTB  T.ASOF+1      ;CHECK IF ANY ATTENTIONS SET
8952 045600 001007      BNE   7$              ;YES GO PROCESS INTERRUPT
8953 045602 052737 000002 001462      BIS   #E.NOAT,E.CONT ;SET NO ATTENTION IN ATTENTION SUMMARY
8954 045610 004777 133644      JSR   PC,QA.CONT    ;GO REPORT ERROR
8955 045614 000137 046674      JMP   I.ATRN        ;GO RESTORE REGISTERS
8956
8957 045620 005004      7$:   CLR   R4            ;CLEAR DRIVE NUMBER
8958 045622 136437 001514 001425 8$:   BITB  INTMSK(R4),T.ASOF+1 ;CHECK IF ATTENTION OF THIS DRIVE
8959 045630 001002      BNE   9$              ;YES, PROCESS INTERRUPT
8960 045632 005204      INC   R4            ;INCREMENT DRIVE NUMBER
8961 045634 000772      BR    8$            ;CHECK ATTENTION ON NEXT DRIVE
8962
8963 045636 010403      9$:   MOV   R4,R3         ;STORE DRIVE NUMBER
8964 045640 006303      ASL   R3            ;MULTIPLY DRIVE NUMBER BY 2
8965 045642 016305 001524      MOV   PBLKT(R3),R5  ;STORE PARAMETER BLOCK ADDRESS
8966 045646 032765 010000 000014      BIT   #DRVSZD,P.PRST(R5) ;CHECK IF DRIVE WAS SEIZED
8967 045654 001402      BEQ   10$           ;NO, PROCESS NORMAL ATTENTION

```

```

8968 045656 000137 046506          JMP      I.DUAL          ;PROCESS THE RELEASE OF DRIVE
8969
8970 045662          10$:
8971 045662 032765 020000 000014          BIT      #E.UNLD,P.PRST(R5) ;CHECK IF DRIVE UNLOADING
8972 045670 001402          BEQ     11$              ;NO CONTINUE
8973 045672 000137 046402          JMP     I.UNLD           ;SERVICE DRIVE IN POSITION AFTER ERROR
8974
8975 045676 042765 000002 000014 11$:          BIC     #DRVPOS,P.PRST(R5) ;RESET DRIVE POSITIONING
8976 045704 005063 000026          CLR     RKMRI(R2)       ;CLEAR MAINTENANCE REGISTER 1
8977 045710 112737 000001 001410          MOVB   #DR.SEL,T.CS1    ;LOAD COMMAND
8978 045716 004037 046774          JSR    RO,I.ISSU        ;SELECT DRIVE WITH ATTENTION HIGH
8979 045722 046674          I.RTRN                  ;ERROR RETURN
8980 045724 013765 001436 000042          MOV     T.MR3,P.B00(R5) ;STORE STATUS BYTE 00 MESS B
8981 045732 032765 000200 000042          BIT     #S.FLT,P.B00(R5) ;CHECK IF DRIVE FAULT
8982 045740 001401          BEQ     12$              ;NO CHECK FOR DRIVE STATUS CHANGE
8983 045742 000473          BR     I.AERR           ;PROCESS ERROR
8984
8985 045744 013765 001434 000040 12$:          MOV     T.MR2,P.A00(R5) ;STORE MAINTENANCE REGISTER 2
8986 045752 032765 040000 000040          BIT     #S.DSC,P.A00(R5) ;CHECK FOR DRIVE STATUS CHANGE
8987 045760 001004          BNE    13$              ;YES PROCESS DRIVE STATUS CHANGE
8988 045762 052765 004000 000014          BIS    #NODSC,P.PRST(R5) ;SET NO DRIVE STATUS CHANGE
8989 045770 000460          BR     I.AERR           ;PROCESS ERROR
8990
8991 045772 112737 000005 001410 13$:          MOVB   #DR.CLR,T.CS1    ;LOAD COMMAND
8992 046000 004037 046774          JSR    RO,I.ISSU        ;CLEAR DRIVE STATUS CHANGE
8993 046004 046674          I.RTRN                  ;ERROR RETURN
8994 046006 013765 001424 000032          MOV     T.ASOF,P.ASOF(R5) ;STORE ATTENTION SUMMARY
8995 046014 136465 001514 000033          BITB   INTMSK(R4),P.ASOF+1(R5) ;CHECK IF ATTENTION RESET
8996 046022 001407          BEQ     15$              ;YES CONTINUE INTERRUPT PROCESSING
8997 046024 052737 000020 001462          BIS    #E.CLAT,E.CONT   ;SET ATTENTION DID NOT RESET
8998                                     ; WITH DRIVE CLEAR
8999          JSR    PC,QA.CONT     ;FLAG ERROR
9000          JMP     I.RTRN        ;RESTORE REGISTERS
9001
9002 046042 013765 001434 000040 15$:          MOV     T.MR2,P.A00(R5) ;STORE MAINTENANCE REGISTER 2
9003 046050 032765 040000 000040          BIT     #S.DSC,P.A00(R5) ;CHECK IF DRIVE STATUS CHANGE
9004                                     ; RESET
9005          BEQ     16$              ;YES CONTINUE INTERRUPT PROCESSING
9006 046060 052765 000040 000014          BIS    #DRVDSC,P.PRST(R5) ;SET DRIVE STATUS CHANGE DID NOT CLEAR
9007 046066 000421          BR     I.AERR           ;GO PROCESS ERROR
9008
9009 046070 146437 001514 001512 16$:          BICB   INTMSK(R4),W.TIME ;RESET TIMING ON THIS DRIVE
9010 046076 005063 001514          CLR     W.DRV(R3)       ;CLEAR DRIVE TIMING COUNT
9011 046102 032765 000004 000014          BIT     #DRVPDT,P.PRST(R5) ;CHECK IF DRIVE IS POSITIONING
9012                                     ; FOR DATA TRANSFER
9013          BNE    17$              ;YES CALL COMMAND OPTIMIZER
9014 046112 004777 133336          JSR    PC,QA.NORM       ;REPORT SUCCESSFUL COMMAND
9015                                     ; COMPLETION
9016          JMP     I.RTRN        ;RESTORE REGISTERS
9017
9018 046122 004737 051740          17$:   JSR    PC,C.OPT        ;CALL COMMAND OPTIMIZER
9019 046126 000137 046674          JMP     I.RTRN        ;RESTORE REGISTERS
9020
9021          .SBTTL  *ATTENTION ERROR HANDLER
9022
9023 046132 042765 000004 000014 I.AERR: BIC     #DRVPDT,P.PRST(R5) ;RESET POSITIONING IN PROGRESS BECAUSE

```

```

9024      046140 146437 001514 001512      BICB   INTMSK(R4),W.TIME ; OF DATA TRANSFER
9025      046146 005063 001544      CLR    W.DRV(R3)       ; CLEAR TIMING FOR THIS DRIVE
9026      046152 042765 177741 000016      BIC    #177741,P.CS1(R5) ; RESET WATCH-DOG TIME
9027      046160 042737 000036 001410      BIC    #36,T.CS1       ; KEEP COMMAND ISSUED
9028      046166 053765 001410 000016      BIS    T.CS1,P.CS1(R5) ; KEEP CURRENT CONTROLLER STATUS
9029      046174 013765 001412 000020      MOV    T.CS2,P.CS2(R5) ; MAKE GOOD MESSAGE
9030      046202 013765 001414 000022      MOV    T.WCR,P.WCR(R5) ; STORE CONTROLLER REGISTERS
9031      046210 013765 001416 000024      MOV    T.BA,P.BAR(R5)
9032      046216 013765 001420 000026      MOV    T.DA,P.DTS(R5)
9033      046224 013765 001422 000030      MOV    T.DC,P.DCYL(R5)
9034      046232 013765 001424 000032      MOV    T.ASOF,P.ASOF(R5)
9035      046240 013765 001426 000034      MOV    T.ER,P.ER(R5)
9036      046246 013765 001430 000036      MOV    T.DS,P.DS(R5)
9037      046254 004037 047470      JSR    RD,I.STAT      ; GATHER DRIVE STATUS
9038      046260 046674      I.RTRN ; ERROR RETURN
9039      046262 112737 000005 001410      MOVB   #DR.CLR,T.CS1 ; LOAD COMMAND
9040      046270 004037 046774      JSR    RD,I.ISSU     ; CLEAR DRIVE ERRORS
9041      046274 046674      I.RTRN ; ERROR RETURN
9042      046276 136437 001514 001425      BITB   INTMSK(R4),T.ASOF+1 ; CHECK IF ATTENTION RESET
9043      046304 001407      BEQ    2$           ; YES, FLAG DRIVE ERROR
9044      046306 052737 000020 001462      BIS    #E.CLAT,E.CONT ; SET ATTENTION DID NOT RESET
9045      046314 004777 133140      JSR    PC,QA.CONT   ; REPORT ERROR
9046      046320 000137 046674      JMP    I.RTRN      ; RESTORE REGISTERS
9047
9048
9049      046324 032765 000020 000014 2$:      BIT    #DRVHRD,P.PRST(R5) ; CHECK IF AHARD DRIVE ERROR
9050      046332 001017      BNE    10$         ; YES, REPORT ERROR
9051      046334 032737 020000 001434      BIT    #S.PIP,T.MR2  ; CHECK IF DRIVE IS UNLOADING
9052      046342 001413      BEQ    10$         ; NO, REPORT ERROR
9053      046344 052765 020000 000014      BIS    #E.UNLD,P.PRST(R5) ; SET DRIVE UNLOADING DUE TO ERROR
9054      046352 156437 001514 001512      BISB  INTMSK(R4),W.TIME ; SET TIMING ON THIS DRIVE
9055      046360 013763 001474 001544      MOV    W.BSEC,W.DRV(R3) ; LOAD 8 SECONDS FOR CYCLE UP
9056      046366 000137 046674      JMP    I.RTRN      ; RESTORE REGISTERS
9057
9058      046372 004777 133060      10$:   JSR    PC,QA.ABNL   ; REPORT ERROR
9059      046376 000137 046674      JMP    I.RTRN      ; RESTORE REGISTERS
9060
9061      .SBTTL *ERROR CAUSING DRIVE TO UNLOAD
9062
9063      046402 052765 020000 000014 I.UNLD: BIS    #E.UNLD,P.PRST(R5) ; CLEAR DRIVE UNLOADING BECAUSE OF ERROR
9064      046410 112737 000005 001410      MOVB   #DR.CLR,T.CS1 ; LOAD IN DRIVE CLEAR
9065      046416 004037 046774      JSR    RD,I.ISSU     ; GO ISSUE DRIVE CLEAR
9066      046422 046674      I.RTRN ; ERROR RETURN
9067      046424 136437 001514 001425      BITB   INTMSK(R4),T.ASOF+1 ; CHECK IF ATTENTION CLEARED
9068      046432 001406      BEQ    15$         ; YES, CONTINUE
9069      046434 012737 000020 001462      MOV    #E.CLAT,E.CONT ; SET ATTENTION DID NOT RESET
9070      046442 004777 133012      JSR    PC,QA.CONT   ; REPORT ERROR
9071      046446 000512      BR     I.RTRN      ; RESTORE REGISTERS
9072
9073      046450 032737 040000 001434 15$:   BIT    #S.DSC,T.MR2  ; CHECK IF DRIVE STAUUS CHANGE RESET
9074      046456 001403      BEQ    20$         ; YES, CONTINUE
9075      046460 052765 000040 000014      BIS    #DRVDSO,P.PRST(R5) ; SET DRIVE STAUUS CHANGE DID NOT CLEAR
9076      046466 146437 001514 001512 20$:   BICB  INTMSK(R4),W.TIME ; RESET TIMING ON THIS DRIVE
9077      046474 005063 001544      CLR    W.DRV(R3)    ; CLEAR TIME COUNT
9078      046500 004777 132752      JSR    PC,QA.ABNL   ; REPORT ERROR
9079      046504 000473      BR     I.RTRN      ; RESTORE REGISTER

```

```

9080
9081      .SBTTL      *DUAL ACCESS INTERRUPT HANDLER
9082
9083 046506 112737 000001 001410 I DUAL: MOVB      #DR.SEL,T.CS1 ;LOAD COMMAND
9084 046514 004037 046774          JSR        RO,I.ISSU ;SELECT DRIVE
9085 046520 046674          I.RTRN      ;ERROR RETL:PN
9086 046522 032737 000200 001436 BIT        #S.FLT,T.MR3 ;CHECK IF DRIVE FAULT
9087 046530 001402          BEQ        10$      ;NO, PROCESS INTERRUPT
9088 046532 000137 046132          JMP        I.AERR   ;PROCESS ATTENTION ERROR
9089
9090 046536 032737 040000 001434 10$: BIT        #S.DSC,T.MR2 ;CHECK IF DRIVE STATUS CHANGE SET
9091 046544 001005          BNE        11$      ;YES, PROCESS INTERRUPT
9092 046546 052765 004000 000014 BIS        #NODSC,P.PRST(R5) ;SET NO DRIVE STATUS CHANGE
9093 046554 000137 046132          JMP        I.AERR   ;PROCESS ATTENTION ERROR
9094
9095 046560 042765 010000 000014 11$: BIC        #DRVSZD,P.PRST(R5) ;RESET DRIVE SEIZED
9096 046566 112737 000005 001410 MOVB      #DR.CLR,T.CS1 ;LOAD COMMAND
9097 046574 004037 046774          JSR        RO,I.ISSU ;CLEAR DRIVE DSC
9098 046600 046674          I.RTRN      ;ERROR RETURN
9099 046602 136437 001514 001425 BITB      INTMSK(R4),T.ASO+1 ;CHECK IF ATTENTION RESET
9100 046610 001406          BEQ        15$      ;YES, CONTINUE
9101 046612 052737 000020 001462 BIS        #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
9102 046620 004777 132634          JSR        PC,QA.CONT ;INDICATE CONTROLLER ERROR
9103 046624 000423          BR         I.ATRN  ;RESTORE REGISTERS
9104
9105 J46626 032737 040000 001434 15$: BIT        #S.DSC,T.MR2 ;CHECK IF DRIVE STATUS CHANGE IS STILL SET
9106 046634 001405          BEQ        20$      ;NO, GO ENQUEUE COMMAND
9107 046636 052765 000040 000014 BIS        #DRVDSC,P.PRST(R5) ;SET DRIVE STATUS CHANGE DID NOT CLEAR
9108 046644 000137 046132          JMP        I.AERR   ;REPORT ATTENTION ERROR
9109
9110 046650 146437 001514 001512 20$: BICB      INTMSK(R4),W.TIME ;STOP TIMING ON THIS DRIVE
9111 046656 005063 001544          CLR        W.DRV(R3) ;CLEAR WATCH DOG TIME OF THIS DRIVE
9112 046662 004037 051406          JSR        RO,Q.PUSH ;PUT PARAMETER BLOCK ADDRESS ON
9113 046666 001324          CINITQ      ;COMMAND INITATION QUEUE
9114 046670 004737 051740          JSR        PC,C.OPT ;CALL COMMAND OPTIMIZER
9115
9116 046674 012600          I.RTRN: MOV      (SP)+,R0 ;RESTORE R0
9117 046676 012601          MOV      (SP)+,R1 ;RESTORE R1
9118 046700 012602          MOV      (SP)+,R2 ;RESTORE R2
9119 046702 012603          MOV      (SP)+,R3 ;RESTORE R3
9120 046704 012604          MOV      (SP)+,R4 ;RESTORE R4
9121 046706 012605          MOV      (SP)+,R5 ;RESTORE R5
9122 046710 000002          RTI         ;RETURN
9123

```

9124
9125
9126
9127
9128
9129
9130
9131
9132
9133
9134
9135
9136
9137
9138
9139
9140
9141
9142
9143
9144
9145
9146
9147
9148
9149
9150
9151
9152
9153
9154
9155
9156
9157
9158

.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      R0,I.CCLR
* <ADDRESS OF ERROR RETURN>
* RETURN
*
*****

```

```

9145 046712 012762 100000 000000 I.CCLR: MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
9146 046720 016237 000000 001410      MOV      RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS REGISTER 1
9147 046726 032737 100000 001410      BIT      #CEPR,T.CS1 ;CHECK IF CONTROLLER CLEAR DID
9148                                ; CLEAR ERROR
9149 046734 001407                                BEQ      S$ ;YES, RETURN TO DRIVER PROCESSING
9150 046736 052737 000001 001462      BIS      #E.CCLR,E.CONT ;SET CLEAR CONTROLLER DID NOT CLEAR ERROR
9151 046744 004777 132510      JSR      PC,QA.CONT ;REPORT CONTROLLER ERROR
9152 046750 011000      MOV      (R0),R0 ;SET UP ERROR RETURN
9153 046752 000200      RTS      R0 ;RETURN
9154
9155 046754 012762 000100 000000 S$:  MOV      #IE,RKCS1(R2) ;SET INTERRUPT ENABLE
9156 046762 112737 177777 001506      MOVB     #-1,I.ISRL ;SET INTERRUPT ENABLE ISSUED
9157 046770 005720      TST     (R0)+ ;ADJUST FOR NORMAL RETURN
9158 046772 000200      RTS      R0 ;RETURN

```

K16

.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      RD,I.ISSU
*      <ADDRESS OF ERROR RETURN>
*      RETURN
*
* ROUTINES USED:
* -----
*
*          I.CCLR
*          I.STOR
*****

```

9159
9160
9161
9162
9163
9164
9165
9166
9167
9168
9169
9170
9171
9172
9173
9174
9175
9176
9177
9178
9179
9180
9181
9182
9183
9184
9185
9186
9187
9188
9189
9190
9191
9192
9193
9194
9195
9196
9197
9198
9199
9200
9201
9202
9203
9204
9205
9206
9207
9208
9209
9210
9211
9212
9213
9214

```

046774 013746 001410
047000 005037 001412
047004 116537 000000 001412
047012 013762 001412 000010
047020 116537 000007 001411
047026 142737 177753 001411
047034 013762 001410 000000
047042 105762 000000
047046 100375
047050 004737 047236
047054 032737 100000 001410
047062 001447
047064 032737 001000 001412
047072 001406
047074 052737 100000 001462
047102 004777 132352
047106 000450
047110 022737 000001 001500
047116 001004
047120 032737 010000 001412
047126 001022
047130 032737 024000 001410
047136 001027
047140 032737 176400 001412
047146 001023

```

```

I.ISSU: MOV      T.CS1,-(SP)      ;STORE COMMAND ISSUED
        CLR      T.CS2          ;CLEAR TEMPORARY CS2
        MOVVB   P.DRVN(R5),T.CS2 ;LOAD IN DRIVE NUMBER
        MOV      T.CS2,RKCS2(R2) ;LOAD DRIVE NUMBER FOR COMMAND
        MOVVB   P.CS1A(R5),T.CS1+1 ;STORE BITS 8-15 OF CS1
        SICB    #T.CB.CDT!B.CFMT>,T.CS1+1 ;CLEAR ALL BITS EXCEPT
        ;FORMAT AND DRIVE TYPE
1$:     MOV      T.CS1,RKCS1(R2) ;ISSUE COMMAND
        TSTB    RKCS1(R2)      ;WAIT FOR READY
        BPL     1$
        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,QA.CONT     ;REPORT CONTROLLER ERROR
        BR      10$           ;RETURN
;CHECK IF ANY CONTROLLER ERROR IS SET
2$:     CMP     #1,OPTFLG      ;SEE IF CURRENTLY IN CMND OPTIMIZER
        BNE     4$            ;BR IF NOT
        BIT     #NED,T.CS2    ;SEE IF NED ERROR SET
        BNE     3$            ;BR IF YES, TO ISSUE CLEAR AND EXIT
4$:     BIT     #CTO!SPAR,T.CS1
        BNE     7$
        BIT     #UFE!PGE!NEM!NED!UPE!WCE!DLT,T.CS2
        BNE     7$

```

9215	047150	032737	131761	001426		BIT	#ILC!DTYE!FMTE!ECH!BSE!HVRC!COE!DTE!OPI!DCK,T.ER
9216	047156	001017				BNE	7\$
9217							
9218	047160	122716	000005			CMPB	#DR.CLR,(SP) ;CHECK IF CLEAR DRIVE
9219	047164	001003				BNE	3\$;NO, DO NOT SET DRIVE HARD ERROR
9220	047166	052765	000020	000014		BIS	#DRVHRD,P.PRST(R5) ;SET HARD DRIVE ERROR
9221	047174	004037	046712		3\$:	JSR	RO,I.CCLR ;GO ISSUE A CONTROLLER CLEAR
9222	047200	047230				'0\$;ERROR RETURN
9223	047202	012762	000100	000000	5\$:	MOV	#IE,RKCS1(R2) ;SET INTERRUPT ENABLE
9224	047210	005726				TST	(SP)+ ;ADJUST STACK
9225	047212	005720				TST	(RO)+ ;ADJUST RO FOR NORMAL RETURN
9226	047214	000200				RTS	RO ;RETURN
9227							
9228	047216	052737	001000	001462	7\$:	BIS	#E.CERR,E.CONT ;SET CONTROLLER ERROR DURING
9229							; DRIVER SERVICING
9230	047224	004777	132230			JSR	PC,DA.CONT ;REPORT ERROR
9231	047230	005726			10\$:	TST	(SP)+ ;ADJUST STACK
9232	047232	011000				MOV	(RO),RO ;ADJUST RO FOR ERROR RETURN
9233	047234	000200				RTS	RO ;RETURN

9234
9235
9236
9237
9238
9239
9240
9241
9242
9243
9244
9245
9246
9247
9248
9249
9250
9251
9252
9253
9254
9255
9256
9257
9258
9259
9260
9261
9262
9263
9264
9265

047236 016237 000000 001410
047244 016237 000010 001412
047252 016237 000002 001414
047260 016237 000004 001416
047266 016237 000006 001420
047274 016237 000012 001430
047302 016237 000014 001426
047310 016237 000016 001424
047316 016237 000020 001422
047324 016237 000026 001432
047332 016237 000034 001434
047340 016237 000036 001436
047346 016237 000030 001440
047354 016237 000032 001442
047362 000207

.SBTTL *STORE RK611 UNIBUS REGISTERS

```
*****
*
* THIS SUBROUTINE IS CALLED BY THE RK06/07 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
```

.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	RK06/07 BASE ADDRESS
R5	ADDRESS OF PARAMETER BLOCK

9266
9267
9268
9269
9270
9271
9272
9273
9274
9275
9276
9277
9278
9279
9280
9281
9282
9283
9284
9285
9286
9287
9288
9289
9290
9291
9292
9293
9294
9295
9296
9297
9298
9299
9300
9301
9302
9303
9304
9305
9306
9307
9308
9309
9310

```

I.CSTS: BIC #177741, P.CS1(R5) ; CLEAR ALL BITS EXCEPT FUNCTION
; OF LAST COMMAND ISSUED
          BIC #36, T.CS1 ; CLEAR FUNCTION OF CS1 STATUS
          BIS T.CS1, P.CS1(R5) ; GENERATE CS1 STATUS INFORMATION
I.CST1:  MOV RKCS2(R2), P.CS2(R5) ; STORE COMMAND AND STATUS REGISTER 2
          MOV RKWC(R2), P.WCR(R5) ; STORE WORD COUNT REGISTER
          MOV RKBA(R2), P.BAR(R5) ; STORE BUS ADDRESS REGISTER
          MOV RKDA(R2), P.DTS(R5) ; STORE DESIRED TRACK AND SECTOR
          MOV RKDS(R2), P.DS(R5) ; STORE DRIVE STATUS REGISTER
          MOV RKER(R2), P.ER(R5) ; STORE ERROR REGISTER
          MOV RKASOF(R2), P.ASOF(R5) ; STORE ATTENTION SUMMARY AND
; OFFSET
          MOV RKDCYL(R2), P.DCYL(R5) ; STORE CYLINDER ADDRESS
          RTS PC ; RETURN

```

