

RK611/RK06

SUBSYSTEM VERIFY PART 2
MD-11-DZR6N-C

EP-DZR6N-C-DL-B
COPYRIGHT © 1976
FICHE 1 OF 2

DEC 1976
digital
MADE IN USA

This microfiche card contains a grid of frames. Each frame contains a small, high-contrast image of a document page, likely a technical manual or verification document. The frames are arranged in approximately 15 rows and 15 columns. The text within the frames is too small to be legible, but they appear to contain various diagrams, tables, and text blocks. The overall appearance is that of a dense data storage medium.

RK611/RK06

SUBSYSTEM VERIFY PART 2
MD-11-DZR6N-C

EP-DZR6N-C-DL-B
COPYRIGHT © 1976
FICHE 2 OF 2

DEC 1976
digital
MADE IN USA

This microfiche card contains a grid of frames. The leftmost column consists of 12 frames, each containing a small table with multiple columns and rows of data. The remaining frames in the grid are mostly blank or contain very faint, illegible text. The data in the first column appears to be organized in a structured format, possibly representing a list of items or a set of parameters.

B01

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6N.C.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 2

SEQ 0001

.REM 0

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZR6N-C-D
PRODUCT NAME: RK611/RK06 SUBSYSTEM VERIFICATION:PART 2
DATE: DECEMBER 1976
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: DAVE HOFFMAN

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

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

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

COPYRIGHT (C) 1976 BY DIGITAL EQUIPMENT CORPORATION

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6N.C.P11 05-OCT-76 10:07

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	HARDWARE REQUIREMENTS
2.1	REQUIREMENTS FOR SUBSYSTEM TESTS
2.2	REQUIREMENTS FOR HEAD ALIGNMENT AID
3.0	PRELIMINARY PROGRAM REQUIREMENTS
4.0	GENERAL PROGRAM CONSIDERATIONS
4.1	SYSMAC
4.2	XXDP
4.3	CHAIN MODE
4.4	DUMP MODE
4.5	ACT/APT
4.6	AUTOMATIC MODE
4.7	DUMP MODE
4.8	APT ETABLE DEFINITIONS
4.9	DUAL-ACCESS
4.10	MEMORY MANAGEMENT
4.11	MEMORY PARITY CHECK
4.12	BAD SECTORS
4.13	EXECUTION TIME
5.0	PROGRAM LOADING
6.0	STARTING PROCEDURE
6.1	STARTING ADDRESSES
6.2	SWITCH REGISTER OPTIONS USED
7.0	OPERATOR ACTION
8.0	PROGRAM ACTION
8.1	DESCRIPTION OF OPERATING PARAMETERS
8.2	SELECTION OF OPERATING PARAMETERS
8.3	DRIVE SELECTION
8.4	TEST SELECTION
8.5	LIST TESTS, (L)
8.6	CHANGE TESTS, (C)
8.7	INPUT PARAMETERS AND RUN TESTS, (I)
8.8	CONTROL Z (fz) FUNCTION
8.9	CONTROL C (fc) FUNCTION
8.10	PARAMETER LIST ALTERATION
8.11	TYPE LIST, (T)
8.12	OPEN LIST, (O)
8.13	SET INDIVIDUAL PARAMETER, (S)
8.14	RUN TESTS, (R)

DZREN.C.P11 05-OCT-76 10:07

001

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 4

SEQ 0003

100
05-OCT-76

8.2.3.5
8.2.3.6

CONTROL Z (IZ) FUNCTION
CONTROL C (IC) FUNCTION

TABLE OF CONTENTS (CONT'D)

00.3	SPECIAL PARAMETER SPECIFICATIONS
00.3.1	PT - DATA PATTERN SELECT WORD
00.3.2	CS - CONTROL SWITCH WORD
00.3	DATA PATTERNS
00.0	DESCRIPTION OF TESTS
00.1	TEST 1 - OFFSET-TO-FAILURE MEASUREMENT
00.0	NPR/MAIN MEMORY TESTS
00.1	TEST 2 - NPR/MEMORY WORD ADDRESSING TEST
00.2	TEST 3 - NPR/MEMORY BLOCK ADDRESSING TEST
00.3	TEST 4 - NPR/MEMORY DATA PATTERN TEST
00.4	TEST 5 - UNIBUS CONTENTION TEST
00.4	TEST 6 - MULTI-DRIVE INTERFERENCE TEST
10.0	ERROR REPORTING
10.1	COMMON ERRORS
10.2	ERROR HANDLING
10.3	ERROR PRINTOUT EXAMPLES
11.0	RK06 HEAD ALIGNMENT AID
11.1	HARDWARE REQUIREMENTS
11.2	OPERATIONAL MODES
11.2.1	MANUAL SELECT MODE
11.2.1.1	MANUAL SELECT ALIGNMENT
11.2.1.2	MANUAL SELECT VERIFY
11.2.1.3	MANUAL SELECT EXERCISE
11.2.2	AUTO SELECT MODE
11.2.2.1	AUTO SELECT ALIGNMENT
11.2.2.2	AUTO SELECT VERIFY
11.2.2.3	AUTO SELECT EXERCISE
11.3	ALIGNMENT AID ERROR MESSAGES
APPENDIX A	SAMPLE ADDRESS 200 DEFAULT RUN
APPENDIX B	SAMPLE ADDRESS 204 RUN
APPENDIX C	SAMPLE MANUAL MODE HEAD ALIGNMENT AID RUN
APPENDIX D	SAMPLE AUTO MODE HEAD ALIGNMENT AID RUN