```

9311
9312
9313
9314
9315
9316
9317
9318
9319
9320
9321
9322
9323
9324
9325
9326
9327
9328
9329
9330
9331
9332
9333
9334
9335
9336
9337 047470 012762 000001 000026
9338
9339 047476 112737 000001 001410
9340 047504 004037 046774
9341 047510 047700
9342 047512 013765 001434 000044
9343 047520 013765 001436 000046
9344 047526 012762 000002 000026
9345
9346 047534 112737 000001 001410
9347 047542 004037 046774
9348 047546 047700
9349 047550 013765 001434 000050
9350 047556 013765 001436 000052
9351 047564 012762 000003 000026
9352
9353 047572 112737 000001 001410
9354 047600 004037 046774
9355 047604 047700
9356 047606 013765 001434 000054
9357 047614 013765 001436 000056
9358 047622 005062 000026
9359
9360 047626 112737 000001 001410
9361 047634 004037 046774
9362 047640 047700
9363 047642 013765 001434 000040
9364 047650 013765 001436 000042
9365 047656 032737 001000 001436
9366 047664 001407

```

.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

```

I. STAT: MOV #1,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 01
MOV #DR.SEL,T.CS1 ;LOAD COMMAND
JSR RO,I.ISSU ;GET STATUS BYTES 01
3$ ;ERROR RETURN
MOV T.MR2,P.A01(R5) ;STORE STATUS BYTE 01 MESS A
MOV T.MR3,P.B01(R5) ;STORE STATUS BYTE 01 MESS B
MOV #2,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 10
MOV #DR.SEL,T.CS1 ;LOAD COMMAND
JSR RO,I.ISSU ;GET STATUS BYTES 10
3$ ;ERROR RETURN
MOV T.MR2,P.A10(R5) ;STORE STATUS BYTE 10 MESS A
MOV T.MR3,P.B10(R5) ;STORE STATUS BYTE 10 MESS B
MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 11
MOV #DR.SEL,T.CS1 ;LOAD COMMAND
JSR RO,I.ISSU ;GET STATUS BYTES 11
3$ ;ERROR RETURN
MOV T.MR2,P.A11(R5) ;STORE STATUS BYTE 11 MESS A
MOV T.MR3,P.B11(R5) ;STORE STATUS BYTE 11 MESS B
CLR RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 00
MOV #DR.SEL,T.CS1 ;LOAD COMMAND
JSR RO,I.ISSU ;GET STATUS BYTES 00
3$ ;ERROR RETURN
MOV T.MR2,P.A00(R5) ;STORE STATUS BYTE 00 MESS A
MOV T.MR3,P.B00(R5) ;STORE STATUS BYTE 00 MESS B
BIT #S.PAR,T.MR3 ;CHECK IF BAD PARITY DETECTED BY DRIVE
BEQ 5$ ;NO, RETURN NORMALLY

```

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 212
*GATHER DRIVE STATUS

SEQ 0210

9367	047666	052737	002000	001462		BIS	#E.DPAR,E.CONT	;INDICATE BAD PARITY DETECTED BY DRIVE
9368	047674	004777	131560			JSR	PC,DA,CONT	;REPORT ERROR
9369	047700	011000			3\$:	MOV	(R0),R0	;LOAD R0 FOR ERROR RETURN
9370	047702	000200				RTS	R0	;RETURN
9371								
9372	047704	052765	001000	000014	5\$:	BIS	#PBSVAL,P.PRST(R5)	;SET PARAMETER BLOCK STATUS VALID
9373	047712	005720				TST	(R0)+	;ADJUST R0 FOR NORMAL RETURN
9374	047714	000200				RTS	R0	;RETURN
9375								

9376
9377
9378
9379
9380
9381
9382
9383
9384
9385
9386
9387
9388
9389
9390
9391
9392
9393
9394
9395
9396
9397
9398
9399
9400
9401
9402
9403
9404
9405
9406
9407
9408
9409
9410
9411
9412
9413
9414
9415
9416
9417
9418
9419
9420
9421
9422
9423
9424
9425
9426
9427
9428
9429
9430
9431

.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
*     CONTROLLER CLEAR
*     SUBSYSTEM CLEAR
*     READ ALL DRIVE STATUS
*     READ SPECIFIED HEADER
*
* THE ABOVE COMMANDS ARE TRANSLATED INTO A SEQUENCE OF COMMANDS
*
*CALL JSR    PC.C.INIT
*      <ADDRESS OF PARAMETER BLOCK>
*      RETURN
*
* FOR THE SEQUENTIAL OPERATIONS, THE DRIVER WILL LOAD THE
* LOCATIONS, PBLKT AND INTMSK.
*
* ROUTINES USED:
*     W.WTCH
*     I.CSTS
*     I.STAT
*     I.CCLR
*
*****

```

```

047716 010546
047720 010446
047722 010346
047724 010246
047726 010146
047730 010046
047732 013746 177776
047736 013737 001452 177776
047744 017605 000016
047750 062766 000002 000016
047756 016504 000000
047762 042704 177770
047766 156437 001514 001512
047774 010403
047776 006303
050000 013763 001472 001544
050006 013702 001446

```

```

*****
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    #1<DRVMSK>,R4 ;MASK OUT JUNK
        BISB  INTMSK(R4),W.TIME ;SET WATCH DOG TIME
        MOV    R4,R3         ;STORE DRIVE NUMBER
        ASL   R3              ;MULTIPLY DRIVE NUMBER BY 2
        MOV    W.SEC,W.DRV(R3);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

```

F01

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 214
*COMMAND INITIATOR

SEQ 0212

```

9432 050012 042765 075176 000014      BIC      #1C<DRVUSE!W.WCF!NOCHK!DRPDRV!DTBAII>,P.PRST(R5)
9433
9434 050020 010500      MOV      R5,R0      ;STORE PARAMETER BLOCK ADDRESS
9435 050022 062700 000016      ADD      #P.CS1,R0    ;CALCULATE FIRST LOCATION TO BE CLEARED
9436 050026 010501      MOV      R5,R1      ;STORE PARAMETER BLOCK ADDRESS
9437 050030 062701 000062      ADD      #P.EPAT,R1  ;CALCULATE LAST LOCATION TO BE CLEARED
9438
9439 050034 005020      1$:     CLR      (R0)+      ;CLEAR RETURN PARAMETER
9440 050036 020001      CMP      R0,R1      ;CHECK IF FINISHED
9441 050040 101775      BLOS     1$         ;NO, CLEAR NEXT RETURN PARAMETER
9442 050042 105037 001506      CLR      I.ISRL     ;CLEAR RELEASE OR INTERRUPT ISSUED
9443 050046 010465 000020      MOV      R4,P.CS2(R5) ;STORE DRIVE NUMBER
9444 050052 005062 000026      CLR      RKMRI(R2)  ;CLEAR RK06 MAINTENANCE REGISTER 1
9445 050056 132765 000040 000001      BITB     #BIT5,P.CMND(R5) ;CHECK IF SPECIAL COMMAND
9446 050064 001402      BEQ      3$         ;NO, PROCESS
9447 050066 000137 050446      JMP      C.SPEC     ;JUMP TO SPECIAL COMMAND PROCESSOR
9448
9449 050072 122765 000107 000001 3$:     CMPB     #UNLOAD,P.CMND(R5) ;CHECK IF POSITIONING COMMAND
9450                                     ;START SPINDLE
9451                                     ;RECALIBRATE
9452                                     ;OFFSET
9453                                     ;SEEK
9454                                     ;UNLOAD
9455
9456 050100 101125      BHI      25$        ;NO, DRIVE COMMAND
9457                                     ;SELECT DRIVE
9458                                     ;PACK ACKNOWLEDGE
9459                                     ;CLEAR
9460
9461 050102 122765 000117 000001      CMPB     #SEEK,P.CMND(R5) ;CHECK IF DATA TRANSFER
9462 050110 103471      BLO      20$        ;YES, DATA TRANSFER COMMAND
9463                                     ;READ DATA
9464                                     ;WRITE DATA
9465                                     ;READ HEADER
9466                                     ;WRITE HEADER
9467                                     ;WRITE CHECK
9468 050112 016562 000020 000010      MOV      P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
9469 050120 052765 000002 000014      BIS      #DRVPOS,P.PRST(R5) ;SET DRIVE POSITIONING
9470 050126 005037 001464      CLR      0.WAIT     ;CLEAR WAIT FOR COMMAND
9471 050132 122765 000117 000001      CMPB     #SEEK,P.CMND(R5) ;CHECK IF SEEK
9472 050140 001007      BNE      5$         ;NO, CHECK FOR OFFSET
9473 050142 016562 000002 000020      MOV      P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
9474 050150 016562 000004 000006      MOV      P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
9475 050156 000431      BR       8$         ;GO ISSUE COMMAND
9476
9477 050160 122765 000115 000001 5$:     CMPB     #OFFSET,P.CMND(R5) ;CHECK IF OFFSET
9478 050166 001007      BNE      6$         ;NO, CHECK FOR UNLOAD
9479 050170 116565 000006 000032      MOVB     P.OFST(R5),P.ASOF(R5) ;STORE OFFSET
9480 050176 016562 000032 000016      MOV      P.ASOF(R5),RKASOF(R2) ;LOAD OFFSET REGISTER
9481 050204 000416      BR       8$         ;GO ISSUE COMMAND
9482
9483 050206 122765 000111 000001 6$:     CMPB     #SRTSPL,P.CMND(R5) ;CHECK IF START SPINDLE
9484 050214 001003      BNE      7$         ;NO, CHECK IF RECAL
9485 050216 013763 001476 001544      MOV      W.MIN,W.DRV(R3) ;LOAD WATCH DOG TIME FOR 1 MINUTE
9486 050224 122765 000113 000001 7$:     CMPB     #RECAL,P.CMND(R5) ;CHECK IF RECAL
9487 050232 001003      BNE      8$         ;NO, CONTINUE

```

9488	050234	013763	001474	001544		MOV	W.8SEC,W.DRV(R3) ;LOAD RECAL TIME TO 8 SECONDS
9489	050242	116565	000007	000017	8\$:	MOV	P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
9490	050250	042765	165777	000016		BIC	*↑C<CFMT!COT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT
9491							AND DRIVE TYPE
9492	050256	116565	000001	000016		MOV	P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
9493	050264	042765	000200	000014		BIC	*W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
9494	050272	000460				BR	30\$;GO, ISSUE COMMAND
9495							
9496	050274	016562	000010	000004	20\$:	MOV	P.BALO(R5),RKBA(R2) ;LOAD BUS ADDRESS REGISTER
9497	050302	016562	000012	000002		MOV	P.WC(R5),RKWC(R2) ;LOAD WORD COUNT REGISTER
9498	050310	016562	000002	000020		MOV	P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
9499	050316	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK NUMBER
9500	050324	122765	000131	000001		CMPB	*WRTCHK,P.CMND(R5) ;CHECK IF WRITE CHECK COMMAND
9501	050332	001010				BNE	25\$;NO, GO ISSUE THE COMMAND
9502	050334	032765	000200	000014		BIT	*W.WCK,P.PRST(R5) ;CHECK IF WRITE COMMAND SHOULD BE ISSUED
9503	050342	001404				BEQ	25\$;NO, GO ISSUE THE COMMAND
9504	050344	012765	000123	000016		MOV	*WRDATA,P.CS1(R5) ;ISSUE WRITE COMMAND
9505	050352	000406				BR	26\$;GO ISSUE COMMAND
9506							
9507	050354	116565	000001	000016	25\$:	MOV	P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
9508	050362	042765	000200	000014		BIC	*W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
9509	050370	116565	000007	000017	26\$:	MOV	P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
9510	050376	142765	177750	000017		BICB	*↑C<B.CFMT!B.COT!B.BA16!B.BA17>,P.CS1+1(R5) ;CLEAR ALL BITS EXCEPT
9511							FORMAT, DRIVE TYPE, AND BUS ADDRESS
9512							BITS 16-17
9513	050404	010537	001464			MOV	R5,0.WAIT ;LOAD WAITING FOR COMMAND
9514	050410	032765	100000	000014		BIT	*DIBAI,P.PRST(R5) ;CHECK IF INHIBIT BUS ADDRESS INCREMENT
9515	050416	001403				BEQ	27\$;NO, LOAD CS2
9516	050420	052765	000020	000020		BIS	*BAI,P.CS2(R5) ;SET INHIBIT BUS ADDRESS INCREMENT
9517	050426	016562	000020	000010	27\$:	MOV	P.CS2(R5),RKCS2(R2) ;LOAD CS2
9518	050434	016562	000016	000000	30\$:	MOV	P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
9519	050442	000137	051364			JMP	C.RTRN ;RESTORE REGISTERS
9520							
9521						.SBTTL	*SPECIAL COMMAND PROCESSING
9522							
9523	050446	122765	000141	000001	C.SPEC:	CMPB	*RDSTAT,P.CMND(R5) ;CHECK IF READ DRIVE STATUS
9524	050454	001132				BNE	10\$;NO, PROCESS OTHER COMMANDS
9525	050456	016562	000020	000010		MOV	P.CS2(R5),RKCS2(R2) ;LOAD CS2 FOR COMMAND
9526	050464	116565	000007	000017		MOV	P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
9527	050472	042765	165777	000016		BIC	*↑C<CFMT!COT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT
9528							AND DRIVE TYPE
9529	050500	112765	000001	000016		MOV	*DR.SEL,P.CS1(R5) ;STORE COMMAND
9530	050506	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
9531	050514	004737	044112		2\$:	JSR	PC,W.WTCH ;CALL WATCH-DOG TIMER
9532	050520	016265	000000	000016		MOV	RKCS1(R2),P.CS1(R5) ;STORE COMMAND AND STATUS REG. 1
9533	050526	032765	000200	000016		BIT	*RDY,P.CS1(R5) ;WAIT FOR READY
9534	050534	001767				BEQ	2\$
9535	050536	004737	047406			JSR	PC,I.CST1 ;STORE CONTROLLER REGISTERS
9536	050542	016265	000034	000040		MOV	RKMR2(R2),P.A00(R5) ;STORE STATUS BYTE 00 MESSAGE A
9537	050550	016265	000036	000042		MOV	RKMR3(R2),P.B00(R5) ;STORE STATUS BYTE 00 MESSAGE B
9538	050556	032765	100000	000016		BIT	*CERR,P.CS1(R5) ;CHECK IF CONTROLLER ERROR
9539	050564	001443				BEQ	6\$;NO, GATHER DRIVE STATUS
9540	050566	146437	001514	001512		BICB	INTMSK(R4),W.TIME ;RESET WATCH DOG TIMING ON THIS DRIVE
9541	050574	005063	001544			CLR	W.DRV(R3) ;CLEAR WATCH DOG COUNT
9542	050600	032765	001000	000020		BIT	*MDS,P.CS2(R5) ;CHECK IF MULTIPLE DRIVE SELECT
9543	050606	001046				BNE	9\$;YES, INDICATE CONTROLLER ERROR

9544	050610	004037	046712			JSR	RO, I.CCLR	: CLEAR ERROR
9545	050614	051364				C.RTRN		: ERROR RETURN
9546	050616	032765	010400	000020		BIT	#NED!UFE, P.CS2(R5)	: CHECK IF NON-EXISTENT DRIVE OR UNIT FIELD ERROR
9547	050624	001017				BNE	5\$: REPORT ERROR
9548	050626	032765	000001	000036		BIT	#DRA, P.DS(R5)	: CHECK IF DRIVE AVAILIABLE
9549	050634	001013				BNE	5\$: YES, REPORT ERROR
9550	050636	156437	001514	001512		BISB	INTMSK(R4), W.TIME	: SET TIMING FOR THIS DRIVE
9551	050644	013763	001476	001544		MOV	W.MIN, W.DRV(R3)	: WAIT 1 MINUTE FOR RELEASE
9552	050652	052765	010000	000014		BIS	#DRVSZD, P.PRST(R5)	: SET DRIVE SEIZED
9553	050660	000137	051364			JMP	C.RTRN	: RESTORE REGISTERS
9554								
9555	050664	004777	130566		5\$:	JSR	PC, QA.ABNL	: REPORT ERROR
9556	050670	000137	051364			JMP	C.RTRN	: RESTORE REGISTERS
9557								
9558	050674	004037	047470		6\$:	JSR	RO, I.STAT	: GATHER DRIVE STATUS
9559	050700	051364				C.RTRN		: ERROR RETURN
9560	050702	146437	001514	001512		BICB	INTMSK(R4), W.TIME	: STOP WATCH-DOG TIMING ON DRIVE
9561	050710	005063	001544			CLR	W.DRV(R3)	: RESET WATCH-DOG TIME
9562	050714	004777	130534			JSR	PC, QA.NORM	: REPORT COMMAND COMPLETE
9563	050720	000137	051364			JMP	C.RTRN	: RESTORE REGISTERS
9564								
9565	050724	052737	100000	001462	9\$:	BIS	#E.MDS, E.CONT	: SET MULTIPLE DRIVE SELECT
9566	050732	004777	130522			JSR	PC, QA.CONT	: INDICATE CONTROLLER ERROR
9567	050736	000137	051364			JMP	C.RTRN	
9568								
9569	050742	122765	000140	000001	10\$:	CMPB	#RELEASES, P.CMND(R5)	: CHECK IF RELEASE COMMAND
9570	050750	001031				BNE	13\$: NO, CHECK IF READ ALL HEADERS
9571	050752	010537	001464			MOV	RS, O.WAIT	: STORE PARAMETER BLOCK ADDRESS IN
9572								: WAIT FOR COMMAND
9573	050756	052765	000010	000020		BIS	#RLS, P.CS2(R5)	: SET RELEASE BIT
9574	050764	016562	000020	000010		MOV	P.CS2(R5), RKCS2(R2)	: LOAD CS2 FOR DESELECT
9575	050772	112737	000001	001506		MOV	#1, I.ISRL	: SET FLAG FOR RELEASE COMMAND
9576	051000	116565	000007	000017		MOVB	P.CS1H(R5), P.CS1+1(R5)	: STORE BITS 8-15 OF CS1
9577	051006	042765	165777	000016		BIC	#↑C<CFMT!COT>, P.CS1(R5)	: CLEAR ALL BITS EXCEPT FORMAT
9578								: AND DRIVE TYPE
9579	051014	112765	000101	000016		MOVB	#SELDRV, P.CS1(R5)	: STORE COMMAND
9580	051022	016562	000016	000000	11\$:	MOV	P.CS1(R5), RKCS1(R2)	: ISSUE COMMAND
9581	051030	000137	051364			JMP	C.RTRN	: RESTORE REGISTERS
9582								
9583	051034	122765	000164	000001	13\$:	CMPB	#RDALHD, P.CMND(R5)	: CHECK IF READ ALL HEADERS
9584	051042	001047				BNE	30\$: NO, CHECK IF CONTROLLER CLEAR
9585	051044	010537	001464			MOV	RS, O.WAIT	: SET WAITING FOR COMMAND COMPLETION
9586	051050	016537	000010	001502		MOV	P.BALO(R5), HDR.AD	: LOAD HEADER ADDRESS
9587	051056	132765	000020	000007		BITB	#B.CFMT, P.CS1H(R5)	: CHECK IF 22 SECTOR FORMANT
9588	051064	001404				BEQ	14\$: YES, LOAD 22 IN HEADER COUNT
9589	051066	012737	000024	001504		MOV	#20., HDR.CT	: LOAD 20 IN SECTOR COUNT
9590	051074	000403				BR	22\$: GO ISSUE READ HEADER COMMAND
9591								
9592	051076	012737	000026	001504	14\$:	MOV	#22., HDR.CT	: LOAD 22 IN SECTOR COUNT
9593	051104	016562	000002	000020	22\$:	MOV	P.CYLN(R5), RKOCYL(R2)	: LOAD CYLINDER ADDRESS
9594	051112	016562	000004	000006		MOV	P.SECT(R5), RKDA(R2)	: LOAD TRACK NUMBER
9595	051120	016562	000020	000010		MOV	P.CS2(R5), RKCS2(R2)	: LOAD DRIVE NUMBER
9596	051126	116565	000007	000017		MOVB	P.CS1H(R5), P.CS1+1(R5)	: STORE BITS 8-15 OF CS1
9597	051134	042765	165777	000016		BIC	#↑C<CFMT!COT>, P.CS1(R5)	: CLEAR ALL BITS EXCEPT DRIVE TYPE
9598								: AND FORMAT
9599	051142	112765	000125	000016		MOVB	#RDHEAD, P.CS1(R5)	: STORE READ HEADER COMMAND

I01

CZR6PCO RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 217
*SPECIAL COMMAND PROCESSING

SEQ 0215

9600	051150	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2)	:ISSUE READ HEADER
9601	051156	000137	051364			JMP	C.RTRN	:RESTORE REGISTERS
9602								
9603	051162	122765	00C176	000001	30\$:	CMPB	#CONCLR,P.CMND(R5)	:CHECK IF CONTROLLER CLEAR
9604	051170	001004				BNE	32\$:NO, CHECK IF SUBSYSTEM CLEAR
9605	051172	004037	046712			JSR	RD,I.CCLR	:CLEAR CONTROLLER
9606	051176	051364				C.RTRN		:ERROR RETURN
9607	051200	000467				BR	40\$:INDICATE NORMAL RETURN
9608								
9609	051202	122765	000177	000001	32\$:	CMPB	#SUBCLR,P.CMND(R5)	:CHECK IF SUBSYSTEM CLEAR
9610	051210	001406				REQ	36\$:YES, CLEAR SUBSYSTEM
9611	051212	052737	000100	001462	34\$:	BIS	#E.ILLD,E.CONT	:SET ILLEGAL DRIVER COMMAND
9612	051220	004777	130234			JSR	PC,QA.CONT	:REPORT ERROR
9613	051224	000457				BR	C.RTRN	:RESTORE REGISTERS
9614								
9615	051226	012762	000040	000010	36\$:	MOV	#SCLR,RKCS2(R2)	:ISSUE SUBSYSTEM CLEAR
9616	051234	016265	000000	000016		MOV	RKCS1(R2),P.CS1(R5)	:STORE COMMAND AND STATUS REGISTER 1
9617	051242	032765	100000	000016		BIT	#CERR,P.CS1(R5)	:CLEAR IF CONTROLLER ERROR RESET
9618	051250	001406				BEQ	37\$:NO, FINISH COMMAND
9619	051252	052737	000001	001462		BIS	#BITO,E.CONT	:SET CLEAR SUBSYSTEM DID NOT CLEAR
9620								:CONTROLLER ERROR
9621	051260	004777	130174			JSR	PC,QA.CONT	:REPORT ERROR
9622	051264	000437				BR	C.RTRN	:RESTORE REGISTERS
9623								
9624	051266	013746	001470		37\$:	MOV	W.MILI,-(SP)	:LOAD 16 MILI-SECOND COUNT FOR ATTENTION
9625								:TO DISAPPEAR
9626	051272	016265	000000	000016	38\$:	MOV	RKCS1(R2),P.CS1(R5)	:STORE CS1
9627	051300	032765	040000	000016		BIT	#DI,P.CS1(R5)	:CHECK IF ATTENTIONS CLEARED
9628	051306	001411				BEQ	39\$:YES, FINISH COMMAND
9629	051310	005316				DEC	(SP)	:DECREMENT 16 MILLISECOND COUNT
9630	051312	001367				BNE	38\$:CHECK DRIVE INTERRUPT AGAIN
9631	051314	005726				TST	(SP)+	:ADJUST STACK
9632	051316	052737	000040	001462		BIS	#E.SCLR,E.CONT	:SET SUBSYSTEM CLEAR DID NOT CLEAR
9633								:DRIVE ATTENTIONS
9634	051324	004777	130130			JSR	PC,QA.CONT	:REPORT ERROR
9635	051330	000415				BR	C.RTRN	:RESTORE REGISTER
9636								
9637	051332	005726			39\$:	TST	(SP)+	:ADJUST STACK
9638	051334	032765	000400	000014		BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE
9639	051342	001010				BNE	C.RTRN	:YES, RESTORE REGISTERS
9640	051344	112737	177777	001506		MOVB	#-1,I.ISRL	:SET INTERRUPT ENABLE SET
9641	051352	012762	000100	000000		MOV	#IE,RKCS1(R2)	:SET INTERRUPT ENABLE
9642	051360	004777	130070		40\$:	JSR	PC,QA.NORM	:INDICATE NORMAL TERMINATION
9643								
9644	051364	012637	177776			C.RTRN: MOV	(SP)+,PS	:RESTORE PSW
9645	051370	012600				MOV	(SP)+,R0	:RESTORE R0
9646	051372	012601				MOV	(SP)+,R1	:RESTORE R1
9647	051374	012602				MOV	(SP)+,R2	:RESTORE R2
9648	051376	012603				MOV	(SP)+,R3	:RESTORE R3
9649	051400	012604				MOV	(SP)+,R4	:RESTORE R4
9650	051402	012605				MOV	(SP)+,R5	:RESTORE R5
9651	051404	000207				RTS	PC	:RETURN

.SBTTL *ENQUEUE ROUTINE

```

*****
*
* THIS ROUTINE WILL BUT THE ADDRESS OF THE PARAMETER IN THE
* QUEUE SPECIFIED BY THE LOCATION FOLLOWING THE CALL.
*
*CALL JSR RD,Q.PUSH
* <ADDRESS OF QUEUE>
* RETURN
*
*****

```

```

9652
9653
9654
9655
9656
9657
9658
9659
9660
9661
9662
9663
9664
9665 051406 010446
9666 051410 010346
9667 051412 013746 177776
9668 051416 013737 001452 177776
9669 051424 012004
9670 051426 005065 000064
9671 051432 005724
9672 051434 001405
9673
9674 051436 011403
9675 051440 010563 000064
9676 051444 010514
9677 051446 000402
9678
9679 051450 010514
9680 051452 010544
9681
9682 051454 012637 177776
9683 051460 012603
9684 051462 012604
9685 051464 000200

```

```

Q.PUSH: MOV R4,-(SP) ;STORE R4 ON STACK
MOV R3,-(SP) ;STORE R3 ON STACK
MOV PS,-(SP) ;STORE PSW ON STACK
MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
MOV (R0)+,R4 ;STORE QUEUE ADDRESS
CLR P.QLNK(R5) ;CLEAR QUEUE LINK
TST (R4)+ ;CHECK IF QUEUE IS EMPTY
BEQ 1$ ;YES, PUT PARAMETER BLOCK ADDRESS
; AT HEAD AND TAIL OF THE QUEUE
MOV (R4),R3 ;STORE TAIL OF QUEUE
MOV R5,P.QLNK(R3) ;UPDATE QUEUE LINK
MOV R5,(R4) ;UPDATE QUEUE TAIL
BR 2$

1$: MOV R5,(R4) ;LOAD TAIL OF QUEUE
MOV R5,-(R4) ;LOAD HEAD OF QUEUE

2$: MOV (SP)+,PS ;RESTORE PSW
MOV (SP)+,R3 ;RESTORE R3
MOV (SP)+,R4 ;RESTORE R4
RTS R0 ;RETURN

```

.SBTTL *POP QUEUE ROUTINE

9686
9687
9688
9689
9690
9691
9692
9693
9694
9695
9696
9697
9698
9699
9700
9701
9702
9703
9704
9705
9706
9707
9708
9709
9710
9711
9712
9713
9714
9715
9716
9717
9718
9719

051466 011546
051470 010546
051472 010446
051474 013746 177776
051500 013737 001452 177776
051506 012004
051510 011405
051512 010566 000010
051516 001404
051520 016524 000064
051524 001001
051526 005014
051530 012637 177776
051534 012604
051536 012605
051540 000200

```

*****
*
* THIS ROUTINE WILL PUT THE ADDRESS OF THE FIRST PARAMTER
* BLOCK ADDRESS OF THIS QUEUE ON THE STACK AND REMOVE IT
* FROM THE QUEUE DESIGNATED BY THE CALL. A RETURN OF ZERO
* ON THE STACK INDICATES THAT THE QUEUE WAS EMPTY. OTHERWISE,
* THE STACK WILL CONTAIN THE ADDRESS OF THE FIRST
* PARAMETER BLOCK ON THE QUEUE.
*
*CALL JSR RO,Q.POP
* <ADDRESS OF QUEUE>
* RETURN
*
*****
Q.POP: MOV (SP),-(SP) ;CREATE SPACE ON THE STACK
MOV R5,-(SP) ;STORE R5 ON THE STACK
MOV R4,-(SP) ;STORE R4 ON THE STACK
MOV PS,-(SP) ;STORE PSW ON THE STACK
MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
MOV (R0)+,R4 ;STORE QUEUE ADDRESS
MOV (R4),R5 ;STORE PARAMETER BLOCK ADDRESS
MOV R5,10(SP) ;LOAD ADDRESS ON STACK
BEQ 1$ ;IF QUEUE IS EMPTY, RETURN
MOV P.QLNK(R5),(R4)+ ;READJUST HEAD OF THE QUEUE
BNE 1$ ;IF NOT LAST ELEMENT OF THE QUEUE, RETURN
CLR (R4) ;CLEAR TAIL POINTER IF LAST ELEMENT OF QUEUE

1$: MOV (SP)+,PS ;RESTORE PSW
MOV (SP)+,R4 ;RESTORE R4
MOV (SP)+,R5 ;RESTORE R5
RTS RO ;RETURN

```

SBTTL *SEARCH QUEUE FOR DRIVE NOT POSITIONING

```

*****
*
* THIS ROUTINE WILL SEARCH THROUGH THE COMMAND INITIATION
* QUEUE FOR THE FIRST PARAMETER BLOCK WHOSE DRIVE IS
* NOT POSITIONING. IF THE QUEUE IS EMPTY OF ALL DRIVES
* ARE POSITIONING ZERO IS PLACED IN R5. OTHERWISE
* THE ADDRESS OF THE PARAMETER BLOCK IS PLACED IN R5.
*
*CALL JSR PC,Q.SRCH
* RETURN
*
*NOTE: THIS ROUTINE DESTROYS R4.
*
*****

```

```

9720
9721
9722
9723
9724
9725
9726
9727
9728
9729
9730
9731
9732
9733
9734
9735
9736
9737 051542 012704 001324 Q.SRCH: MOV #CINITQ,R4 ;LOAD ADDRESS OF COMMAND INITIATION
9738 ; QUEUE IN R4
9739 051546 011405 1$: MOV (R4),R5 ;GET NEXT QUEUE ELEMENT
9740 051550 001425 BEQ 5$ ;IF END OF QUEUE, RETURN
9741 051552 032765 0000C2 000014 BIT #DRVPOS,P.PRST(R5) ;CHECK IF DRIVE IS POSITIONING
9742 051560 001404 BEQ 2$ ;NO, RETURN PARAMETER BLOCK ADDRESS
9743 051562 010504 MOV R5,R4 ;UPDATE LINK ADDRESS POINTER
9744 051564 062704 000064 ADD #P.QLNK,R4 ;CALCULATE LINK ADDRESS
9745 051570 000766 BR 1$ ;GET NEXT ELEMENT OF QUEUE
9746
9747 051572 016514 000064 2$: MOV P.QLNK(R5),(R4) ;UPDATE QUEUE LINK
9748 051576 001012 BNE 5$ ;CHECK IF LAST ELEMENT OF QUEUE
9749 ; IF NOT, DO NOT UPDATE QUEUE TAIL
9750 051600 022704 001324 CMP #CINITQ,R4 ;CHECK IF ONLY ELEMENT IN QUEUE
9751 051604 001405 BEQ 4$ ;YES, CLEAR HEAD AND TAIL
9752 051606 162704 000064 SUB #P.QLNK,R4 ;ADJUST R4 FOR TAIL POINTER
9753 051612 010437 001326 MOV R4,CINITQ+2 ;LOAD TAIL
9754 051616 000207 RTS PC ;RETURN
9755
9756 051620 005024 4$: CLR (R4)+ ;CLEAR HEAD OF QUEUE
9757 051622 005014 CLR (R4) ;CLEAR TAIL OF QUEUE
9758 051624 000207 5$: RTS PC ;RETURN

```

MO1

CZR6PC0 RK611-06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 221

SEQ 0219

*REMOVE PARAMETER BLOCK FROM COMMAND INITIATION QUEUE

.SBTTL *REMOVE PARAMETER BLOCK FROM COMMAND INITIATION QUEUE

```

*****
*
* THIS ROUTINE WILL CHECK IF THE CURRENT PARAMETER BLOCK
* IS IN THE COMMAND INITIATION QUEUE. IF IT IS, THIS
* ROUTINE WILL REMOVE IT FROM THE COMMAND INITIATION QUEUE.
*
*CALL JSR PC,Q.RMOV
* RETURN
*
*NOTE: R5 CONTAINS ADDRESS OF CURRENT PARAMETER BLOCK.
*
*****

```

```

9759
9760
9761
9762
9763
9764
9765
9766
9767
9768
9769
9770
9771
9772
9773
9774 051626 032765 040000 000014 Q.RMOV: BIT #Q.INIT,P.PRST(R5) ;CHECK IF PARAMETER BLICK ENQUEUED IN
9775 ; INITIATION QUEUE
9776 051634 001001 BNE 5$ ; YES, REMOVE PARAMETER BLOCK
9777 ; FROM COMMAND INITIATION QUEUE
9778 051636 J0207 RTS PC ;RETURN
9779
9780 051640 J10446 5$: MOV R4,-(SP) ;STORE R4 ON STACK
9781 051642 010346 MOV R3,-(SP) ;STORE R3 ON THE STACK
9782 051644 012704 001324 MOV #CINITQ,R4 ;LOAD ADDRESS OF COMMAND INITIATION
9783 ; QUEUE IN R4
9784 051650 011403 10$: MOV (R4),R3 ;GET NEXT QUEUE ELEMENT
9785 051652 001001 BNE 11$ ;CHECK IF NO MORE ELEMENTS
9786 051654 000000 HALT ; *** FLAGS DO NOT AGREE WITH
9787 ; COMMAND INITIATION QUEUE
9788 051656 020305 11$: CMP R3,R5 ;CHECK IF CORRECT PARAMETER BLOCK
9789 051660 001404 BEQ 12$ ;YES, ADJUST QUEUE
9790 051662 010304 MOV R3,R4 ;UPDATE LINK ADDRESS POINTER
9791 051664 062704 000064 ADD #P.QLNK,R4 ;CALCULATE LINK ADDRESS
9792 051670 000767 BR 10$ ;GET NEXT ELEMENT OF QUEUE
9793
9794 051672 016314 000064 12$: MOV P.QLNK(R3),(R4) ;UPDATE QUEUE LINK
9795 051676 001012 BNE 15$ ;CHECK IF LAST ELEMENT OF QUEUE
9796 ; IF NOT, DO NOT UPDATE QUEUE TAIL
9797 051700 022704 001324 CMP #CINITQ,R4 ;CHECK IF ONLY ELEMENT IN QUEUE
9798 051704 001405 BEQ 14$ ;YES, CLEAR HEAD AND TAIL
9799 051706 162704 000064 SUB #P.QLNK,R4 ;ADJUST R4 FOR TAIL POINTER
9800 051712 010437 001326 MOV R4,CINITQ+2 ;LOAD TAIL
9801 051716 000402 BR 15$ ;RETURN
9802
9803 051720 005024 14$: CLR (R4)+ ;CLEAR HEAD OF QUEUE
9804 051722 005014 CLR (R4) ;CLEAR TAIL OF QUEUE
9805 051724 042765 040006 000014 15$: BIC #DRVPOD!DRVPOS!Q ;INIT P.PRST(R5) ;CLEAR DRIVE POSITIONING,
9806 ; DRIVE POSITIONING FOR DATA TRANSFER, AND
9807 ; PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
9808
9809 051732 012603 MOV (SP)+,R3 ;RESTORE R3
9810 051734 012604 MOV (SP)+,R4 ;RESTORE R4
9811 051736 000207 RTS PC ;RETURN

```

.SBTTL *COMMAND OPTIMIZER

```

*****
*
* THIS ROUTINE WILL INITIATE THE COMMAND AS SPECIFIED IN THE
* PARAMETER BLOCK OF THE FIRST DRIVE WHICH IS NOT POSITIONING
* IN THE COMMAND INITIATION QUEUE.
*
*CALL JSR PC,C.OPT
* RETURN
*
* ROUTINES USED:
* C.INIT
* Q.PUSH
* Q.SRCH
*
*****

```

```

9812
3613
9814
9815
9816
9817
9818
9819
9820
9821
9822
9823
9824
9825
9826
9827
9828
9829
9830 051740 010546
9831 051742 010446
9832 051744 010346
9833 051746 010246
9834 051750 010146
9835 051752 010046
9836 051754 013746 177776
9837 051760 012737 000001 001500
9838 051766 013737 001452 177776
9839 051774 005737 001464
9840 052000 001027
9841 052002 013702 001446
9842
9843 052006 016237 000000 001410
9844 052014 032737 000200 001410
9845 052022 001416
9846 052024 004737 051542
9847 052030 005705
9848 052032 001014
9849 052034 032737 040000 001410
9850 052042 001406
9851 052044 112737 177777 001506
9852 052052 012762 000300 000000
9853 052060 000137 052444
9854
9855 052064 112737 000001 001410
9856 052072 004037 046774
9857 052076 052444
9858 052100 032737 010000 001412
9859 052106 001034
9860 052110 032737 000001 001430
9861 052116 001030
9862 052120 013703 001412
9863 052124 042703 177770
9864 052130 156337 001514 001512
9865 052136 006303
9866 052140 013763 001476 001544
9867 052146 005037 001464

```

```

C.OPT: 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 #1,OPTFLG ;SET FLAG FOR CMND OPTIMIZER
MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
TST O.WAIT ;CHECK IF WAITING FOR COMMAND COMPLETION
BNE 12$ ;YES, RESTORE REGISTERS
MOV RKBAS,R2 ;LOAD R2 WITH BASE ADDRESS OF THE RK06
;REGISTERS
MOV RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS REGISTER 1
BIT #RDY,T.CS1 ;CHECK IF READY SET
BEQ 12$ ;NO, WAIT FOR COMMAND COMPLETION
JSR PC,Q.SRCH ;SEARCH FOR PARAMETER BLOCK ADDRESS
TST R5 ;CHECK IF NO COMMAND CAN BE ISSUED
BNE 1$ ;YES, ISSUE COMMAND
BIT #DI,T.CS1 ;CHECK IF DRIVE INTERRUPT WAITING
BEQ 12$ ;NO RETURN
MOVB #-1,I.ISRL ;SET FLAG THAT INTERRUPT WAS FORCED
MOV #INTR,RKCS1(R2) ;FORCE INTERRUPT TO SCAN DRIVE ATTENTIONS
JMP O.RTRN ;RESTORE REGISTERS

12$:
1$: MOVB #DR.SEL,T.CS1 ;SET DRIVE SELECT BIT
JSR RO,I.ISSU ;ISSUE DRIVE SELECT COMMAND
O.RTRN ;ERROR RETURN ADDRESS
BIT #NED,T.CS2 ;SEE IF NED ERROR SET
BNE 24$ ;BR IF YES
BIT #DRA,T.DS ;SEE IF DRIVE IS AVAILABLE
BNE 24$ ;BR IF AVAILABLE
MOV T.CS2,R3 ;GET RKCS2
BIC #177770,R3 ;CLEAR ALL BUT DRIVE NO.
BISB INTMSK(R3),W.TIME ;SET BIT FOR THIS DRIVE
ASL R3
MOV W.MIN,W.DRV(R3) ;SET TIMER FOR 1 MINUTE FOR RELEASE
CLR O.WAIT ;CLEAR WAITING FOR CMND COMPLETION

```

```

9868 052152 004037 046712 JSR RO,I.CCLR ;ISSUE CONTROLLER CLEAR
9869 052156 052444 J.RTRN ;ERROR RETURN ADDRESS
9870 052160 042765 C40006 000014 BIC #DRVPOS!DRVPDT!Q.INIT,P.PRST(R5)
9871 052166 052765 010000 000014 10$: BIS #DRVSZD,P.PRST(R5) ;SET DRIVE SIEZED FLAG
9872 052174 000137 052444 JMP O.RTRN ;EXIT
9873 052200 24$: CLR P.ERR(R5) ;CLEAR ERROR STATUS INFORMATION
9874 052200 005065 000212 CMPB #RDALHD,P.CMND(R5) ;CHECK IF READ ALL HEADERS
9875 052204 122765 000164 000001 BEQ 2$ ;YES, CHECK IF IMPLIED SEEK
9876 052212 001410 CMPB #RDDATA,P.CMND(R5) ;CHECK IF DATA TRANSFER
9877 052214 122765 000121 000001 BHI 3$ ;NO, GO TO THE COMMAND INITIATOR
9878 052222 101021 CMPB #!RTCHK,P.CMND(R5) ;CHECK IF SPECIAL COMMAND
9879 052224 122765 000131 000001 BLO 3$ ;YES, GO TO THE COMMAND INITIATOR
9880 052232 103415 MOVB P.DRVN(R5),R4 ;STORE DRIVE NUMBER
9881 052234 116504 000000 2$: BITB INTMSK(R4),O.OVER ;CHECK IF OVERLAPPED OPERATIONS FOR THIS DRIVE
9882 052240 136437 001514 001513 BEQ 3$ ;NO, GO TO THE COMMAND INITIATOR
9883 052246 001407 BIT #DRVPDT,P.PRST(R5) ;CHECK IF POSITIONING FOR
9884 052250 032765 000004 000014 ; DATA TRANSFER OCCURED
9885 BEQ 5$ ;NO, ISSUE SEEK
9886 052256 001411 BIC #DRVPDT!Q.INIT,P.PRST(R5) ;CLEAR POSITIONING IN PROGRESS FOR
9887 052260 042765 040004 000014 ; DATA TRANSFER AND PARAMETER BLOCK ENQUEUED
9888 ; IN INITIATION QUEUE
9889
9890
9891 052266 010537 052276 3$: MOV R5,4$ ;LOAD PARAMETER BLOCK ADDRESS
9892 052272 004737 047716 JSR PC,C.INIT ;ISSUE COMMAND
9893 052276 000000 4$: .WORD 0 ;ADDRESS OF PARAMETER BLOCK
9894 ; PLACED HERE
9895 052300 000461 BR O.RTRN ;RESTORE REGISTERS
9896
9897 052302 156437 001514 001512 5$: BISB INTMSK(R4),W.TIME ;SET WATCH DOG TIMING
9898 052310 010403 MOV R4,R3 ;STORE DRIVE NUMBER
9899 052312 006303 ASL R3 ;MULTIPLY DRIVE NUMBER BY 2
9900 052314 013763 001472 001544 MOV W.SEC,W.DRV(R3) ;LOAD WATCH-DOG TIME
9901 052322 005037 001464 CLR O.WAIT ;CLEAR WAIT FOR COMMAND
9902 052326 105037 001506 CLRB I.ISRL ;CLEAR RELEASE OF INTERRUPT ISSUED
9903
9904 ; RESET ALL THE BITS IN THE PROGRAM STATUS REGISTER EXCEPT
9905 ; DRIVE IN USE
9906 ; WRITE FOR WRITE CHECK
9907 ; NO CHECK
9908 ; DROP DRIVE FROM TEST SEQUENCE
9909 ; INHIBIT BUS ADDRESS INCREMENT
9910 052332 042765 075176 000014 BIC #!C<DRVUSE!W.WCK!NOCHK!DRPDRV!DTBAII>,P.PRST(R5)
9911 ;
9912 ; SET DRIVE POSITIONING, DRIVE POSITIONING BECAUSE OF DATA TRANSFER,
9913 ; AND PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
9914 052340 052765 040006 000014 BIS #DRVPOS!DRVPDT!Q.INIT,P.PRST(R5)
9915
9916 052346 010500 MOV R5,RO ;STORE PARAMTER BLOCK ADDRESS
9917 052350 062700 000016 ADD #P.CS1,RO ;CALCULATE FIRST LOCATION TO BE CLEARED
9918 052354 010501 MOV R5,R1 ;STORE PARAMETER BLOCK ADDRESS
9919 052356 062701 000062 ADD #P.EPAT,R1 ;CALCULATE LAST LOCATION TO BE CLEARED
9920
9921 052362 005020 6$: CLR (RO)+ ;CLEAR RETURN PARAMETERS
9922 052364 020001 CMP RO,R1 ;CHECK IF FINISHED
9923 052366 101775 BLOS 6$ ;NO, CLEAR NEXT RETURN PARAMETER

```

9924	052370	005062	000026		CLR	RKMR1(R2)	;CLEAR MAINTENANCE REGISTER 1
9925	052374	016562	000002	000020	MOV	P.CYLN(R5),RKDCYL(R2)	;LOAD CYLINDER ADDRESS
9926	052402	010462	000010		MOV	R4,RKCS2(R2)	;LOAD DEVICE NUMBER
9927	052404	116565	000007	000017	MOVB	P.CS1H(R5),P.CS1+1(R5)	;STORE BITS 8-15 OF CS1
9928	052414	042765	165777	000016	BIC	#1C<CFMT!COT>,P.CS1(R5)	;CLEAR BITS EXCEPT FORMAT AND DRIVE TYPE
9929							
9930	052422	112765	000117	000016	MOVB	#SEEK,P.CS1(R5)	;STORE COMMAND
9931	052430	016562	000016	000000	MOV	P.CS1(R5),RKCS1(R2)	;ISSUE SEEK
9932	052436	004037	051406		JSR	RD,Q.PUSH	;REQUEST COMMAND IN COMMAND
9933	052442	001324			CINITQ		;INITIATION QUEUE
9934							
9935	052444	005037	001500		CLR	OPTFLG	;CLEAR OPTIMIZER FLAG
9936	052450	012637	177776		MOV	(SP)+,PS	;RESTORE PSW
9937	052454	012600			MOV	(SP)+,R0	;RESTORE R0
9938	052456	012601			MOV	(SP)+,R1	;RESTORE R1
9939	052460	012602			MOV	(SP)+,R2	;RESTORE R2
9940	052462	012603			MOV	(SP)+,R3	;RESTORE R3
9941	052464	012604			MOV	(SP)+,R4	;RESTORE R4
9942	052466	012605			MOV	(SP)+,R5	;RESTORE R5
9943	052470	000207			RTS	PC	;RETURN

O.RTRN:

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 225
MEMORY CHECK ENABLE TEST

SEQ 0223

9944			
9945			
9946	052472	104401	064700
9947	052476	011646	
9948	052500	004737	052774
9949	052504	104401	001165
9950	052510	000002	

.SBTTL MEMORY CHECK ENABLE TEST

MEMERR:	TYPE	ERR400		;TYPE UNEXPECTED MEMORY PARITY TRAP
	MOV	{SP}-(SP)		;TYPE PC VALUE
	JSR	PC,BINOC1		
	TYPE	,%CRLF		
	RTI			;RETURN

.SBTTL TYPE ROUTINE

9951
9952
9953
9954
9955
9956
9957
9958
9959
9960
9961
9962
9963
9964
9965
9966
9967
9968
9969
9970
9971
9972
9973
9974
9975
9976
9977
9978
9979
9980
9981
9982
9983
9984
9985
9986
9987
9988
9989
9990
9991
9992
9993
9994
9995
9996
9997
9998
9999
10000
10001
10002
10003
10004
10005
10006

052512 105737 001157
052516 100002
052520 000000
052522 000430
052524 010046
052526 017600 000002
052532 122737 000001 001210
052540 001011
052542 132737 000100 001211
052550 001405
052552 010037 052562
052556 004737 055424
052562 000000
052564 132737 000040 001211
052572 001003
052574 112046
052576 001005
052600 005726
052602 012600
052604 062716 000002
052610 000002
052612 122716 000011
052616 001430
052620 122716 000200
052624 001006
052626 005726
052630 104401
052632 001165
052634 105037 052770
052640 000755
052642 004737 052724
052646 123726 001156
052652 001350
052654 013746 001154
052660 105366 000001
052664 002770
052666 004737 052724
052672 105337 052770

```
*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*
$TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
BPL 1$ ;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3$ ;; LEAVE
1$: MOV RO, -(SP) ;; SAVE RO
MOV 22(SP), RO ;; GET ADDRESS OF ASCIZ STRING
CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
BNE 62$ ;; NO GO CHECK FOR APT CONSOLE
BITB #APTSPOOL, $ENVM ;; SPOOL MESSAGE TO APT
BEQ 62$ ;; NO GO CHECK FOR CONSOLE
MOV RO, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
0 ;; MESSAGE ADDRESS
61$: .WORD 0 ;; APT CONSOLE SUPPRESSED
62$: BITB #APTCSUP, $ENVM ;; YES, SKIP TYPE OUT
BNE 60$ ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2$: MOVB (RO)+, -(SP) ;; BR IF IT ISN'T THE TERMINATOR
BNE 4$ ;; IF TERMINATOR POP IT OFF THE STACK
TST (SP)+ ;; RESTORE RO
60$: MOV (SP)+, RO ;; ADJUST RETURN PC
3$: ADD #2, (SP) ;; RETURN
4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
BEQ 8$ ;; BRANCH IF NOT <CRLF>
CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
BNE 5$ ;; POP <CR><LF> EQUIV
TST (SP)+ ;; TYPE A CR AND LF
5$: JSR PC, $TYPEC ;; CLEAR CHARACTER COUNT
6$: CMPB $FILLC, (SP)+ ;; GET NEXT CHARACTER
BNE 2$ ;; GO TYPE THIS CHARACTER
MOV $NULL, -(SP) ;; IS IT TIME FOR FILLER CHARS.?
7$: DECB 1(SP) ;; IF NO GO GET NEXT CHAR.
BLT 6$ ;; GET # OF FILLER CHARS. NEEDED
JSR PC, $TYPEC ;; AND THE NULL CHAR.
DECB $CHARCNT ;; DOES A NULL NEED TO BE TYPED?
; ; BR IF NO--GO POP THE NULL OFF OF STACK
; ; GO TYPE A NULL
; ; DO NOT COUNT AS A COUNT
```

```

10007 052676 000770          BR      7$          ;;LOOP
10008
10009          ;HORIZONTAL TAB PROCESSOR
10010
10011 052700 112716 000040      8$:   MOVB   #' (SP)          ;; REPLACE TAB WITH SPACE
10012 052704 004737 052724      9$:   JSR    PC,$TYPEC          ;; TYPE A SPACE
10013 052710 132737 000007 052770  BITB   #7,$SCHARCNT          ;; BRANCH IF NOT AT
10014 052716 001372          BNE    9$          ;; TAB STOP
10015 052720 005726          TST    (SP)+          ;; POP SPACE OFF STACK
10016 052722 000724          BR      2$          ;; GET NEXT CHARACTER
10017 052724 105777 126220      $TYPEC: TSTB  2$TPS          ;; WAIT UNTIL PRINTER IS READY
10018 052730 100375          BPL    $TYPEC
10019 052732 116677 000002 126212  MOVB   2(SP),2$TPB          ;; LOAD CHAR TO BE TYPED INTO DATA REG.
10020 052740 122766 000015 000002  CMPB   #CR,2(SP)          ;; IS CHARACTER A CARRIAGE RETURN?
10021 052746 001003          BNE    1$          ;; BRANCH IF NO
10022 052750 105037 052770          CLRB   $SCHARCNT          ;; YES--CLEAR CHARACTER COUNT
10023 052754 000406          BR      $TYPEX          ;; EXIT
10024 052756 122766 000012 000002 1$:   CMPB   #LF,2(SP)          ;; IS CHARACTER A LINE FEED?
10025 052764 001402          BEQ    $TYPEX          ;; BRANCH IF YES
10026 052766 105227          INCB   (PC)+          ;; COUNT THE CHARACTER
10027 052770 000000          $SCHARCNT: .WORD 0          ;; CHARACTER COUNT STORAGE
10028 052772 000207      $TYPEX: RTS      PC
10029
10030
10031          .SBTTL  BINARY TO OCTAL (ASCII) CONVERSION
10032
10033          ;*****
10034          ;
10035          ; THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A
10036          ; 6-DIGIT OCTAL (ASCII) NUMBER.
10037          ;
10038          ;CALL  MOV     NUM, -(SP)
10039          ;      JSR    PC,BINOCT
10040          ;      RETURN          ;NUMBER IN OCTSTG
10041          ;
10042          ;*****
10043
10044          BINOCT:
10045          MOV     R3, -(SP)          ;; PUSH R3 ON STACK
10046          MOV     R4, -(SP)          ;; PUSH R4 ON STACK
10047          MOV     R5, -(SP)          ;; PUSH R5 ON STACK
10048          MOV     #6,R4          ;; STORE COUNT
10049          MOV     10(SP),R5          ;; PICKUP INPUT NUMBER
10050          MOV     #OCTSTG,R3          ;; LOAD ADDRESS OF CONVERTED STRING
10051          CLR     (R3)          ;; CLEAR OUTPUT BYTE
10052          ROL    R5          ;; ROTATE MSB INTO "C"
10053          BR     3$          ;; GO DO MSB
10054
10055          2$:   ROL    R5          ;; FORM THIS DIGIT
10056          ROL    R5
10057          ROL    R5
10058          MOVB   R5,(R3)
10059          3$:   ROLB   (R3)          ;; GET LSB OF THIS DIGIT
10060          BICB   #177770,(R3)          ;; GET RID OF JUNK
10061          BISB   #'0,(R3)+          ;; MAKE IT ASCII
10062          DEC    R4          ;; CHECK IF DONE

```

```

10063 053050 001365          BNE      2$          ;NO CONVERT NEXT DIGIT
10064 053052 012605          MOV      (SP)+,R5    ;POP STACK INTO R5
10065 053054 012604          MOV      (SP)+,R4    ;POP STACK INTO R4
10066 053056 012603          MOV      (SP)+,R3    ;POP STACK INTO R3
10067 053060 012616          MOV      (SP)+,(SP)  ;SETUP STACK FOR RETURN
10068 053062 104401 053070    TYPE    OCTSTG      ;TYPE NUMBER
10069 053066 000207          RTS      PC          ;RETURN
10070
10071 053070 000006          OCTSTG: .BLKB      6
10072 053076 000          .BYTE    0,0
10073
10074          .SBTTL  TTY INPUT ROUTINE
10075
10076          ;*****
10077          .ENABL  LSB
10078
10079          .DSABL  LSB
10080
10081          ;*****
10082          ;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
10083          ;CALL:
10084          ;*
10085          ;*      RDCHR          ;INPUT A SINGLE CHARACTER FROM THE TTY
10086          ;*      RETURN HERE    ;CHARACTER IS ON THE STACK
10087          ;*                    ;WITH PARITY BIT STRIPPED OFF
10088          ;*
10089          ;
10090 053100 011646          $RDCHR: MOV      (SP),-(SP) ;PUSH DOWN THE PC
10091 053102 016666 000004 000002    MOV      4(SP),2(SP) ;SAVE THE PS
10092 053110 105777 126030    1$:      TSTB    2$TKS    ;WAIT FOR
10093 053114 100375          BPL      1$          ;A CHARACTER
10094 053116 117766 126024 000004    MOVB    2$TKB,4(SP) ;READ THE TTY
10095 053124 042766 177600 000004    BIC     #1C<177>,4(SP) ;GET RID OF JUNK IF ANY
10096 053132 026627 000004 000023    CMP     4(SP),#23    ;IS IT A CONTROL-S?
10097 053140 001013          BNE     3$          ;BRANCH IF NO
10098 053142 105777 125776    2$:      TSTB    2$TKS    ;WAIT FOR A CHARACTER
10099 053146 100375          BPL     2$          ;LOOP UNTIL ITS THERE
10100 053150 117746 125772    MOVB    2$TKB,-(SP) ;GET CHARACTER
10101 053154 042716 177600    BIC     #1C177,(SP) ;MAKE IT 7-BIT ASCII
10102 053160 022627 000021    CMP     (SP)+,#21    ;IS IT A CONTROL-Q?
10103 053164 001366          BNE     2$          ;IF NOT DISCARD IT
10104 053166 000750          BR      1$          ;YES, RESUME
10105 053170 026627 000004 000140    3$:      CMP     4(SP),#140 ;IS IT UPPER CASE?
10106 053176 002407          BLT     4$          ;BRANCH IF YES
10107 053200 026627 000004 000175    CMP     4(SP),#175 ;IS IT A SPECIAL CHAR?
10108 053206 003003          BGT     4$          ;BRANCH IF YES
10109 053210 042766 000040 000004    BIT     #40,4(SP)   ;MAKE IT UPPER CASE
10110 053216 000002          RTI          ;GO BACK TO USER
10111          ;*****
10112          ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
10113          ;CALL:
10114          ;*
10115          ;*      RDLIN          ;INPUT A STRING FROM THE TTY
10116          ;*      RETURN HERE    ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
10117          ;*                    ;TERMINATOR WILL BE A BYTE OF ALL 0'S
10118          ;*
10119 053220 010346          $RDLIN: MOV      R3,-(SP) ;SAVE R3

```

10119	053222	005046			CLR	-(SP)	;; CLEAR THE RUBOUT KEY
10120	053224	012703	053454	1\$:	MOV	#\$TTYIN,R3	;; GET ADDRESS
10121	053230	022703	053505	2\$:	CMP	#\$TTYIN+25.,R3	;; BUFFER FULL?
10122	053234	101456			BLOS	4\$;; BR IF YES
10123	053236	104402			ROCHR		;; GO READ ONE CHARACTER FROM THE TTY
10124	053240	112613			MOVW	(SP)+,(R3)	;; GET CHARACTER
10125	053242	122713	000177	10\$:	CMPB	#177,(R3)	;; IS IT A RUBOUT
10126	053246	001022			BNE	5\$;; BR IF NO
10127	053250	005716			TST	(SP)	;; IS THIS THE FIRST RUBOUT?
10128	053252	001007			BNE	6\$;; BR IF NO
10129	053254	112737	000134 053452		MOVW	#'\,9\$;; TYPE A BACK SLASH
10130	053262	104401	053452		TYPE	9\$	
10131	053266	012716	177777		MOV	#-1,(SP)	;; SET THE RUBOUT KEY
10132	053272	005303		6\$:	DEC	R3	;; BACKUP BY ONE
10133	053274	020327	053454		CMP	R3, \$TTYIN	;; STACK EMPTY?
10134	053300	103434			BLO	4\$;; BR IF YES
10135	053302	111337	053452		MOVW	(R3),9\$;; SETUP TO TYPEOUT THE DELETED CHAR.
10136	053306	104401	053452		TYPE	9\$;; GO TYPE
10137	053312	000746			BR	2\$;; GO READ ANOTHER CHAR.
10138	053314	005716		5\$:	TST	(SP)	;; RUBOUT KEY SET?
10139	053316	001406			BEQ	7\$;; BR IF NO
10140	053320	112737	000134 053452		MOVW	#'\,9\$;; TYPE A BACK SLASH
10141	053326	104401	053452		TYPE	9\$	
10142	053332	005016			CLR	(SP)	;; CLEAR THE RUBOUT KEY
10143	053334	122713	000025	7\$:	CMPB	#25,(R3)	;; IS CHARACTER A CTRL U?
10144	053340	001003			BNE	8\$;; BR IF NO
10145	053342	104401	053505		TYPE	\$CNTLU	;; TYPE A CONTROL "U"
10146	053346	000726			BR	1\$;; GO START OVER
10147	053350	122713	000022	8\$:	CMPB	#22,(R3)	;; IS CHARACTER A "↑R"?
10148	053354	001011			BNE	3\$;; BRANCH IF NO
10149	053356	105013			CLRB	(R3)	;; CLEAR THE CHARACTER
10150	053360	104401	001165		TYPE	\$SCRLF	;; TYPE A "CR" & "LF"
10151	053364	104401	053454		TYPE	\$TTYIN	;; TYPE THE INPUT STRING
10152	053370	000717			BR	2\$;; GO PICKUP ANOTHER CHARACTER
10153	053372	104401	001164	4\$:	TYPE	\$QUES	;; TYPE A '?'
10154	053376	000712			BR	1\$;; CLEAR THE BUFFER AND LOOP
10155	053400	111337	053452	3\$:	MOVW	(R3),9\$;; ECHO THE CHARACTER
10156	053404	104401	053452		TYPE	9\$	
10157	053410	122723	000015		CMPB	#15,(R3)+	;; CHECK FOR RETURN
10158	053414	001305			BNE	2\$;; LOOP IF NOT RETURN
10159	053416	105063	177777		CLRB	-1(R3)	;; CLEAR RETURN (THE 15)
10160	053422	104401	001166		TYPE	\$LF	;; TYPE A LINE FEED
10161	053426	005726			TST	(SP)+	;; CLEAN RUBOUT KEY FROM THE STACK
10162	053430	012603			MOV	(SP)+,R3	;; RESTORE R3
10163	053432	011646			MOV	(SP)-,(SP)	;; ADJUST THE STACK AND PUT ADDRESS OF THE
10164	053434	016666	000004 000002		MOV	4(SP),2(SP)	;; FIRST ASCII CHARACTER ON IT
10165	053442	012766	053454 000004		MOV	#\$TTYIN,4(SP)	
10166	053450	000002			RTI		;; RETURN
10167	053452	000		9\$:	.BYTE	0	;; STORAGE FOR ASCII CHAR. TO TYPE
10168	053453	000			.BYTE	0	;; TERMINATOR
10169	053454	000031			.BLKB	25.	;; RESERVE 25. BYTES FOR TTY INPUT
10170	053505	136	006525 000012		\$CNTLU:	.ASCIZ /↑U/<15><12>	;; CONTROL "U"
10171	053512	043536	005015 000		\$CNTLG:	.ASCIZ /↑G/<15><12>	;; CONTROL "G"
10172	053517	015	051412 051127		\$MSWR:	.ASCIZ <15><12>/SWR = /	
10173	053524	036440	000040				
10174	053530	020040	042516 020127		\$MNEW:	.ASCIZ / NEW = /	

10175 053536 020075 000
10176 053542

.EVEN
.SBTTL TK INITIALIZE ROUTINE

```
*****
*
* THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD TO ALL THE
* OPERATOR TO COMMUNICATE WITH THE PROGRAM.
*
*CALL
* JSR PC,$TKINT
* RETURN
*****
```

10191 053542 012737 053572 000060 \$TKINT: MOV #STKSRV,TKVEC ;INITIALIZE THE KEY BOARD VECTOR
10192 053550 013737 001452 000062 MOV RKPRI,TKVEC+2 ;LOCK RK06 AND TTY INTERRUPTS
10193 053556 005777 125364 TST @STKB ;CLEAR DONE FLAG
10194 053562 012777 000100 125354 MOV #IE,@STKS ;ENABLE INTERRUPT
10195 053570 000207 RTS PC ;RETURN

.SBTTL TK SERVICE ROUTINE (ONLY CONTROL C IS REGNIZED)

10199 053572 117737 125350 054134 \$TKSRV: MOV @STKB,2\$;PICK UP CHARACTER AND STORE FOR
10200 PRINT OUT
10201 053600 142737 000200 054134 BICB #BIT7,2\$;STRIP THE JUNK
10202 053606 122737 000003 054134 CMPB #3,2\$;IS IT A CONTROL-C
10203 053614 001022 BNE 1\$;NO, PRINT <CHAR>?<15><12>
10204 053616 104401 054141 TYPE ,SCNTLC ;TYPE A CONTROL-C
10205 053622 005077 125316 CLR @STKS ;CLEAR INTERRUPT ENABLE
10206 053626 016637 000002 177776 MOV 2(SP),PS ;REINSTATE PREVIOUS PSW
10207 053634 004737 011614 JSR PC,OPRCMD ;PROCESS OPERATOR COMMAND
10208 053640 013737 001452 177776 MOV RKPRI,PS ;LOAD PSW TO LOCK OUT TTY AND RK06
10209 053646 005777 125274 TST @STKB ;CLEAR DONE FLAG
10210 053652 012777 000100 125264 MOV #IE,@STKS ;ENABLE INTERRUPTS
10211 053660 000002 RTI ;RETURN

10213 053662 122737 000007 054134 1\$: CMPB #7,2\$;CHECK IF CONTROL G
10214 053670 001403 BEQ 5\$;YES, GET NEW SWITCH REGISTER VALUE
10215 053672 104401 054134 TYPE ,2\$;PRINT CHARACTER ?<15><12>
10216 053676 000002 3\$: RTI ;RETURN

10218 053700 022737 000176 001140 5\$: CMP #SWREG,SWR ;CHECK IF SOFTWARE SWITCH REG
10219 053706 001373 BNE 3\$;NO, RETURN
10220 053710 122737 000001 001134 CMPB #1,\$AUTOB ;CHECK IF IN AUTO ACCEPT MODE
10221 053716 001767 BEQ 3\$;YES, RETURN
10222 053720 104401 053512 TYPE ,SCNTLG ;TYPE CONTROL G
10223 053724 104401 053517 TYPE ,SMSWR ;TYPE OLD SWITCH REGISTER

10224 053730 013746 000176 MOV SWREG,-(SP)
10225 053734 004737 052774 JSR PC,BINCCT
10226 053740 104401 053530 TYPE ,SMNEW
10227 053744 005077 125174 CLR @STKS ;GO INTO FLAG MODE
10228 053750 005046 19\$: CLR -(SP) ;CLEAR COUNTER
10229 053752 005046 CLR -(SP) ;THE NEW SWR
10230 053754 105777 125164 7\$: TSTB @STKS ;CHAR THERE?

```

10231 053760 100375          BPL      7$          ;; IF NOT TRY AGAIN
10232
10233 053762 117746 125160    MOVB     2$TKB,-(SP)    ;; PICK UP CHAR
10234 053766 042716 177600    BIC     #1C17,(SP)    ;; MAKE IT 7-BIT ASCII
10235
10236 053772 021627 000025    9$:    CMP     (SP),#25    ;; IS IT A CONTROL-U?
10237 053776 001005          BNE     10$          ;; BRANCH IF NOT
10238 054000 104401 053505          TYPE    .SCNTLU       ;; YES, ECHO CONTROL-U (↑U)
10239 054004 062706 000006    20$:   ADD     #6,SP        ;; IGNORE PREVIOUS INPUT
10240 054010 000757          BR      19$          ;; LET'S TRY IT AGAIN
10241
10242
10243 054012 021627 000015    10$:   CMP     (SP),#15    ;; IS IT A <CR>?
10244 054016 001016          BNE     16$          ;; BRANCH IF NO
10245 054020 005766 000004          TST     4(SP)        ;; YES, IS IT THE FIRST CHAR?
10246 054024 001403          BEQ     11$          ;; BRANCH IF YES
10247 054026 016677 000002 125104  MOV     2(SP),2$SWR   ;; SAVE NEW SWR
10248 054034 062706 000006    11$:   ADD     #6,SP        ;; CLEAR UP STACK
10249 054040 104401 001165    14$:   TYPE    $CRLF      ;; ECHO <CR> AND <LF>
10250 054044 012777 000100 125072  MOV     #100,2$TKS   ;; RE-ENABLE TTY KBD INTERRUPTS
10251 054052 000002          RTI                    ;; RETURN
10252 054054 004737 052724    15$:   JSR     PC,$TYPEPC   ;; ECHO CHAR
10253 054060 021627 000060    16$:   CMP     (SP),#60    ;; CHAR < 0?
10254 054064 002420          BLT     18$          ;; BRANCH IF YES
10255 054066 021627 000067          CMP     (SP),#67    ;; CHAR > 7?
10256 054072 003015          BGT     18$          ;; BRANCH IF YES
10257 054074 042726 000060          BIC     #60,(SP)+    ;; STRIP-OFF ASCII
10258 054100 005766 000002          TST     2(SP)        ;; IS THIS THE FIRST CHAR
10259 054104 001403          BEQ     17$          ;; BRANCH IF YES
10260 054106 006316          ASL     (SP)         ;; NO, SHIFT PRESENT
10261 054110 006316          ASL     (SP)         ;; CHAR OVER TO MAKE
10262 054112 006316          ASL     (SP)         ;; ROOM FOR NEW ONE.
10263 054114 005266 000002    17$:   INC     2(SP)        ;; KEEP COUNT OF CHAR
10264 054120 056616 177776          BIS     -2(SP),(SP)  ;; SET IN NEW CHAR
10265 054124 000713          BR      7$          ;; GET THE NEXT ONE
10266 054126 104401 001164    18$:   TYPE    $QUES      ;; TYPE ?<CR><LF>
10267 054132 000724          BR      20$          ;; SIMULATE CONTROL-U
10268
10269 054134 000          2$:    .BYTE  0            ;; STORAGE FOR QUESTIONABLE INPUT
10270 054135 077 005015 000          .ASCIZ  /?/<15><12>
10271 054141 136 006503 000012  $CNTLC: .ASCIZ /↑C/<15><12> ;CONTROL-C

```

.SBTTL OCTAL TO BINARY CONVERSION ROUTINE

```

*****
*
* THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
* WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL
* IT WILL GENERATE TWO BINARY WORDS PLACING THE LOW 16 BITS
* ON THE STACK AND THE HIGH 16 BITS IN LOCATION $HIOCT.
*
*CALL
* MOV     <ADDRESS OF ASCII STRING>,-(SP)
* JSR     PC,OCTBIN
* <ADDRESS OF ERROR RETURN>
* RETURN

```

```

10272
10273
10274
10275
10276
10277
10278
10279
10280
10281
10282
10283
10284
10285
10286

```

```

10287
10288
10289
10290 054146 010046
10291 054150 010146
10292 054152 010246
10293 054154 016600 000010
10294 054160 005001
10295 054162 005002
10296 054164 112046
10297 054166 001423
10298 054170 121627 000054
10299 054174 001420
10300 054176 122716 000060
10301 054202 003030
10302 054204 122716 000067
10303 054210 002425
10304 054212 006301
10305 054214 006102
10306 054216 006301
10307 054220 006102
10308 054222 006301
10309 054224 006102
10310 054226 042716 177770
10311 054232 062601
10312 054234 000753
10313 054236 005726
10314 054240 010166 000010
10315 054244 010237 054300
10316 054250 012602
10317 054252 012601
10318 054254 012600
10319 054256 062716 000002
10320 054262 000207
10321
10322 054264 005726
10323 054266 012602
10324 054270 012601
10325 054272 012600
10326 054274 013616
10327 054276 000207
10328 054300 000000
10329
10330
10331
10332
10333
10334
10335
10336
10337
10338
10339
10340
10341
10342

```

```

;*
;*****
OCTBIN: MOV R0,-(SP) ;SAVE R0
MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
MOV 10(SP),R0 ;GET ADDRESS OF ASCII STRING
CLR R1 ;CLEAR DATA WORDS
CLR R2
2$: MOV8 (R0)+,-(SP) ;PICK THIS CHARACTER
BEQ 3$ ;IF ZERO GET OUT
CMPB (SP),#', ;CHECK IF COMMA
BEQ 3$ ;IF COMMA GET OUT
CMPB #'0,(SP) ;MAKE SURE THIS CHARACTER IS
BGT 4$ ; AN OCTAL DIGIT
CMPB #'7,(SP)
BLT 4$
ASL R1 ; *2
ROL R2
ASL R1 ; *4
ROL R2
ASL R1 ; *8
ROL R2
BIC #1C7,(SP) ;STRIP THE ASCII JUNK
ADD (SP)+,R1 ;ADD THIS DIGIT
BR 2$ ;LOOP
3$: TST (SP)+ ;CLEAN PARTIAL FROM STACK
MOV R1,10(SP) ;SAVE RESULT
MOV R2,$HIOCT
MOV (SP)+,R2 ;RESTORE R2
MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R0 ;RESTORE R0
ADJ #2,(SP) ;ADJUST RETURN
RTS PC ;RETURN
4$: TST (SP)+ ;CLEAN UP PARTIAL FROM STACK
MOV (SP)+,R2 ;RESTORE R2
MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R0 ;RESTORE R0
MOV 2(SP)+,(SP) ;PUT ADDRESS OF ERROR ROUTINE ON STACK
RTS PC ;GO PROCESS ERROR
$HIOCT: .WORD 0 ;HIGH ORDER BITS GO HERE
.SBTL DECIMAL TO BINARY CONVERSION ROUTINE
;*****
;
; THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
; WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL,
; IT WILL GENERATE A BINARY WORD PLACING IT ON THE STACK.
;
;CALL
; MOV <ADDRESS OF ASCII STRING>,-(SP)
; JSR PC,DECBIN
; <ADDRESS OF ERROR RETURN>
; RETURN
;

```



```

10343                                     ;*****
10344                                     ;*****
10345 054302 010046 DECBIN: MOV      RO,-(SP)      ;SAVE RO
10346 054304 010146      MOV      R1,-(SP)      ;SAVE R1
10347 054306 010246      MOV      R2,-(SP)      ;SAVE R2
10348 054310 016600 000010      MOV      10(SP),RO      ;GET ADDRESS OF ASCII STRING
10349 054314 005046      CLR      -(SF)      ;CLEAR DATA WORD
10350 054316 005002      CLR      R2      ;SIGN SET POSITIVE
10351 054320 122710 000055      CMPB   #'-',(RO)      ;SEE IF A MINUS SIGN
10352 054324 001001      BNE    2$      ;BRANCH IF NO MINUS SIGN
10353 054326 112002      MOVB   (RO)+,R2      ;SAVE FOR LATER USE
10354 054330 112001      2$:  MOVB   (RO)+,R1      ;PICKUP THIS CHARACTER
10355 054332 001427      BEQ    3$      ;GET OUT IF ZERO
10356 054334 120127 000054      CMPB   R1,#',      ;CHECK IF COMMA
10357 054340 001424      BEQ    3$      ;GET OUT IF COMMA
10358 054342 122701 000060      CMPB   #'0,R1      ;MAKE SURE THIS CHARACTER IS
10359 054346 003034      BGT    5$      ; A DIGIT BETWEEN 0 & 9
10360 054350 122701 000071      CMPB   #'9,R1
10361 054354 002431      BLT    5$
10362 054356 032716 170000      BIT    #170000,(SP) ;DON'T LET NUMBER GET TO BIG
10363 054362 001026      BNE    5$      ;BRANCH IF NUMBER WOULD OVERFLOW
10364 054364 006316      ASL   (SP)      ; *2
10365 054366 011646      MOV   (SP),-(SP) ;SAVE FOR LATER
10366 054370 006316      ASL   (SP)      ; *4
10367 054372 006316      ASL   (SP)      ; *8
10368 054374 062616      ADD   (SP)+,(SP) ; *10
10369 054376 102420      BVS   5$      ;OVERFLOW ISN'T ALLOWED
10370 054400 162701 000060      SUB   #'0,R1      ;STRIP AWAY THE ASCII JUNK
10371 054404 060116      ADD   R1,(SP)      ;ADD IN THIS DIGIT
10372 054406 102414      BVS   5$      ;OVERFLOW ISN'T ALLOWED
10373 054410 000747      BR    2$      ;LOOP
10374 054412 005702      3$:  TST   R2      ;CHECK IF NUMBER IS NEGATIVE
10375 054414 001401      BEQ   4$      ;BRANCH IF NO
10376 054416 005416      NEG   (SP)      ;YES--NEGATE THE NUMBER
10377 054420 012666 000010      4$:  MOV   (SP)+,10(SP) ;SAVE RESULT
10378 054424 012602      MOV   (SP)+,R2      ;RESTORE R2
10379 054426 012601      MOV   (SP)+,R1      ;RESTORE R1
10380 054430 012600      MOV   (SP)+,RO      ;RESTORE RO
10381 054432 062716 000002      ADD   #2,(SP)      ;ADJUST RETURN
10382 054436 000207      RTS   PC      ;RETURN
10383
10384 054440 005726      5$:  TST   (SP)+      ;CLEAN PARTIAL NUMBER FROM STACK
10385 054442 012602      MOV   (SP)+,R2      ;RESTORE R2
10386 054444 012601      MOV   (SP)+,R1      ;RESTORE R1
10387 054446 012600      MOV   (SP)+,RO      ;RESTORE RO
10388 054450 013616      MOV   2(SP)+,(SP) ;PUT ADDRESS OF ERROR ON STACK
10389 054452 000207      RTS   PC      ;GO PROCESS ERROR
10390
10391 .SBTTL ROUTINE TO SIZE MEMORY
10392
10393 ;*****
10394 ;*CALL: JSR    PC,$SIZE
10395 ;*      RETURN
10396 ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
10397
10398 $SIZE: MOV    RO,-(SP)      ;;SAVE RO ON THE STACK

```

```

10399 054456 010146          MOV      R1, -(SP)          ;; SAVE R1 ON THE STACK
10400 054460 013746 000004    MOV      @#ERRVEC, -(SP)    ;; SAVE PRESENT ERROR VECTOR PS & PC
10401 054464 013746 000006    MOV      @#ERRVEC+2, -(SP)
10402 054470 010600          MOV      SP, R0            ;; SAVE THE STACK POINTER
10403                                     ;; SET THE ERRVEC PS TO THE PRESENT PS
10404 054472 104400          TRAP                                ;; PUSH OLD PSW AND PC ON STACK
10405 054474 012637 000006    MOV      (SP)+, @#ERRVEC+2    ;; SAVE THE PSW IN @#ERRVEC+2
10406 054500 012737 054520 000004    MOV      #25, @#ERRVEC        ;; SET FOR T, MEOUT
10407 054506 012701 020000    MOV      #20000, R1          ;; FIRST ADDRESS
10408 054512 005711          1$: TST      (R1)            ;; TEST THIS ADDRESS
10409 054514 005721          TST      (R1)+              ;; STEP TO NEXT ADDRESS
10410 054516 000775          BR      1$                  ;; TRY ANOTHER
10411 054520 162701 000002    2$: SUB      #2, R1          ;; DROP BACK
10412 054524 010006          MOV      R0, SP            ;; RESTORE THE STACK
10413 054526 012637 000006    MOV      (SP)+, @#ERRVEC+2    ;; RESTORE ERROR VECTOR
10414 054532 012637 000004    MOV      (SP)+, @#ERRVEC
10415 054536 010137 054550    MOV      R1, $LSTAD         ;; LAST ADDRESS
10416 054542 012601          MOV      (SP)+, R1         ;; RESTORE R1
10417 054544 012600          MOV      (SP)+, R0         ;; RESTORE R0
10418 054546 000207          RTS      PC
10419 054550 000000    $LSTAD: .WORD 0            ;; CONTAINS THE LAST ADDRESS
10420                                     ;; *****
10421                                     .SBTTL  POWER DOWN AND UP ROUTINES
10422
10423                                     :POWER DOWN ROUTINE
10424                                     $PWRDN: MOV      @SWR, SAVSWR    ;; SAVE SWITCH REGISTER
10425 054552 017737 12436c 001404    MOV      @#PWRUP, PWRVEC    ;; SET UP VECTOR
10426 054560 012737 054572 000024    HALT
10427 054566 000000          BR      -2                ;; HANG UP
10428 054570 000776
10429
10430                                     :POWER UP ROUTINE
10431 $PWRUP: CLR      $PWRCT      ;; WAIT LOOP FOR THE TTY
10432 054572 005037 054676          MOV      #100, $PWRCT+2
10433 054576 012737 000144 054700    1$: INC      $PWRCT        ;; WAIT FOR THE INCREMENT
10434 054604 005237 054676          BNE     1$                ;; OF WORD
10435 054610 001375          DEC      $PWRCT-2
10436 054612 005337 054700          BNE     1$
10437 054616 001372          MOV      @#PWRDN, PWRVEC    ;; SET POWER DOWN VECTOR
10438 054620 012737 054552 000024    MOV      #PR7, PWRVEC+2    ;; PRIORITY 7
10439 054626 012737 000340 000026    MOV      @#PR7, TRAPVEC+2  ;; LOCK OUT ALL INTERRUPTS FOR TRAPS
10440 054634 012737 000340 00003c    MOV      #STACK, $P        ;; INITIALIZE STACK
10441 054642 012706 001100          TYPE     $POWER           ;; REPORT POWER FAIL
10442 054646 104401 054702          JSR     PC, PRITIM        ;; PRINT TIME
10443 054652 004737 043670          MCVB    #-1, FLAG         ;; SET FLAG FOR RESTART SEQUENCE
10444 054656 112737 177777 001361    MOV      SAVSWR, @SWR      ;; RESTORE SWITCH REGISTER
10445 054664 013777 001404 124246    JMP     PWSRT             ;; RESTART PROGRAM
10446 054672 000137 007172
10447 054676 000000 000000    $PWRCT: .WORD 0, 0        ;; WAIT COUNT FOR TTY
10448 054702 005015 047520 042527    $POWER: .ASCIZ <15><12>/POWER/<15><12>
10449 054710 006522 000012
10450                                     .EVEN
10451
10452                                     .SBTTL  INTEGER MULTIPLY ROUTINE
10453
10454                                     ;; *****

```

```

10455
10456
10457
10458
10459
10460
10461
10462
10463
10464
10465
10466
10467
10468
10469
10470 054714
10471 054714 010046
10472 054716 010146
10473 054720 010246
10474 054722 010346
10475 054724 010446
10476 054726 010546
10477 054730 016605 000016
10478 054734 016603 000020
10479 054740 005000
10480 054742 005001
10481 054744 005002
10482 054746 005004
10483 054750 012746 000041
10484 054754 006000
10485 054756 006001
10486 054760 006002
10487 054762 006003
10488 054764 103003
10489 054766 060501
10490 054770 005500
10491 054772 060400
10492 054774 005316
10493 054776 001366
10494 055000 005726
10495 055002 010266 000020
10496 055006 010366 000016
10497 055012 012605
10498 055014 012604
10499 055016 012603
10500 055020 012602
10501 055022 012601
10502 055024 012600
10503 055026 000207
10504
10505
10506
10507
10508
10509
10510

```

```

*CALL
*   MOV    MULTIPLIER, -(SP)
*   MOV    MULTIPLICAND, -(SP)
*   JSR    PC, $MULT
*   RETURN                                ;; PRODUCT IS ON THE STACK
*
*   STACK  PRODUCT
*   -----
*
*   TOP    LSB'S
*   +2     MSB'S
*
* *****
$MULT:
MOV    R0, -(SP)                ;; PUSH R0 ON STACK
MOV    R1, -(SP)                ;; PUSH R1 ON STACK
MOV    R2, -(SP)                ;; PUSH R2 ON STACK
MOV    R3, -(SP)                ;; PUSH R3 ON STACK
MOV    R4, -(SP)                ;; PUSH R4 ON STACK
MOV    R5, -(SP)                ;; PUSH R5 ON STACK
MOV    16(SP), R5                ;; STORE MULTIPLICAND
MOV    20(SP), R3                ;; STORE MULTIPLIER
CLR    R0                        ;; CLEAR HIGH ORDER WORDS
CLR    R1
CLR    R2
CLR    R4
MOV    #41, -(SP)                ;; MOVE 33 DEC TO COUNTER
1$:   ROR    R1
ROR    R2                        ;; SHIFT TO ADD
ROR    R3
BCC    R5, R1                    ;; NO CARRY NO ADD
ADD    R0, R1                    ;; ADD DOUBLE PRECISION TO OBTAIN NEW PARTIAL
ADD    R4, R0                    ;; PRODUCT
2$:   DEC    (SP)                ;; DECREMENT COUNTER
BNE    1$
TST    (SP)+                    ;; REMOVE COUNTER
MOV    R2, 20(SP)                ;; GET LEAST SIGNIFICANT BITS
MOV    R3, 16(SP)                ;; GET MOST SIGNIFICANT BITS
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
RTS    PC                        ;; RETURN

```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

```

* *****
* THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
* WITH A RANGE OF 0 TO 2(+33)-1.

```

```

10511
10512
10513
10514
10515
10516
10517 055030
10518 055030 010046
10519 055032 010146
10520 055034 010246
10521 055036 013700 055130
10522 055042 013701 055126
10523 055046 012702 177771
10524 055052 006300
10525 055054 006101
10526 055056 005202
10527 055060 001374
10528 055062 063700 055130
10529 055066 005501
10530 055070 063701 055126
10531 055074 062700 001057
10532 055100 005501
10533 055102 062701 047401
10534 055106 010037 055130
10535 055112 010137 055126
10536 055116 012602
10537 055120 012601
10538 055122 012600
10539 055124 000207
10540 055126 176543
10541 055130 123456
10542
10543
10544
10545
10546
10547
10548
10549
10550
10551
10552
10553
10554
10555
10556
10557
10558
10559
10560 055132
10561 055132 010046
10562 055134 010146
10563 055136 010246
10564 055140 010346
10565 055142 010446
10566 055144 010546

```

```

; *CALL:
; * JSR PC,$RAND ;: CALL THE ROUTINE
; * RETURN ;: RETURN HERE THE RANDOM
; * ;: NUME'R WILL BE IN
; * ;: $HINUM,$LONUM

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

```

.SBTTL SINGLE/DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINES

```

; *****
; * THESE ROUTINE WIL CONVERT A 16-BIT OR 32-BIT BINARY NUMBER
; * TO AN UNSIGNED DECIMAL (ASCII) NUMBER.
; *
; *CALL MOV #PNTR,-(SP) ;: POINTER TO LOW WORD OF BINARY NUMBER
; * JSR PC,$DB2D ;: DECIMAL NUMBER IN $DECVL
; * RETURN
; *
; *CALL MOV NUM,-(SP)
; * JSR PC,$SB2D ;: DECMAL NUMBER IN $DECVL
; * RETURN
; *
; *****

```

```

$SB2D:
MOV RO,-(SP) ;: PUSH RO ON STACK
MOV R1,-(SP) ;: PUSH R1 ON STACK
MOV R2,-(SP) ;: PUSH R2 ON STACK
MOV R3,-(SP) ;: PUSH R3 ON STACK
MOV R4,-(SP) ;: PUSH R4 ON STACK
MOV R5,-(SP) ;: PUSH R5 ON STACK

```

```

10567 055146 016601 000016      MOV      16(SP),R1      ;STORE NUMBER IN R1
10568 055152 005002              CLR      R2            ;CLEAR MOST SIGNIFICANT BITS
10569 055154 012737 000005 055300  MOV      #5,$DECNT     ;SET UP FOR 5 CONVERSIONS
10570 055162 012704 055356      MOV      #STNPW,R4     ;ADDRESS FOR 5 POWER
10571 055166 012705 055360      MOV      #STNPW+2,R5
10572 055172 000421              BR       $B2D          ;START CONVERSION
10573
10574 055174              $DB2D:
10575 055174 010046      MOV      R0,-(SP)     ;;PUSH R0 ON STACK
10576 055176 010146      MOV      R1,-(SP)     ;;PUSH R1 ON STACK
10577 055200 010246      MOV      R2,-(SP)     ;;PUSH R2 ON STACK
10578 055202 010346      MOV      R3,-(SP)     ;;PUSH R3 ON STACK
10579 055204 010446      MOV      R4,-(SP)     ;;PUSH R4 ON STACK
10580 055206 010546      MOV      R5,-(SP)     ;;PUSH R5 ON STACK
10581 055210 016602 000016      MOV      16(SP),R2     ;PICKUP DATA POINTER
10582 055214 012201      MOV      (R2)+,R1     ;PICKUP BINARY NUMBER
10583 055216 011202              MOV      (R2),R2
10584 055220 012737 000012 055300  MOV      #10,$DECNT    ;SET UP TO DO 10 CONVERSIONS
10585 055226 012704 055332      MOV      #STNPWR,R4   ;ADDRESS OF TEN POWER
10586 055232 012705 055334      MOV      #STNPWR+2,R5
10587
10588 055236 012700 055402      $B2D:  MOV      #SDECVL,R0 ;GET ADDRESS OF "$DECVL" STRING
10589 055242 005003      $2DEC: CLR      R3      ;CLEAR PARTIAL
10590 055244 161401      2$:    SUB      (R4),R1     ;SUBTRACT TEN POWER
10591 055246 005602              SBC      R2
10592 055250 161502              SUB      (R5),R2
10593 055252 002402              BLT      3$          ;BRANCH IF TEN POWER TOO LARGE
10594 055254 005203              INC      R3          ;ADD 1 TO PARTIAL
10595 055256 000772              BR       2$          ;LOOP
10596
10597 055260 062401      3$:    ADD      (R4)+,R1     ;RESTORE SUBTRACTED VALUE
10598 055262 005502              ADC      R2
10599 055264 062402              ADD      (R4)+,R2
10600 055266 022525              CMP      (R5)+,(R5)+ ;MOVE TO NEXT TEN POWER
10601 055270 052703 000060      BIS      #0,R3        ;CHANGE IT TO ASCII
10602 055274 110320              MOV      R3,(R0)+    ;SAVE IT
10603 055276 005327              DEC      (PC)+       ;DONE?
10604 055300 000000      $DECNT: .WORD      0
10605 055302 001357              BNE      $2DEC       ;BRANCH IF NO
10606 055304 105020              CLRB    (R0)+        ;TERMINATOR
10607 055306 012605              MOV      (SP)+,R5    ;;POP STACK INTO R5
10608 055310 012604              MOV      (SP)+,R4    ;;POP STACK INTO R4
10609 055312 012603              MOV      (SP)+,R3    ;;POP STACK INTO R3
10610 055314 012602              MOV      (SP)+,R2    ;;POP STACK INTO R2
10611 055316 012601              MOV      (SP)+,R1    ;;POP STACK INTO R1
10612 055320 012600              MOV      (SP)+,R0    ;;POP STACK INTO R0
10613 055322 012616 055402      MOV      (SP)+,(SP)  ;SETUP STACK FOR RETURN
10614 055324 104401              TYPE    $DECVL
10615 055330 000207              RTS     PC           ;RETURN
10616
10617 055332 145000      $TNPWR: 145000        ;1.0E09
10618 055334 035632              35632
10619 055336 160400              160400        ;1.0E08
10620 055340 002765              2765
10621 055342 113200              113200        ;1.0E07
10622 055344 000230              230

```

10623	055346	041100			041100		;1.0E06
10624	055350	000017			17		
10625	055352	103240			103240		;1.0E05
10626	055354	000001			1		
10627	055356	023420			23420		;1.0E04
10628	055360	000000			0		
10629	055362	001750			1750		;1.0E03
10630	055364	000000			0		
10631	055366	000144			144		;1.0E02
10632	055370	000000			0		
10633	055372	000012			12		;1.0E01
10634	055374	000000			0		
10635	055376	000001			1		;1.0E00
10636	055400	000000			0		
10637	055402	000014			14		
10638							
10639							
10640							
10641	055416	112737	000001	055662	\$ATY1: MOVB #1,\$FFLG		;;TO REPORT FATAL ERROR
10642	055424	112737	000001	055660	\$ATY3: MOVB #1,\$MFLG		;;TO TYPE A MESSAGE
10643	055432	000403			BR \$ATYC		
10644	055434	112737	000001	055662	\$ATY4: MOVB #1,\$FFLG		;;TO ONLY REPORT FATAL ERROR
10645	055442				\$ATYC:		
10646	055442	010046			MOV RO,-(SP)		;;PUSH RO ON STACK
10647	055444	010146			MOV R1,-(SP)		;;PUSH R1 ON STACK
10648	055446	105737	055660		TSTB \$MFLG		;;SHOULD TYPE A MESSAGE?
10649	055452	001450			BEQ 5\$;;IF NOT: BR
10650	055454	122737	000001	001210	CMPB #APTENV,\$ENV		;;OPERATING UNDER APT?
10651	055462	001031			BNE 3\$;;IF NOT: BR
10652	055464	132737	000100	001211	BITB #APTPOOL,\$ENVM		;;SHOULD SPOOL MESSAGES?
10653	055472	001425			BEQ 3\$;;IF NOT: BR
10654	055474	017600	000004		MOV #4(SP),RO		;;GET MESSAGE ADDR.
10655	055500	062766	000002	000004	ADD #2,4(SP)		;;BUMP RETURN ADDR.
10656	055506	005737	001170		1\$: TST \$MSGTYPE		;;SEE IF DONE W/ LAST XMISSION?
10657	055512	001375			BNE 1\$;;IF NOT: WAIT
10658	055514	010037	001204		MOV RO,\$MSGAD		;;PUT ADDR IN MAILBOX
10659	055520	105720			2\$: TSTB (RO)+		;;FIND END OF MESSAGE
10660	055522	001376			BNE 2\$		
10661	055524	163700	001204		SUB \$MSGAD,RO		;;SUB START OF MESSAGE
10662	055530	006200			ASR RO		;;GET MESSAGE LNTH IN WORDS
10663	055532	010037	001206		MOV RO,\$MSGLGT		;;PUT LENGTH IN MAILBOX
10664	055536	012737	000004	001170	MOV #4,\$MSGTYPE		;;TELL APT TO TAKE MSG.
10665	055544	000413			BR 5\$		
10666	055546	017637	000004	055572	3\$: MOV #4(SP),4\$;;PUT MSG ADDR IN JSR LINKAGE
10667	055554	062766	000002	000004	ADD #2,4(SP)		;;BUMP RETURN ADDRESS
10668	055562	013746	177776		MOV 177776,-(SP)		;;PUSH 177776 ON STACK
10669	055566	004737	052512		JSR PC,\$TYPE		;;CALL TYPE MACRO
10670	055572	000000			4\$: .WORD 0		
10671	055574				5\$:		
10672	055574	105737	055662		10\$: TSTB \$FFLG		;;SHOULD REPORT FATAL ERROR?
10673	055600	001416			BEQ 12\$;;IF NOT: BR
10674	055602	005737	001210		TST \$ENV		;;RUNNING UNDER APT?
10675	055606	001413			BEQ 12\$;;IF NOT: BR
10676	055610	005737	001170		11\$: TST \$MSGTYPE		;;FINISHED LAST MESSAGE?
10677	055614	001375			BNE 11\$;;IF NOT: WAIT
10678	055616	017637	000004	001172	MOV #4(SP),\$FATAL		;;GET ERROR #

\$DECVL: .BLK2 12. ;RESERVE STORAGE FOR ASCII STRING
.SBTTL APT COMMUNICATIONS ROUTINE

10679 055624 062766 000002
 10680 055632 005237 001170
 10681 055636 105037 055662
 10682 055642 105037 055661
 10683 055646 105037 055660
 10684 055652 012601
 10685 055654 012600
 10686 055656 000207
 10687 055660 000
 10688 055661 000
 10689 055662 000
 10690 055664
 10691 000200
 10692 001
 10693 100
 10694 00040
 10695
 10696
 10697
 10698
 10699
 10700
 10701
 10702
 10703 055664 010046
 10704 055666 016600 000002
 10705 055672 005740
 10706 055674 111000
 10707 055676 006300
 10708 055700 016000 055720
 10709 055704 000200
 10710
 10711
 10712
 10713
 10714 055706 011646
 10715 055710 016666 000004 000002
 10716 055716 000002
 10717
 10718
 10719
 10720
 10721
 10722
 10723
 10724
 10725 055720 055706
 10726 055722 052512
 10727
 10728
 10729 055724 053100
 10730 055726 053220

```

000004      ADD      #2,4(SP)          ;; BUMP RETURN ADDR.
            INC      $MSGTYPE        ;; TELL APT TO TAKE ERROR
12$:        CLR      $FFLG          ;; CLEAR FATAL FLAG
            CLR      $LFLG          ;; CLEAR LOG FLAG
            CLR      $MFLG          ;; CLEAR MESSAGE FLAG
            MOV      (SP)+,R1        ;; POP STACK INTO R1
            MOV      (SP)+,RO        ;; POP STACK INTO RO
            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     TRAP DECODER

; *****
; *THIS ROUT. E WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
; *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
; *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
; *GO TO THAT ROUTINE.
$TRAP:      MOV      RO,-(SP)         ;; SAVE RO
            MOV      2(SP),RO        ;; GET TRAP ADDRESS
            TST      -(RO)           ;; BACKUP BY 2
            MOVB     (RO),RO         ;; GET RIGHT BYTE OF TRAP
            ASL      RO              ;; POSITION FOR INDEXING
            MOV      $TRPAD(RO),RO   ;; INDEX TO TABLE
            RTS      RO              ;; GO TO ROUTINE

;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
$TRAP2:     MOV      (SP),-(SP)       ;; MOVE THE PC DOWN
            MOV      4(SP),2(SP)     ;; MOVE THE PSW DOWN
            RTI                      ;; RESTORE THE PSW

.SBTTL     TRAP TABLE

; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; *BY THE "TRAP" INSTRUCTION.
;
; ROUTINE
;-----
$TRPAD:     .WORD   $TRAP2           TRAP+1(104401)  TTY TYPEOUT ROUTINE
            .TYPE   ;;CALL=TYPE
$RDCHR     ;;CALL=RDCHR           TRAP+2(104402)  TTY TYPEIN CHARACTER ROUTINE
$RDLIN     ;;CALL=RDLIN           TRAP+3(104403)  TTY TYPEIN STRING ROUTINE

```

```

10731
10732
10733 055730 005015 044103 047101 .SBTTL ASCII MESSAGES
10734 055736 042507 054040 042130 XXDPMG: .ASCII <CR><LF>/CHANGE XXDP PACK/
10735 055744 020120 040520 045503
10736 055752 005015 046103 040505 .ASCIZ <CR><LF>/CLEAR LOC 40,RESTART PROGRAM/
10737 055760 020122 047514 020103
10738 055766 030064 051054 051505
10739 055774 040524 052122 050040
10740 056002 047522 051107 046501
10741 056010 000
10742 056011 111 046114 043505 ILLCOM: .ASCII /ILLEGAL OPERATOR COMMAND /
10743 056016 046101 047440 042520
10744 056024 040522 047524 020122
10745 056032 047503 046515 047101
10746 056040 020104
10747 056042 020040 006477 000012 ILLCMD: .ASCIZ / ?/<15><12>
10748 056050 051104 053111 020105 OPRO00: .ASCII /DRIVE NUMBER /
10749 056056 052516 041115 051105
10750 056064 040
10751 056065 060 000040 OPRO01: .ASCIZ /0 /
10752 056070 046111 042514 040507 OPRO02: .ASCIZ /ILLEGAL?/<15><12>
10753 056076 037514 005015 000
10754 056103 125 042116 051105 OPRO03: .ASCIZ /UNDER TEST/
10755 056110 052040 051505 000124
10756 056116 046101 042522 042101 OPRO04: .ASCIZ /ALREADY ASSIGNED?/<15><12>
10757 056124 020131 051501 044523
10758 056132 047107 042105 006477
10759 056140 000012
10760 056142 047516 020124 051501 OPRO05: .ASCIZ /NOT ASSIGNED?/<15><12>
10761 056150 044523 047107 042105
10762 056156 006477 000012
10763 056162 040510 020123 042502 OPRO06: .ASCIZ /HAS BEEN DROPPED FROM TEST SEQUENCE/<15><12>
10764 056170 047105 042040 047522
10765 056176 050120 042105 043040
10766 056204 047522 020115 042524
10767 056212 052123 051440 050505
10768 056220 042525 041516 006505
10769 056226 000012
10770 056230 046120 040505 042523 OPRO07: .ASCIZ /PLEASE ENTER COMMAND/<15><12> /*/
10771 056236 042440 052116 051105
10772 056244 041440 046517 040515
10773 056252 042116 005015 000052
10774 056260 050117 051105 052101 OPRO08: .ASCIZ /OPERATOR INITIATED DROP DRIVE DURING WRITE PACK/
10775 056266 051117 044440 044516
10776 056274 044524 052101 042105
10777 056302 042040 047522 020120
10778 056310 051104 053111 020105
10779 056316 052504 044522 043516
10780 056324 053440 044522 042524
10781 056332 050040 041501 000113
10782 056340 005015 051104 053111 OPRO10: .ASCII <15><12>/DRIVE NUMBER /
10783 056346 020105 052516 041115
10784 056354 051105 040
10785 056357 060 000040 OPRO11: .ASCIZ /0 /
10786 056362 051104 053111 020105 OPRO12: .ASCIZ /DRIVE WRITE LOCKED/<15><12>

```


10787	056370	051127	052111	020105	
10788	056376	047514	045503	042105	
10789	056404	005015	000		
10790	056407	101	046114	042040	OPR013: .ASCIZ /ALL DRIVES CURRENTLY UNDER TEST/<15><12>
10791	056414	044522	042526	020123	
10792	056422	052503	051122	047105	
10793	056430	046124	020131	047125	
10794	056436	042504	020122	042524	
10795	056444	052123	005015	000	
10796	056451	116	020117	051104	OPR014: .ASCIZ /NO DRIVE IN USE/<15><12>
10797	056456	053111	020105	047111	
10798	056464	052440	042523	005015	
10799	056472	000			
10800	056473	101	046114	043511	OPR015: .ASCIZ /ALIGNMENT PACK IN DRIVE/<15><12>
10801	056500	046516	047105	020124	
10802	056506	040520	045503	044440	
10803	056514	020116	051104	053111	
10804	056522	006505	040012		
10805	056526	025052	051104	053111	DRVSTT: .ASCIZ /**DRIVE STATISTICS**/<15><12>
10806	056534	020105	052123	052101	
10807	056542	051511	044524	051503	
10808	056550	025052	005015	000	
10809	056555	015	042012	044522	STT000: .ASCII <15><12>/DRIVE SERIAL NO. /
10810	056562	042526	051440	051105	
10811	056570	040511	020114	047516	
10812	056576	020056			
10813	056600	000003			STT100: .BLKB 3
10814	056603	000			.BYTE 0
10815	056604	051117	042504	051522	STT001: .ASCIZ /ORDERS PERFORMED /
10816	056612	050040	051105	047506	
10817	056620	046522	042105	000040	
10818	056626	005015	047527	042122	STT002: .ASCIZ <15><12>/WORDS WRITTEN*65K /
10819	056634	020123	051127	052111	
10820	056642	042524	025116	032466	
10821	056650	020113	000		
10822	056653	015	053412	051117	STT003: .ASCIZ <15><12>/WORDS READ*65K /
10823	056660	051504	051040	040505	
10824	056666	025104	032466	020113	
10825	056674	000			
10826	056675	015	051412	043117	STT004: .ASCIZ <15><12>/SOFT DATA ERRORS/<15><12>/ ECC /
10827	056702	020124	040504	040524	
10828	056710	042440	051122	051117	
10829	056716	006523	020012	042440	
10830	056724	041503	000040		
10831	056730	005015	020040	042522	STT005: .ASCIZ <15><12>/ REREAD /
10832	056736	042522	042101	000040	
10833	056744	005015	020040	043117	STT006: .ASCIZ <15><12>/ OFFSET /
10834	056752	051506	052105	000040	
10835	056760	005015	040510	042122	STT007: .ASCIZ <15><12>/HARD DATA ERRORS /
10836	056766	042040	052101	020101	
10837	056774	051105	047522	051522	
10838	057002	000040			
10839	057004	005015	042523	045505	STT008: .ASCIZ <15><12>/SEEK INCOMPLETES /
10840	057012	044440	041516	046517	
10841	057020	046120	052105	051505	
10842	057026	000040			

10843	057030	005015	050117	051105	STT009: .ASCIZ <15><12>"OPERATION INCOMPLETES/MISPOSITIONS "
10844	057036	052101	047511	020116	
10845	057044	047111	047503	050115	
10846	057052	042514	042524	027523	
10847	057060	044515	050123	051517	
10848	057066	052111	047511	051516	
10849	057074	000040			
10850	057076	005015	052117	042510	STT010: .ASCIZ <15><12>/OTHER ERRORS /
10851	057104	020122	051105	047522	
10852	057112	051522	000040		
10853	057116	005015	040520	J45503	STT011: .ASCII <15><12>/PACK SERIAL NUMBER /
10854	057124	051440	051105	040511	
10855	057132	020114	052516	041115	
10856	057140	051105	040		
10857	057143	000013			STT012: .BLKB 11.
10858	057156	005015	000		STT013: .ASCIZ <15><12>
10859	057161	103	052520	044440	SYS000: .ASCIZ /CPU ID=/ /
10860	057166	036504	000		
10861	057171	117	042526	046122	SYS001: .ASCIZ /OVERLAY LOADER=/ /
10862	057176	054501	046040	040517	
10863	057204	042504	036522	000	
10864	057211	115	054101	052040	SYS002: .ASCIZ /MAX TRANSFER SIZE (MAX: /
10865	057216	040522	051516	042506	
10866	057224	020122	044523	042532	
10867	057232	024040	046440	054101	
10868	057240	020072	000		
10869	057243	040	036451	000	SYS003: .ASCIZ /)=/ /
10870	057247	116	027117	047440	SYS004: .ASCIZ /NO. OF SOFTWARE COMPARES=/ /
10871	057254	020106	047523	052106	
10872	057262	040527	042522	041440	
10873	057270	046517	040520	042522	
10874	057276	036523	000		
10875	057301	122	030113	026466	SYS005: .ASCIZ /RK06-RK07 BUS ADD=/ /
10876	057306	045522	033460	041040	
10877	057314	051525	040440	042104	
10878	057322	000075			
10879	057324	045522	033060	051055	SYS006: .ASCIZ /RK06-RK07 VEC ADD=/ /
10880	057332	030113	020067	042526	
10881	057340	020103	042101	036504	
10882	057346	000			
10883	057347	116	047117	054105	SYS007: .ASCIZ /NONEXISTENT MEMORY/<15><12>
10884	057354	051511	042524	052116	
10885	057362	046440	046505	051117	
10886	057370	006531	000012		
10887	057374	042522	042101	020131	SYS008: .ASCIZ /READY TO BEGIN PERFORMANCE TESTING/<15><12><12>
10888	057402	047524	041040	043505	
10889	057410	047111	050040	051105	
10890	057416	047506	046522	047101	
10891	057424	042503	052040	051505	
10892	057432	044524	043516	005015	
10893	057440	000012			
10894	057442	052123	052101	051511	SYS009: .ASCIZ /STATISTIC INTERVAL=/ /
10895	057450	044524	020103	047111	
10896	057456	042524	053122	046101	
10897	057464	000075			
10898	057466	045522	033060	051055	SYS010: .ASCIZ /RK06-RK07 PRIOR=/ /

CZR6PCO RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53MACY11 30(1046) 01-DEC-77 15:04 PAGE 243
ASCII MESSAGES

SEQ 0241

10899	057474	030113	020067	051120	
10900	057502	047511	036522	000	
10901	057507	120	051101	042515	PAR000: .ASCII /PARAMETERS FOR DRIVE /
10902	057514	042524	051522	043040	
10903	057522	051117	042040	044522	
10904	057530	042526	040		
10905	057533	060	005015	000012	PAR001: .ASCIZ /0/15>'><12>
10906	057540	054503	044514	042116	PAR002: .ASCIZ /CYLINDER MAX,MIN =/
10907	057546	051105	046440	054101	
10908	057554	046454	047111	036440	
10909	057562	000			
10910	057563	124	040522	045503	PAR003: .ASCIZ /TRACK MAX,MIN =/
10911	057570	046440	054101	046454	
10912	057576	047111	036440	000	
10913	057603	123	041505	047524	PAR004: .ASCIZ /SECTOR MAX,MIN =/
10914	057610	020122	040515	026130	
10915	057616	044515	020116	000075	
10916	057624	042522	042101	053457	PAR005: .ASCIZ "READ/WRITE RATIO ="
10917	057632	044522	042524	051040	
10918	057640	052101	047511	036440	
10919	057646	000			
10920	057647	127	044522	042524	PAR006: .ASCIZ /WRITE CHECK AFTER WRITE =/
10921	057654	041440	042510	045503	
10922	057662	040440	052106	051105	
10923	057670	053440	044522	042524	
10924	057676	036440	000		
10925	057701	103	051117	042522	PAR007: .ASCIZ /CORRECTABLE READ ERROR THRESHOLD =/
10926	057706	052103	041101	042514	
10927	057714	051040	040505	020104	
10928	057722	051105	047522	020122	
10929	057730	044124	042522	044123	
10930	057736	046117	020104	000075	
10931	057744	047125	047503	051122	PAR008: .ASCIZ /UNCORRECTABLE READ ERROR THRESHOLD =/
10932	057752	041505	040524	046102	
10933	057760	020105	042522	042101	
10934	057766	042440	051122	051117	
10935	057774	052040	051110	051505	
10936	060002	047510	042114	036440	
10937	060010	000			
10938	060011	123	042505	020113	PAR009: .ASCIZ /SEEK ERROR THRESHOLD =/
10939	060016	051105	047522	020122	
10940	060024	044124	042522	044123	
10941	060032	046117	020104	000075	
10942	060040	047111	044510	044502	PAR010: .ASCIZ /INHIBIT ERROR CORRECTION AND RETRY =/
10943	060046	020124	051105	047522	
10944	060054	020122	047503	051122	
10945	060062	041505	044524	047117	
10946	060070	040440	042116	051040	
10947	060076	052105	054522	036440	
10948	060104	000			
10949	060105	117	042520	040522	PAR011: .ASCIZ /OPERATION COUNT THRESHOLD*65K =/
10950	060112	044524	047117	041440	
10951	060120	052517	052116	052040	
10952	060126	051110	051505	047510	
10953	060134	042114	033052	045465	
10954	060142	036440	000		

10955	060145	127	051117	051504	PAR012: .ASCIZ	/WORDS TRANSFERRED THRESHOLD*1,049K =/
10956	060152	052040	040522	051516		
10957	060160	042506	051122	042105		
10958	060166	052040	051110	051505		
10959	060174	047510	042114	030452		
10960	060202	030054	034464	020113		
10961	060210	000075				
10962	060212	044506	051522	0C0124	PAR013: .ASCIZ	/FIRST/
10963	060220	042523	047503	042116	PAR014: .ASCIZ	/SECOND/
10964	060226	000				
10965	060227	124	044510	042122	PAR015: .ASCIZ	/THIRD/
10966	060234	000				
10967	060235	106	052517	052122	PAR016: .ASCIZ	/FOURTH/
10968	060242	000110				
10969	060244	044506	052106	000110	PAR017: .ASCIZ	/FIFTH/
10970	060252	042440	041530	052514	PAR018: .ASCIZ	/ EXCLUDED PACK AREA =/
10971	060260	042504	020104	040520		
10972	060266	045503	040440	042522		
10973	060274	020101	000075			
10974	060300	044103	047101	042507	PAR019: .ASCIZ	/CHANGE/
10975	060306	000				
10976	060307	125	040516	046102	PAR020: .ASCII	/UNABLE TO GENERATE NEW COMMAND DUE TO/<15><12>
10977	060314	020105	047524	043440		
10978	060322	047105	051105	052101		
10979	060330	020105	042516	020127		
10980	060336	047503	046515	047101		
10981	060344	020104	052504	020105		
10982	060352	047524	005015			
10983	060356	054105	046103	042125	.ASCIZ	/EXCLUDED PACK AREAS AND MAX TRANSFER SIZE CONFLICT/<15><12>
10984	060364	042105	050040	041501		
10985	060372	020113	051101	040505		
10986	060400	020123	047101	020104		
10987	060406	040515	020130	051124		
10988	060414	047101	043123	051105		
10989	060422	051440	055111	020105		
10990	060430	047503	043116	044514		
10991	060436	052103	005015	000		
10992	060443	123	046501	046120	PAR021: .ASCIZ	/SAMPLED COMPARES =/
10993	060450	042105	041440	046517		
10994	060456	040520	042522	020123		
10995	060464	000075				
10996	06046	042524	052123	047111	PAR022: .ASCIZ	/TESTING RK06? Y OR N /
10997	06047	020107	045522	033060		
10998	060502	020077	020131	051117		
10999	060510	047040	020040	000		
11000	060515	040	037440	037477	QUESMK: .ASCIZ	/ ?????? /
11001	060522	037477	020077	000040		
11002	060530	020040	000		BLANKS: .ASCIZ	/ /
11003	060533	052	025052	040506	ERROR0: .ASCIZ	/***FATAL ERROR***/<15><12><12>
11004	060540	040524	020114	051105		
11005	060546	047522	025122	025052		
11006	060554	005015	0000.2			
11007	060560	047503	052116	047522	ERROR01: .ASCIZ	/CONTROLLER CLEAR DID NOT CLEAR ERROR/
11008	060566	046114	051105	041440		
11009	060574	042514	051101	042040		
11010	060602	042111	047040	052117		

11011	060610	041440	042514	051101	
11012	060616	042440	051122	051117	
11013	060624	000			
11014	060625	116	020117	052101	ERROR2: .ASCIZ /NO ATTENTION IN RKASOF/
11015	060632	042524	052116	047511	
11016	060640	020116	047111	051040	
11017	060646	040513	047523	000106	
11018	060654	047125	054105	042520	ERROR3: .ASCIZ /UNEXPECTED DATA TYPE ERROR/
11019	060662	052103	042105	042040	
11020	060670	052101	020101	054524	
11021	060676	042520	042440	051122	
11022	060704	051117	000		
11023	060707	104	044522	042526	ERROR4: .ASCIZ /DRIVE CLEAR DID NOT RESET ATTENTION/
11024	060714	041440	042514	051101	
11025	060722	042040	042111	047040	
11026	060730	052117	051040	051505	
11027	060736	052105	040440	052124	
11028	060744	047105	044524	047117	
11029	060752	000			
11030	060753	111	046114	043505	ERROR5: .ASCIZ /ILLEGAL DRIVER COMMAND/
11031	060760	046101	042040	044522	
11032	060766	042526	020122	047503	
11033	060774	046515	047101	000104	
11034	061002	052515	052114	050111	ERROR6: .ASCIZ /MULTIPLE DRIVE SELECT/
11035	061010	042514	042040	044522	
11036	061016	042526	051440	046105	
11037	061024	041505	000124		
11038	061030	042523	020124	053523	ERROR7: .ASCIZ /SET SWITCH 14 TO CYCLE ON ERROR/<15><12>
11039	061036	052111	044103	030440	
11040	061044	020064	047524	041440	
11041	061052	041531	042514	047440	
11042	061060	020116	051105	047522	
11043	061066	006522	000012		
11044	061072	047503	052116	030122	ERROR8: .ASCIZ /CONTROLLER ERROR THRESHOLD EXCEEDED/
11045	061100	046114	051105	042440	
11046	061106	051122	051117	052040	
11047	061114	051110	051505	047510	
11048	061122	042114	042440	041530	
11049	061130	042505	042504	000104	
11050	061136	040504	040524	046040	ERROR9: .ASCIZ /DATA LATE THRESHOLD EXCEEDED/
11051	061144	052101	020105	044124	
11052	061152	042522	044123	046117	
11053	061160	020104	054105	042507	
11054	061166	042105	042105	000	
11055	061173	101	042104	042522	ERROR12: .ASCII /ADDRESS GOOD BAD SECTOR DATA/<15><12>
11056	061200	051523	043440	047517	
11057	061206	020104	020040	041040	
11058	061214	042101	020040	020040	
11059	061222	051440	041505	047524	
11060	061230	020122	042040	052101	
11061	061236	006501	012		
11062	061241	040	020040	020040	.ASCIZ / DATA DATA POS PAT #/<15><12>
11063	061246	020040	042040	052101	
11064	061254	020101	020040	042040	
11065	061262	052101	020101	020040	
11066	061270	050040	051517	020040	

11067	061276	020040	050040	052101	
11068	061304	021440	005015	000	
11069	061311	103	047117	051124	ERR013: .ASCIZ /CONTROLLER ERROR NOT FLAGGED/<15><12>
11070	061316	046117	042514	020122	
11071	061324	051105	047522	020122	
11072	061332	047516	020124	046106	
11073	061340	043501	042507	006504	
11074	061346	000012			
11075	061350	047527	042122	041440	ERR014: .ASCIZ /WORD COUNT NOT EQUAL 0/<15><12>
11076	061356	052517	052116	047040	
11077	061364	052117	042440	052521	
11078	061372	046101	030040	005015	
11079	061400	000			
11080	061401	102	051525	040440	ERR015: .ASCIZ /BUS ADD INCORRECT/<15><12>
11081	061406	042104	044440	041516	
11082	061414	051117	042522	052103	
11083	061422	005015	000		
11084	061425	103	046131	052054	ERR016: .ASCIZ /CYL,TRK,SEC INCORRECT/<15><12>
11085	061432	045522	051454	041505	
11086	061440	044440	041516	051117	
11087	061446	042522	052103	005015	
11088	061454	000			
11089	061455	103	047117	051124	ERR017: .ASCIZ /CONTROLLER COMMAND TIME OUT/
11090	061462	046117	042514	020122	
11091	061470	047503	046515	047101	
11092	061476	020104	044524	042515	
11093	061504	047440	052125	000	
11094	061511	120	047522	020107	ERR020: .ASCIZ /PROG ERROR/
11095	061516	051105	047522	000122	
11096	061524	046111	020114	052506	ERR021: .ASCIZ /ILL FUNCT CODE/
11097	061532	041516	020124	047503	
11098	061540	042504	000		
11099	061543	106			ERR022: .BYTE 106 .ASCIZ /ERROR WHILE WAITING TO REPORT ERROR/
11100	061544	051105	047522	020122	
11101	061552	044127	046111	020105	
11102	061560	040527	052111	047111	
11103	061566	020107	047524	051040	
11104	061574	050105	051117	020124	
11105	061602	051105	047522	000122	
11106	061610	106			ERR023: .BYTE 106 .ASCIZ /NON-EXIST DRV/
11107	061611	116	047117	042455	
11108	061616	044530	052123	042040	
11109	061624	053122	000		
11110	061627	106			ERR024: .BYTE 106 .ASCIZ /SERCON PAR/
11111	061630	042523	041522	047117	
11112	061636	050040	051101	000	
11113	061643	106			ERR025: .BYTE 106 .ASCIZ /UNIT FIELD ERR/
11114	061644	047125	052111	043040	
11115	061652	042511	042114	042440	
11116	061660	051122	000		
11117	061663	015	044412	052116	ERR026: .ASCIZ <15><12>/INT FROM DRV NOT UNDER TEST/<15><12>
11118	061670	043040	047522	020115	
11119	061676	051104	020126	047516	
11120	061704	020124	047125	042504	
11121	061712	020122	042524	052123	
11122	061720	005015	000		

11123	061723	110	051101	020104	ERR027: .ASCIZ /HARD DRV ERR/<<15><12>
11124	061730	051104	020126	051105	
11125	061736	006522	000012		
11126	061742	051104	020126	052123	ERR029: .ASCIZ /DRV STATUS CHANGE DID NOT CLR/<<15><12>
11127	061750	052101	051525	041440	
11128	061756	040510	043516	02010	
11129	061764	044504	020104	047516	
11130	061772	020124	046103	006522	
11131	062000	000012			
11132	062002	106			ERR029: .BYTE 106
11133	062003	104	044522	042526	.ASCIZ /DRIVE BECAME NOT AVAIL/
11134	062010	041040	041505	046501	
11135	062016	020105	047516	020124	
11136	062024	053101	044501	000114	
11137	062032	041125	054105	042520	ERR030: .ASCIZ /UNEXPECT ATTN/<<15><12>
11138	062040	052103	040440	052124	
11139	062046	006516	000012		
11140	062052	046111	020114	051504	ERR031: .ASCIZ /ILL DSK ADD/
11141	062060	020113	042101	000104	
11142	062066	051127	020124	047514	ERR032: .ASCIZ /WRT LOCK ERR/
11143	062074	045503	042440	051122	
11144	062102	000			
11145	062103	101	020103	047514	ERR033: .ASCIZ /AC LOW/
11146	062110	000127			
11147	062112	050123	042505	020104	ERR034: .ASCIZ /SPEED LOSS/
11148	062120	047514	051523	000	
11149	062125	104	053122	052040	ERR035: .ASCIZ /DRV TYPE ERR/
11150	062132	050131	020105	051105	
11151	062140	000122			
11152	062142	047506	046522	052101	ERR036: .ASCIZ /FORMAT ERR/
11153	062150	042440	051122	000	
11154	062155	116	047117	042455	ERR037: .ASCIZ /NON-EXIST DRV FUNCT/
11155	062162	044530	052123	042040	
11156	062170	053122	043040	047125	
11157	062176	052103	000		
11158	062201	106			ERR038: .BYTE 106
11159	062202	042523	045505	044440	.ASCIZ /SEEK INCOMP/
11160	062210	041516	046517	000120	
11161	062216	106			ERR039: .BYTE 106
11162	062217	104	053122	047440	.ASCIZ /DRV OFF TRK/
11163	062224	043106	052040	045522	
11164	062232	000			
11165	062233	106			ERR040: .BYTE 106
11166	062234	044515	050123	051517	.ASCIZ /MISPOS/
11167	062242	000			
11168	062243	126			ERR042: .BYTE 126
11169	062244	050117	051105	052101	.ASCIZ /OPERAT INCOMP/
11170	062252	044440	041516	046517	
11171	062260	000120			
11172	062262	127			ERR043: .BYTE 127
11173	062263	110	051104	053040	.ASCIZ /HOR VRC ERR/
11174	062270	041522	042440	051122	
11175	062276	000			
11176	062277	116	047117	042455	ERR044: .ASCIZ /NON-EXIST MEM/
11177	062304	044530	052123	046440	
11178	062312	046505	000		

11179	062315	125	044516	052502	ERROR45: .ASCIZ /UNIBUS PAR/
11180	062322	020123	040520	000122	
11181	062330	117			ERROR46: .BYTE 117
11182	062331	104	053122	052040	.ASCIZ /DRV TIMING ERR/
11183	062336	046511	047111	020107	
11184	062344	051105	000122		
11185	062350	040504	040524	046040	ERROR47: .ASCIZ /DATA LATE/<<15><12>
11186	062356	052101	006505	000012	
11187	062364	117			ERROR48: .BYTE 117
11188	062365	104	052101	020101	.ASCIZ /DATA CHK/
11189	062372	044103	000113		
11190	062376	107			ERROR50: .BYTE 107
11191	062377	127	044522	042524	.ASCIZ /WRITE CHK ERR/
11192	062404	041440	045510	042440	
11193	062412	051122	000		
11194	062415	103	046131	040440	ERROR51: .ASCIZ /CYL ADDR OVRFLW/
11195	062422	042104	020122	053117	
11196	062430	043122	053514	000	
11197	062435	106			ERROR52: .BYTE 106
11198	062436	046503	042116	052040	.ASCIZ /CMND TIME OUT DURING POSITIONING OPERATION/<<15><12>
11199	062444	046511	020105	052517	
11200	062452	020124	052504	044522	
11201	062460	043516	050040	051517	
11202	062466	052111	047511	044516	
11203	062474	043516	047440	042520	
11204	062502	040522	044524	047117	
11205	062510	005015	000		
11206	062513	106			ERROR53: .BYTE 106
11207	062514	051104	020126	047516	.ASCIZ /DRV NOT RDY AFTER START SPINDLE/
11208	062522	020124	042122	020131	
11209	062530	043101	042524	020122	
11210	062536	052123	051101	020124	
11211	062544	050123	047111	046104	
11212	062552	000105			
11213	062554	106			ERROR54: .BYTE 106
11214	062555	126	046117	053040	.ASCIZ /VOL VALID DID NOT SET AFTER PACK ACK/
11215	062562	046101	042111	042040	
11216	062570	042111	047040	052117	
11217	062576	051440	052105	040440	
11218	062604	052106	051105	050040	
11219	062612	041501	020113	041501	
11220	062620	000113			
11221	062622	051104	020126	047125	ERROR56: .ASCIZ /DRV UNSAFE/
11222	062630	040523	042506	000	
11223	062635	104	053122	042440	ERROR57: .ASCIZ /DRV ERR NOT INDICATED BY FAULT/<<15><12>
11224	062642	051122	047040	052117	
11225	062650	044440	042116	041511	
11226	062656	052101	042105	041040	
11227	062664	020131	040506	046125	
11228	062672	006524	000012		
11229	062676	051104	020126	052101	ERROR58: .ASCIZ /DRV ATTN BUT NO FAULT OR DRV STATUS CHANGE/<<15><12>
11230	062704	047124	041040	052125	
11231	062712	047040	020117	040506	
11232	062720	046125	020124	051117	
11233	062726	042040	053122	051440	
11234	062734	040524	052524	020123	

11235	062742	044103	047101	042507	
11236	062750	005015	000		
11237	062753	103	047115	020104	ERR059: .ASCIZ /CMND TIME OUT DRV SEIZED BY OTHER PORT/<15><12>
11238	062760	044524	042515	047447	
11239	062766	052125	042040	053122	
11240	062774	051440	044505	042532	
11241	063002	020104	054502	047440	
11242	063010	044124	051105	050040	
11243	063016	051117	006524	000012	
11244	063024	041505	020103	044515	ERR060: .ASCIZ /ECC MISCORRECTION/<15><12>
11245	063032	041523	051117	042522	
11246	063040	052103	047511	006516	
11247	063046	000012			
11248	063050	040504	040524	04' 40	ERR062: .ASCIZ /DATA COMPARE ERROR/<15><12>
11249	063056	046517	040520	042522	
11250	063064	042440	051122	05:117	
11251	063072	005015	000		
11252	063075	126			ERR064: .BYTE 126
11253	063076	044515	050123	051517	.ASCIZ /MISPOS/
11254	063104	000			
11255	063105	106			ERR065: .BYTE 106
11256	063106	042510	042101	051440	.ASCIZ /HEAD SELECT ERROR/
11257	063114	046105	041505	020124	
11258	063122	051105	047522	000122	
11259	063130	041505	020103	047514	ERR066: .ASCIZ /ECC LOGIC ERROR/
11260	063136	044507	020103	051105	
11261	063144	047522	000122		
11262	063150	045522	041505	052120	ERR067: .ASCIZ /RKECPT RKECPS/<15><12>
11263	063156	020040	045522	041505	
11264	063164	051520	005015	000	
11265	063171	127	051117	020104	ERR070: .ASCIZ /WORD COUNT INVALID/
11266	063176	047503	047125	020124	
11267	063204	047111	040525	044514	
11268	063212	000104			
11269	063214	052502	020123	042101	ERR071: .ASCIZ /BUS ADDRESS INVALID/
11270	063222	051104	051505	020123	
11271	063230	047111	040526	044514	
11272	063236	000104			
11273	063240	054503	044514	042116	ERR072: .ASCIZ /CYLINDER INVALID/
11274	063246	051105	044440	053116	
11275	063254	046101	042111	000	
11276	063261	124	040522	045503	ERR073: .ASCIZ "TRACK/SECTOR INVALID"
11277	063266	051457	041505	047524	
11278	063274	020122	047111	040526	
11279	063302	044514	000104		
11280	063306	106			ERR074: .BYTE 106
11281	063307	105	051122	051117	.ASCIZ /ERROR IN DRV DID NOT RELOAD HEADS/
11282	063314	044440	020116	051104	
11283	063322	020126	044504	020104	
11284	063330	047516	020124	042522	
11285	063336	047514	042101	044040	
11286	063344	040505	051504	000	
11287	063351	103	051125	042522	ERR100: .ASCII /CURRENT SUPPLIED/<15><12>
11288	063356	052116	051440	050125	
11289	063364	046120	042511	006504	
11290	063372	012			

11347	064054	052040	051110	051505	
11348	064062	047510	042114	042440	
11349	064070	041530	042505	042504	
11350	064076	006504	000012		
11351	064102	052521	051505	044524	ERR206: .ASCIZ /QUESTIONABLE DRIVE STATUS/<15><12>
11352	064110	047117	041101	042514	
11353	064116	042040	044522	042526	
11354	064124	051440	040524	052524	
11355	064132	006523	000012		
11356	064136	047503	052116	047522	ERR207: .ASCIZ /CONTROLLER TIMED OUT WITH GO SET/<15><12>
11357	064144	046114	051105	052040	
11358	064152	046511	042105	047440	
11359	064160	052125	053440	052111	
11360	064166	020110	047507	051440	
11361	064174	052105	005015	000	
11362	064201	105	051122	051117	ERR208: .ASCIZ /ERROR WHILE DIAGNOSING/<15><12>
11363	064206	053440	044510	042514	
11364	064214	042040	040511	047107	
11365	064222	051517	047111	006507	
11366	064230	000012			
11367	064232	127			ERR209: .BYTE 127
11368	064233	110	040505	042504	.ASCIZ /HEADER ERROR/<15><12>
11369	064240	020122	051105	047522	
11370	064246	006522	000012		
11371	064252	025052	043052	051111	ERR210: .ASCIZ /***FIRST ERROR INFO***/<15><12>
11372	064260	052123	042440	051122	
11373	064266	051117	044440	043116	
11374	064274	025117	025052	005015	
11375	064302	000			
11376	064303	122	050113	051517	ERR211: .ASCIZ /RKPOS /
11377	064310	000040			
11378	064312	045522	040520	020124	ERR212: .ASCIZ /RKPAT /
11379	064320	000			
11380	064321	101	042104	042522	ERR213: .ASCIZ /ADDRESS OF BEGINNING OF SECTOR IN ERROR /
11381	064326	051523	047440	020106	
11382	064334	042502	044507	047116	
11383	064342	047111	020107	043117	
11384	064350	051440	041505	047524	
11385	064356	020122	047111	042440	
11386	064364	051122	051117	000040	
11387	064372	005015	040504	040524	ERR214: .ASCIZ <15><12>/DATA READ<<15><12>
11388	064400	051040	040505	006504	
11389	064406	000012			
11390	064410	045522	051503	020061	ERR300: .ASCIZ /RKCS1 RKCS2 RKWCR RKBA RKDA RKDC RKASOF RKER/<15><12>
11391	064416	020040	045522	051503	
11392	064424	020062	020040	045522	
11393	064432	041527	020122	020040	
11394	064440	045522	040502	020040	
11395	064446	020040	045522	040504	
11396	064454	020040	020040	045522	
11397	064462	041504	020040	020040	
11398	064470	045522	051501	043117	
11399	064476	020040	045522	051105	
11400	064504	005015	000		
11401	064507	122	042113	020123	ERR301: .ASCIZ /RKDS RKMR1 RKMR2 RKMR3 RKPOS RKPAT/<15><12>
11402	064514	020040	051040	046513	

11403 064522 030522 020040 051040
 11404 064530 046513 031122 020040
 11405 064536 051040 045513 031522
 11406 064544 020040 051040 050113
 11407 064552 051517 020040 051040
 11408 064560 050113 052101 005015
 11409 064566 000
 11410 064567 122 042113 006523
 11411 064574 000012
 11412 064576 005015 042515 051523
 11413 064604 020040 020040 020040
 11414 064612 040440 020040 020040
 11415 064620 020040 041040 005015
 11416 064626 030040 020060 020040
 11417 064634 020040 000
 11418 064637 015 020012 030460
 11419 064644 020040 020040 000040
 11420 064652 005015 030440 020060
 11421 064660 020040 020040 000
 11422 064665 015 020012 030461
 11423 064672 020040 020040 000040
 11424 064700 047125 054105 042520
 11425 064706 052103 042105 046440
 11426 064714 046505 051117 020131
 11427 064722 040520 044522 054524
 11428 064730 052040 040522 000120
 11429 064736 044524 042515 072
 11430 064743 040 020040 020072
 11431 064750 035040 020040 005015
 11432 064756 000
 11433 064760

ERR302: .ASCIZ /RKDS/<15><12>
 ERR303: .ASCIZ <15><12>/MESS A B/<15><12>/ 00 /
 ERR304: .ASCIZ <15><12>/ 01 /
 ERR305: .ASCIZ <15><12>/ 10 /
 ERR306: .ASCIZ <15><12>/ 11 /
 ERR400: .ASCIZ /UNEXPECTED MEMORY PARITY TRAP/
 TIM000: .ASCII /TIME: /
 TIM001: .ASCIZ / : : /<15><12>
 .EVEN

```

11434          .SBTTL  *** DUMP MEMORY ***
11435
11436          .EVEN
11437 064760 000040          .BLKW 40          ;RESERVE SPACE FOR STACK
11438 065060 000000          ..STCK: .WORD 0          ;STORED STACK POINTER
11439
11440 065062 012737 000340 177776 ..DUMP: MOV #PR7,PS          ;LOCK OUT ALL INTERRUPTS
11441 065070 010637 065060 MOV SP,..STCK          ;STORE STACK POINTER
11442 065074 012706 065060 MOV #..STCK,SP          ;LOAD STACK POINTER
11443 065100 010546 MOV R5,-(SP)          ;PUSH R5 ON STACK
11444 065102 010446 MOV R4,-(SP)          ;PUSH R4 ON STACK
11445 065104 010346 MOV R3,-(SP)          ;PUSH R3 ON STACK
11446 065106 010246 MOV R2,-(SP)          ;PUSH R2 ON STACK
11447 065110 010146 MOV R1,-(SP)          ;PUSH R1 ON STACK
11448 065112 010046 MOV R0,-(SP)          ;PUSH R0 ON STACK
11449 065114 013705 000224 MOV ..LOW,R5          ;STORE FIRST ADDRESS
11450 065120 004737 065452 JSR PC,..CR          ;TY <CR><LF>
11451 065124 004037 065532 JSR R0,..ASCII          ;TYPE R0 =
11452 065130 065640          ..R0
11453 065132 011646 MOV (SP),-(SP)
11454 065134 004737 065556 JSR PC,..OCT
11455 065140 004737 065452 JSR PC,..CR
11456 065144 004037 065532 JSR R0,..ASCII          ;TYPE R1 =
11457 065150 065653          ..R1
11458 065152 016646 000002 MOV 2(SP),-(SP)
11459 065156 004737 065556 JSR PC,..OCT
11460 065162 004737 065452 JSR PC,..CR
11461 065166 004037 065532 JSR R0,..ASCII          ;TYPE R2 =
11462 065172 065666          ..R2
11463 065174 016646 000004 MOV 4(SP),-(SP)
11464 065200 004737 065556 JSR PC,..OCT
11465 065204 004737 065452 JSR PC,..CR
11466 065210 004037 065532 JSR R0,..ASCII          ;TYPE R3 =
11467 065214 065701          ..R3
11468 065216 016646 000006 MOV 6(SP),-(SP)
11469 065222 004737 065556 JSR PC,..OCT
11470 065226 004737 065452 JSR PC,..CR
11471 065232 004037 065532 JSR R0,..ASCII          ;TYPE R4 =
11472 065236 065714          ..R4
11473 065240 016646 000010 MOV 10(SP),-(SP)
11474 065244 004737 065556 JSR PC,..OCT
11475 065250 004737 065452 JSR PC,..CR
11476 065254 004037 065532 JSR R0,..ASCII          ;TYPE R5 =
11477 065260 065727          ..R5
11478 065262 016646 000012 MOV 12(SP),-(SP)
11479 065266 004737 065556 JSR PC,..OCT
11480 065272 004737 065452 JSR PC,..CR
11481 065276 004037 065532 JSR R0,..ASCII          ;TYPE SP =
11482 065302 065742          ..SP
11483 065304 013746 065060 MOV ..STCK,-(SP)
11484 065310 004737 065556 JSR PC,..OCT
11485 065314 004737 065452 JSR PC,..CR
11486 065320 004737 065452 JSR PC,..CR
11487 065324 004037 065532 JSR R0,..ASCII          ;TYPE HEADER
11488 065330 065755          ..ADD
11489 065332 004737 065452 JSR PC,..CR

```

```

11490 065336 004737 065452          JSR      PC,..CR
11491
11492 065342 010546          10$:    MOV      R5,-(SP)          ;TYPE ADDRESS
11493 065344 004737 065556          JSR      PC,..OCT
11494 065350 012703 000004          MOV      #4,R3                ;TYPE 4 LOCATIONS PER LINE
11495 065354 004037 065532          12$:    JSR      RD,..ASCII
11496 065360 066000          ..BLNK
11497 065362 012546          MOV      (R5)+,-(SP)          ;TYPE CONTENTS
11498 065364 004737 065556          JSR      PC,..OCT
11499 065370 005303          DEC      R3                    ;CHECK IF START OF NEXT LINE
11500 065372 001370          BNE      12$
11501 065374 004737 065452          JSR      PC,..CR
11502 065400 005777 113534          TST      @SWR
11503 065404 100001          BPL      15$                   ;CHECK IF STOP PRINT OUT
11504 065406 000000          HALT
11505 065410 023705 000226          15$:    CMP      ..HIGH,R5          ;CHECK IF FINISHED
11506 065414 103352          BHIS    10$
11507 065416 004737 065452          JSR      PC,..CR
11508 065422 004737 065452          JSR      PC,..CR
11509 065426 012600          MOV      (SP)+,R0              ;: POP STACK INTO R0
11510 065430 012601          MOV      (SP)+,R1              ;: POP STACK INTO R1
11511 065432 012602          MOV      (SP)+,R2              ;: POP STACK INTO R2
11512 065434 012603          MOV      (SP)+,R3              ;: POP STACK INTO R3
11513 065436 012604          MOV      (SP)+,R4              ;: POP STACK INTO R4
11514 065440 012605          MOV      (SP)+,R5              ;: POP STACK INTO R5
11515 065442 013706 065060          MOV      ..STCK,SP            ;RESTORE STACK POINTER
11516 065446 000000          HALT
11517 065450 000776          BR      .-2
11518
11519 065452 105777 113472          ..CR:   TSTB    @STPS          ;WAIT FOR READY
11520 065456 100375          BPL      .CR
11521 065460 112777 000015 113464          MOVB    #15,@STPB            ;LOAD <CR>
11522 065466 105777 113456          5$:     TSTB    @STPS          ;WAIT FOR READY
11523 065472 100375          BPL      5$
11524 065474 112777 000012 113450          MOVB    #12,@STPB            ;LOAD <LF>
11525 065502 113701 001155          MOVB    $FILLS,R1            ;LOAD FILL COUNT
11526 065506 001410          BEQ     20$                   ;RETURN IF NO FILLS
11527 065510 105777 113434          10$:    TSTB    @STPS          ;WAIT FOR READY
11528 065514 100375          BPL      10$
11529 065516 113777 001154 113426          MOVB    $NULL,@STPB          ;LOAD FILL CHARACTER
11530 065524 005301          DEC     R1                    ;CHECK IF FILL COMPLETE
11531 065526 001370          BNE     10$
11532 065530 000207          20$:    RTS      PC              ;RETURN
11533
11534 065532 012001          ..ASCII: MOV     (R0)+,R1        ;STORE ADDRESS OF ASCII
11535
11536 065534 105711          1$:     TSTB    (R1)              ;CHECK IF FINISHED
11537 065536 001406          BEQ     20$
11538 065540 105777 113404          2$:     TSTB    @STPS          ;WAIT FOR READY
11539 065544 100375          BPL      2$
11540 065546 112177 113400          MOVB    (R1)+,@STPB          ;TYPE CHARACTER
11541 065552 000770          BR      1$
11542
11543 065554 000200          20$:    RTS      RD              ;RETURN
11544
11545 065556 012701 000006          ..OCT:  MOV     #6,R1          ;LOAD DIGIT COUNT INTO R1

```

```

11546 065562 005002          CLR      R2          ;CLEAR R2 FOR OUTPUT
11547 065564 000407          BR       10$        ;GET LEADING DIGIT
11548
11549 065566 005002          5$:     CLR      R2          ;GET OCTAL DIGIT
11550 065570 006366 000002          ASL     R2(SP)
11551 065574 006102          ROL     R2
11552 065576 006366 000002          ASL     2(SF)
11553 065602 006102          ROL     R2
11554 065604 006366 000002          10$:    ASL     2(SP)
11555 065610 006102          ROL     R2
11556 065612 052702 000060          BIS     #60,R2      ;MAKE IT ASCII
11557 065616 105777 113326          15$:    TSTB   @STPS    ;WAIT FOR READY
11558 065622 100375          BPL     15$
11559 065624 110277 113322          MOVB   R2,@STPB    ;TYPE OCTAL DIGIT
11560 065630 005301          DEC     R1          ;CHECK IF WHOLE WORD PRINTED
11561 065632 001355          BNE     5$
11562 065634 012616          MOV     (SP)+,(SP) ;ADJUST RETURN
11563 065636 000207          RTS     PC          ;RETURN
11564

```

```

11565 065640 030122 036440 02004C ..R0: .ASCIZ /R0 = /
11566 065646 020040 020040 000          ..R1: .ASCIZ /R1 = /
11567 065653 122 020061 020075
11568 065660 020040 020040 000040 ..R2: .ASCIZ /R2 = /
11569 065666 031122 036440 020040
11570 065674 020040 020040 000          ..R3: .ASCIZ /R3 = /
11571 065701 122 020063 020075
11572 065706 020040 020040 000040 ..R4: .ASCIZ /R4 = /
11573 065714 032122 036440 020040
11574 065722 020040 020040 000          ..R5: .ASCIZ /R5 = /
11575 065727 122 020065 020075
11576 065734 020040 020040 000040 ..SP: .ASCIZ /SP = /
11577 065742 050123 036440 020040
11578 065750 020040 020040 000          ..ADD: .ASCIZ /ADDRESS CONTENTS/
11579 065755 101 042104 042522
11580 065762 051523 020040 041440
11581 065770 047117 042524 052116

```

```

11582 065776 000123
11583 066000 020040 020040 U00 ..BLNK: .ASCIZ / /
11584 066006 .EVEN
11585 066006 H0BUFF=.
11586 066212 BUFADD=+132.
11587 066006 044124 020105 045522 HELP: .ASCII /THE RK06-RK07-K MUST BE PROPERLY FORMATED 22 SECTORS PER TRACK/<15><12>
11588 066014 033060 051055 030113
11589 066022 026467 020113 052515
11590 066030 052123 041040 020105
11591 066036 051120 050117 051105
11592 066044 054514 043040 051117
11593 066052 040515 042524 020104
11594 066060 031062 051440 041505
11595 066066 047524 051522 050040
11596 066074 051105 052040 040522
11597 066102 045503 050115
11598 066106 047524 041040 043505 .ASCII /TO BEGIN PERFORMANCE TESTING TYPE CONTROL-C < C>/<15><12>
11599 066114 047111 050040 051105
11600 066122 047506 046522 047101
11601 066130 042503 052040 051505

```

11602	066136	044524	043516	052040	
11603	066144	050131	020105	047503	
11604	066152	052116	047522	026514	
11605	066160	020103	057074	037103	
11606	066166	005015			
11607	066170	042514	040507	020114	.ASCII /LEGAL COMMANDS ARE:/(15)(12)(12)
11608	066176	047503	046515	047101	
11609	066204	051504	040440	042522	
11610	066212	006472	005012		
11611	066216	047124	026440	020040	.ASCII /TN - TEST DRIVE N/(15)(12)
11612	066224	042524	052123	042040	
11613	066232	044522	042526	047040	
11614	066240	005015			
11615	066242	047127	026440	020040	.ASCII /WN - WRITE PACK, VERIFY PACK, AND TEST DRIVE N/(15)(12)
11616	066250	051127	052111	020105	
11617	066256	040520	045503	053054	
11618	066264	051105	054506	050040	
11619	066272	041501	026113	040440	
11620	066300	042116	052140	051505	
11621	066306	020124	051104	053111	
11622	066314	020105	006516	012	
11623	066321	120	020116	020055	.ASCII /PN - CHANGE PARAMETER & TEST DRIVE N/(15)(12)
11624	066326	041440	040510	043516	
11625	066334	020105	040520	040522	
11626	066342	042515	042524	020122	
11627	066350	020046	042124	052123	
11628	066356	042040	044522	042526	
11629	066364	047040	005015		
11630	066370	047123	026440	020040	.ASCII /SN - GET STATISTICS ON DRIVE N/(15)(12)
11631	066376	042507	020124	052123	
11632	066404	052101	051511	044524	
11633	066412	051503	047440	020116	
11634	066420	051104	053111	020105	
11635	066426	006516	012		
11636	066431	104	020116	020055	.ASCII /DN - DROP DRIVE N AND GET STATISTICS/(15)(12)(12)
11637	066436	042040	047522	020120	
11638	066444	051104	053111	020105	
11639	066452	020116	047101	020104	
11640	066460	042507	020124	052123	
11641	066466	052101	051511	044524	
11642	066474	051503	005015	012	
11643	066501	116	041440	047101	.ASCII /N CAN BE 0-7 OR A/(15)(12)(12)
11644	066506	041040	020105	026460	
11645	066514	020067	051117	040440	
11646	066522	005015	012		
11647	066525	104	052101	020101	.ASCII /DATA PATTERNS MUST BE WRITTEN BY THE EXERCISER/(15)(12)
11648	066532	040520	052124	051105	
11649	066540	051516	046440	051525	
11650	066546	020124	042502	053440	
11651	066554	044522	052124	047105	
11652	066562	041040	020131	044124	
11653	066570	020105	054105	051105	
11654	066576	044503	042523	006522	
11655	066604	012			
11656	066605	102	043105	051117	.ASCII /BEFORE PERFORMANCE TESTING/(15)(12)(12)
11657	066612	020105	042520	043122	

11658	066620	051117	040515	041516
11659	066626	020105	042524	052123
11660	066634	047111	006507	005012
11661	066642	025052	053452	051101
11662	066650	044516	043516	025052
11663	066656	020052	043111	050040
11664	066664	047522	042503	051523
11665	066672	051117	044040	046101
11666	066700	020124	052504	044522
11667	066706	043516	042040	052101
11668	066714	020101	051124	047101
11669	066722	043123	051105	005015
11670	066730	020040	020040	052040
11671	066736	020117	051104	053111
11672	066744	026105	041040	042101
11673	066752	042440	041503	046440
11674	066760	054501	041040	020105
11675	066766	051127	052111	042524
11676	066774	020116	047117	050040
11677	067002	041501	027113	005015
11678	067010	020040	020040	042040
11679	067016	044522	042526	020123
11680	067024	052515	052123	041040
11681	067032	020105	051104	050117
11682	067040	042520	020104	054502
11683	067046	050040	047522	051107
11684	067054	046501	047440	020122
11685	067062	044527	044124	042040
11686	067070	047522	020120	047503
11687	067076	046515	047101	006504
11688	067104	005012	000	
11689		000001		

.ASCII /***WARNING*** IF PROCESSOR HALT DURING DATA TRANSFER/<15><12>

.ASCII / TO DRIVE, BAD ECC MAY BE WRITTEN ON PACK./<15><12>

.ASCIZ / DRIVES MUST BE DROPPED BY PROGRAM OR WITH DROP COMMAND/<15><12><12

.END

ATESTN=	000000	1743	1747																	
AUNIT =	000000	1743	1750																	
AUSWR =	000000	1743	1757																	
AVAILQ	001314	1810#	3419	3814	4361	4528	5992	6955	7708	8277	8414									
AVECT1=	120210	1173#	1743	1782																
AVECT2=	000000	1743	1783																	
A.ABNL	001456	1872#	8676	8847	8864	9058	9078	9555												
A.CONT	001460	1873#	8743	8826	8835	8906	8928	8954	8999	9046	9070	9102	9151	9203						
		9230	9368	9566	9612	9621	9634													
A.NORM	001454	1871#	8796	8893	9014	9562	9642													
BAI =	000020	1260#	9516																	
BA16 =	000400	1236#	6075	7143																
BA17 =	001000	1237#	6075	7143																
BOSECT	042356	7331	8345#																	
BINOCT	052774	3569	6233	6238	6241	6244	6247	6250	6253	6256	6259	6280	6555	6560						
		6648	6651	6655	6659	6662	6668	6682	6685	6689	6693	6699	6702	6705						
		6711	6832	6835	6841	6844	6847	7425	7429	7603	7629	7632	7636	7645						
		7672	7855	7858	9948	10044#	10225													
BIT0 =	000001	1157#	1233	1250	1276	1298	1479	1880	4092	4167	4176	4182	5927	6394						
		6495	6513	6634	6735	6784	6794	6826	6852	6922	7085	7506	7654	7724						
		9619																		
BIT00 =	000001	1147#	1157																	
BIT01 =	000002	1146#	1156																	
BIT02 =	000004	1145#	1155																	
BIT03 =	000010	1144#	1154																	
BIT04 =	000020	1143#	1153																	
BIT05 =	000040	1142#	1152																	
BIT06 =	000100	1141#	1151																	
BIT07 =	000200	1140#	1150																	
BIT08 =	000400	1139#	1149																	
BIT09 =	001000	1138#	1148																	
BIT1 =	000002	1156#	1251	1278	1480	1881	4199	4204	4206	6072	6498	6744	6786	6792						
		6814	7085	7502	7504	7724														
BIT10 =	002000	1137#	1238	1267	1289	1324	1338	1351	1366	1380	1492	1890	4209	7083						
		7894	7948																	
BIT11 =	004000	1136#	1239	1268	1290	1309	1325	1339	1352	1367	1381	1493	4217	7089						
BIT12 =	010000	7123	7124	7300	7894	7953	8018	8020	8082	8171										
BIT13 =	020000	1135#	1241	1269	1291	1326	1340	1353	1354	1368	1382	1494	7088	7123						
		8021	8027	8029																
BIT14 =	040000	1134#	1242	1270	1292	1310	1327	1341	1355	1369	1383	1495	7088	7123						
		8037	8043	8045																
BIT15 =	100000	1133#	1243	1271	1293	1311	1328	1342	1356	1370	1384	1496	7991	8216						
		8260	8345	8399																
BIT2 =	000004	1132#	1244	1245	1272	1294	1312	1329	1389	1497	1892	6165	6524	6854						
		6874	6892	7020	7352	7516	7871	8369	8391	8419										
BIT3 =	000010	1155#	1252	1279	1280	1300	1481	1882	4193	4212	4214	6084	6163	6501						
		6753	6797	6802	6823	7085	7493	7724	8245	8438										
BIT4 =	000020	1154#	1258	1259	1281	1301	1482	1883	4230	4264	4278	4311	4327	5946						
		6124	6504	6537	6539	6849	6934	7724	7894	7959	8348									
BIT5 =	000040	1153#	1253	1260	1282	1302	1303	1318	1360	1374	1483	1884	3477	4053						
		4069	4341	4404	6203	6837	6997	7320	8088	8092	8442	8446	8455							
BIT6 =	000100	1152#	1261	1262	1283	1304	1319	1333	1346	1361	1375	1484	1885	4417						
		4472	4486	5958	6211	6746	6816	7018	8458	9445										
BIT7 =	000200	1151#	1234	1263	1284	1305	1320	1334	1347	1362	1376	1485	1887	4224						
		4227	4229	6159	6750	6820	7021	7030	7724	8067	8070	8442								
		1150#	1235	1264	1285	1306	1307	1321	1335	1348	1363	1377	1487	4315						

ERR032	062066	7944	11142#			
ERR033	062103	7912	11145#			
ERR034	062112	7915	11147#			
ERR035	062125	7926	11149#			
ERR036	062142	7931	11152#			
ERR037	062155	7936	11154#			
ERR038	062201	7112	11158#			
ERR039	062216	7120	11161#			
ERR040	062233	7297	11165#			
ERR042	062243	8187	11168#			
ERR043	062262	8131	11172#			
ERR044	062277	7098	11176#			
ERR045	062315	7104	11179#			
ERR046	062330	7222	11181#			
ERR047	062350	7339	11185#			
ERR048	062364	7390	11187#			
ERR050	062376	7469	11190#			
ERR051	062415	7131	11194#			
ERR052	062435	8075	8079	8448	8460	11197#
ERR053	062513	7950	11206#			
ERR054	062554	7956	11213#			
ERR056	062622	7921	11221#			
ERR057	062635	7961	11223#			
ERR058	062676	7971	8472	11229#		
ERR059	062753	7000	8457	11237#		
ERR060	063024	7422	11244#			
ERR062	063050	7510	11248#			
ERR064	063075	8168	11252#			
ERR065	063105	7312	11255#			
ERR066	063130	7404	7851	11259#		
ERR067	063150	6830	7853	11262#		
ERR070	063171	7140	8366	11265#		
ERR071	063214	7155	8358	11269#		
ERR072	063240	7163	11273#			
ERR073	063261	7170	11276#			
ERR074	063306	7015	11280#			
ERR100	063351	6673	11287#			
ERR101	063462	6641	11301#			
ERR102	063544	6644	6678	11310#		
ERR108	063552	7669	11311#			
ERR111	063563	6839	11313#			
ERR200	063611	6384	11317#			
ERR201	063651	6392	6396	6439	11323#	
ERR202	063721	6742	6809	7513	7985	11330#
ERR203	063757	7691	11336#			
ERR204	064012	6939	6949	8432	11341#	
ERR205	064047	6899	6929	7683	11346#	
ERR206	064102	6748	6818	11351#		
ERR207	064136	6406	11356#			
ERR208	064201	8445	11362#			
ERR209	064232	8193	8452	11367#		
ERR210	064252	8478	11371#			
ERR211	064303	7423	11376#			
ERR212	064312	7427	11378#			
ERR213	064321	6550	11380#			
ERR214	064372	6556	11387#			

P.A10 = 000050	1517*	9349*												
P.A11 = 000054	1519*	4183	9356*											
P.BAHI= 000007	1472*													
P.BALO= 000010	1473*	4281	4435*	5454*	5462*	5676	6080	6701	7145	7151	7180	7254*	7272	
	7323*	7365	7454*	7484*	7538*	7549	7552	7735*	8155*	8266*	8335	8383*	9496	
	9586													
P.BAR = 000024	1507*	6081	7145	7152	7478	8355	9032*	9302*						
P.BFCP= 000222	1648*	5677*	5678*	5718	7412*									
P.BUFF= 000145	1568*	3456*	5457*	5466*	5506	5508*								
P.B00 = 000042	1514*	8980*	8981	9364*	9537*									
P.B01 = 000046	1516*	9343*												
P.B10 = 000052	1518*	7288	9350*											
P.B11 = 000056	1520*	7308	8094	8892*	9357*									
P.CERT= 000100	1537*	4718*	4730*	4944*	7678									
P.CMLG= 000220	1647*	5674*	5719	7413*										
P.CMND= 000001	1466*	4169*	4191*	4197*	4222*	4234*	4259*	4308*	4373*	4410*	4421	4437	4439*	
	4442*	5209*	5218*	5220	5290*	5615	5956	5958	5984	5987	5995*	6006	6012	
	6062*	6681	6970*	7022	7025*	7090*	7125*	7251*	7257	7301*	7324*	7451*	7457	
	7481*	7535*	7703	7705	7711*	7731*	7804*	8022*	8038*	8059*	8060	8071*	8083*	
	8152*	8158	8172*	8211*	8225	8227	8236	8248	8297*	8351	8392	8783	9445	
	9449	9461	9471	9477	9483	9486	9492	9500	9507	9523	9569	9583	9603	
	9609	9875	9877	9879										
P.CS1 = 000016	1504*	5899	6075	6269	7037	7073	7081	7143	8804*	8860*	8898*	8899*	8901*	
	8902	9027*	9029*	9296*	9299*	9435	9489*	9490*	9492*	9504*	9507*	9509*	9510*	
	9518	9526*	9527*	9529*	9530	9532*	9533	9538	9576*	9577*	9579*	9580	9596*	
	9597*	9599*	9600	9616*	9617	9626*	9627	9917	9927*	9928*	9930*	9931	9576	
P.CS1H= 000007	1471*	5008*	5011*	6456	6458*	6462*	6466	8898	9191	9489	9509	9526	9576	
	9587	9596	9927											
P.CS2 = 000020	1505*	5911	7043	7061	7067	7095	7101	7334	7464	8810	9030*	9300*	9443*	
	9468	9516*	9517	9525	9542	9546	9573*	9574	9595					
P.CYLN= 000002	1467*	4238*	4260*	4306*	4376*	4433*	4444	4446	5210*	5219*	5360*	6120	6684	
	7158	7187	7192	7253*	7280	7322*	7364	7453*	7483*	7537*	7562	7734*	7815	
	8153*	8265*	8295*	8313	8381*	8896	9473	9498	9593	9925				
P.DCYL= 000030	1509*	4433	4434	6121	7158	7187	7196	7271	7280	7371	8165	8284*	8302	
	9034*	9308*												
P.JPAT= 000121	1545*	4263*	4380*	4484*	5217*	5262*	5622	6710						
P.DRVN= 000000	1465*	4088	4102	4159	4353	4508	4535	6185	6423	6631	6951	6963	6985	
	7244	7446	7530	7698	8147	8780	9189	9417	9881					
P.DS = 000036	1512*	4188	4207	4215	4345	4366	5944	6232	6270	7117	7878	7910	7993	
	8072	8813	9037*	9304*	9548									
P.DSTT= 000150	1588*	3458*	5200*	5966	6153	6165*	6537	6539*	6740	6786	6792*	6794*	6797	
	6802*	6807	6854*	6872	6874*	6882	6889	6892*	6922	6968*	7020*	7021*	7083	
	7088*	7089*	7123*	7124*	7300*	7320*	7350	7352*	7502	7504*	7506*	7511	7516*	
	7654	7694*	7869	7871*	7983	8018	8020*	8021*	8027	8029*	8037*	8043	8045*	
	8067	8070*	8082*	8088	8092*	8171*	8216	8260*	8345	8369*	8391*	8399*	8419*	
	8442	8446												
P.DTS = 000026	1508*	4431	4432	6105	6118	7166	7185	7195	7270	7277	7370	8283*	8301	
	9033*	9303*												
P.ECMP= 000117	1543*	4165*	5789*	7414*										
P.EPAT= 000062	1522*	6310	6831	7401	7428	7755	7769	7854	8806*	9437	9919			
P.EPOS= 000060	1521*	6299	6834	7397	7399	7424	7857	8807*						
P.ER = 000034	1511*	5925	5937	6420*	6828	7049	7075	7109	7128	7173	7264	7329	7358	
	7395	7887	7919	7924	7929	7934	7939	7995	9036*	9305*				
P.ERAD= 000217	1646*	5781*	7497	7598	7604	7614	7635	7637						
P.ERHD= 000214	1645*	5676*	5717	5723*	5827	5832	7410*	7499	7596	7609	7643			
P.ERR = 000212	1633*	3457*	3492*	4087*	4209*	4217*	5201*	5946*	5951*	6150*	6159*	6163*	6203*	

T.ASOF	001424	10249	10266	10441	10614	10726*	8867*	8868	8950*	8951	8958	8994	9035	9043
T.BA	001416	1855*	6442	8793*	8794	8822								
T.COMT=	000012	9067	9099	9258*										
T.CS1	001410	1852*	9032	9254*										
		1428*	7862											
		1847*	6188*	6189	6217	6430	6527*	8763*	8790*	8791	8804	8819*	8852*	8854
		8860	8889*	8911*	8913	8945	8977*	8991*	9028*	9029	9040*	9064*	9083*	9096*
		9146*	9147	9187	9191*	9192*	9194	9198	9211	9251*	9298*	9299	9339*	9346*
		9353*	9360*	9843*	9844	9849	9855*							
T.CS2	001412	1849*	6434	8739*	8740	8755	8758	8773	8880*	8881	8917	8931	9030	9188*
		9189*	9190	9200	9209	9213	9252*	9858	9862					
		1853*	9033	9255*										
T.DA	001420	1864*	6221											
T.DB	001444	1864*	6221											
T.DC	001422	1854*	9034	9259*										
T.DLT =	001000	1427*	7343											
T.DS	001430	1858*	6218	9037	9256*	9860								
T.ER	001426	1857*	6390	6436	8920*	8923	9036	9215	9257*					
T.MR1	001432	1859*	9260*											
T.MR2	001434	1860*	8829	8840	8985	9002	9051	9073	9090	9105	9261*	9342	9349	9356
		9363												
		1861*	8832	8892	8980	9086	9262*	9343	9350	9357	9364	9365		
T.MR3	001436	1414*	6876											
T.NER =	000020	1863*	9264*											
T.PAT	001442	1862*	9263*											
T.POS	001440	1851*	9031	9253*										
T.WCR	001414	1482*	7903	7976	8461	8799								
UEXATT=	000010	1265*	5911	6434	7067	8755	8810	9213	9546					
UFE	= 000400	1205*	9449											
UNLOAD=	000107	6915	6920*	7233	7439	7475	7524	8006	8141					
UNRECY	031502	1293*	5937	7887	7919	7995								
UNS	= 040000	1270*	5911	6434	7101	6917	9213							
UPE	= 020000	1305*	4215											
VV	= 000100	1271*	5911	7464	8917	9213								
WCE	= 040000	1290*	5937	7887	8923									
WLE	= 004000	1211*	5268	9504										
WRDATA=	000123	1213*												
WRHEAD=	000127	1309*	4345	4366										
WRL	= 004000	1214*	4373	4374	4410	4437	4439	5220	5274	6012	7257	7457	8060	8158
WRTCHK=	000131	8248	8392	9500	9879									
		3786	3903	4046*										
WRTEPK	012614	1328*												
WRTGAT=	040000	4313	4404*											
WRTVER	015234	4408	4413*	8287										
WVOS	015276	4303	4503*	8292										
WV1\$	016040	4419	4507*											
WV2\$	016052	4505	4529*											
WV5\$	016162	4530*												
WV9\$	016166	1947*	3424	6449*	8664*	8788*	8844*	8859*	8872*	8886*	8936*	9010*	9026*	9055*
W.DRV	001544	9077*	9111*	9422*	9485*	9488*	9541*	9551*	9561*	9866*	9900*			
		1897*	3431	8656	9624									
W.MILI	001470	1913*	9485	9551	9866									
W.MIN	001476	1896*	3431*	8654*	8656*									
W.MTIM	001466	1910*	9422	9900										
W.SEC	001472	1920*	3430*	6447*	8662	8669*	8789*	8843*	8858*	8871*	8887*	8935*	9009*	9025*
W.TIME	001512	9054*	9076*	9110*	9419*	9540*	9550*	9560*	9864*	9897*				
W.WCK =	000200	1487*	4266	4294	4296	4375	4407	4409	4420	4441	5222	5275	6014	6016

..SX	= 001000	1681*	1686											
..ADC	065755	11488	11579*											
..ASCII	065532	11451	11456	11461	11466	11471	11476	11481	11487	11495	11534*			
..BLNK	066000	11496	11583*											
..CR	065452	11450	11455	11460	11465	11470	11475	11480	11485	11486	11489	11490	11501	11507
		11508	11519*	11520										
..DUMP	065062	1670	11440*											
..HIGH	000226	1672*	11505											
..LOW	000224	1671*	11439	11449										
..OCT	065556	11454	11459	11464	11469	11474	11479	11484	11493	11498	11545*			
..RO	065640	11452	11565*											
..R1	065653	11457	11567*											
..R2	065656	11462	11569*											
..R3	065701	11467	11571*											
..R4	065714	11472	11573*											
..R5	065727	11477	11575*											
..SP	065742	11482	11577*											
..STCK	065060	11438*	11441*	11442	11483	11515								

CZR6PCD RK611/06 PERF EXEC
CZR6PC.P11 01-DEC-77 14:53

MACY11 30(1046) 01-DEC-77 15:04 PAGE 284
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0280

.EQUAT	1018#	1062				
.HEADE	1018#	1021				
.SETUP	1018#	3331				
.SWRHI	1018#	1046				
.SWRLO	1018#	1054#	1056	1058	1060	1061
.\$APT8	1740#					
.\$APTH	1018#	1676				
.\$APTY	1018#	10638				
.\$CATC	1018#	1656				
.\$CMTA	1018#	1698				
.\$READ	1018#	10074				
.\$SIZE	1018#	10390				
.\$TRAP	1018#	10695				
.\$TYPE	1018#	9951				

. ABS. 067107 000

ERRORS DETECTED: 0

RM03:CZR6PC, RM03:CZR6PC, SEQ/5, CORE NL: 700 000 EQ-ONEWSW=RM03:DRIV11.P11, RM03:DUMP.P11, RM03:CZR6PC.P11

RUN-TIME: 31 32 3 SECONDS

RUN-TIME RATIO: 2044 67=30.4

CORE USED: 42K (83 PAGES)

DOCUMENT PAGES: 280