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
148

1.0 ABSTRACT

THIS PROGRAM PERFORMS PART 2 OF THE RK611/RK06 SUBSYSTEM VERIFICATION TESTS, WHICH PROVIDE A FUNCTIONAL SHAKEDOWN OF THE ENTIRE SUBSYSTEM, INCLUDING THE UNIBUS INTERFACE AND ACCESS TO MAIN MEMORY. THE TESTING IN PART 2 EMPLOYS WORST-CASE SITUATIONS INVOLVING HEAD OFFSETTING, MEMORY ADDRESSING AND DATA TRANSFER, UNIBUS CYCLE CONTENTION, AND MULTIPLE DRIVE OPERATIONS. ADDITIONALLY, AN RK06 HEAD ALIGNMENT AID IS PROVIDED TO FACILITATE ON-LINE ALIGNMENT OF DRIVE HEADS.

WITHIN THE VARIOUS SUBSYSTEM TESTS, EMPHASIS IS GIVEN TO USEFUL SCOPE LOOPS AND OPERATOR SPECIFICATION OF TEST PARAMETER VALUES. AT THE BEGINNING OF TESTING, THE FOLLOWING OPTIONS MAY BE SPECIFIED BY THE USER:

RK611 REGISTER ADDRESS
RK06 VECTOR ADDRESS
RK06 PRIORITY LEVEL
DRIVE (S) TO BE TESTED
TEST (S) TO BE RUN
NUMBER OF TEST ITERATIONS
INITIAL DISK ADDRESS ON TRANSFERS
DATA PATTERNS USED
STALL TIME BETWEEN OPERATIONS

IN ADDITION, STANDARD SWITCH REGISTER OPTIONS ARE PROVIDED, FOR A VARIETY OF PROGRAM LOOPING, RUNNING, AND REPORTING MODES.

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

2.0 HARDWARE REQUIREMENTS

2.1 REQUIREMENTS FOR SUBSYSTEM TESTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN PART 2 OF THE SUBSYSTEM VERIFICATION TESTS:

PDP-11/04, (05, 10 MFG. ONLY), 20, 34, 35, 40, 45, 50, 70 OR PDQ
16 K MEMORY
CONSOLE TELETYPE
RK06 UNIBUS CONTROLLER (RK611)
1 TO 8 RK06 DRIVES
1 TO 8 RK06 DISK CARTRIDGES (FORMATTED IN 20 OR 22 SECTOR FORMAT)

2.2 REQUIREMENTS FOR HEAD ALIGNMENT AID

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
00
01
02
03
04

ADDITIONAL HARDWARE IS REQUIRED BY THE RK06 HEAD ALIGNMENT AID:

RK06 FIELD TEST BOX (OR ALIGNMENT SECTION THEREOF)
RK06 ALIGNMENT CARTRIDGE
RK06 HEAD ALIGNMENT TOOL

3.0 PRELIMINARY PROGRAM REQUIREMENTS

BEFORE RUNNING THE SUBSYSTEM VERIFICATION PROGRAM, THE RK611 CONTROLLER DIAGNOSTIC (MAINDEC-11-DZR6A THRU DZR6E AND DZR6K) AND RK06 DRIVE DIAGNOSTIC (MAINDEC-11-DZR6H THRU DZR6J) SHOULD FIRST BE RUN, TO RESOLVE BASIC, SOLID HARDWARE FAULTS.

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 SYSMAC

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

4.2 XXDP

THIS PROGRAM MAY BE LOADED UNDER XXDP. SUBSYSTEM TESTS 1-6 MAY BE RUN IN CHAIN OR DUMP MODE, BUT THE ALIGNMENT AID MAY ONLY BE RUN IN DUMP MODE.

4.2.1 CHAIN MODE

THIS PROGRAM IS DESIGNED TO BE RUN IN THE DEFAULT MODE WHEN CHAINED UNDER XXDP. THUS, INPUT DIALOGUE IS BYPASSED, AND RK06 UNIBUS ADDRESS AND INTERRUPT VECTOR ARE DEFAULTED TO STANDARD VALUES. ALL DRIVES PRESENT AND READY WILL BE TESTED WITH THE EXCEPTION OF DRIVE 0, IF THE RK06 CONTAINS THE XXDP MEDIUM.

4.2.2 DUMP MODE

THE PROGRAM CAN BE RUN IN DUMP MODE, WITH OR WITHOUT DEFAULT PARAMETERS. DRIVE 0 MAY BE TESTED, BUT IF IT CONTAINS THE XXDP MEDIUM, THE OPERATOR MUST REPLACE THE XXDP PACK WITH A FORMATTED SCRATCH PACK PRIOR TO TESTING (OR AN ALIGNMENT CARTRIDGE IF ALIGNMENT IS TO BE DONE ON DRIVE 0). A MESSAGE WILL INFORM THE OPERATOR WHEN

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
257
258
259
260

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

THIS IS NECESSARY.

4.3 ACT/APT

THE PROGRAM IS COMPATIBLE WITH ACT/APT CONVENTIONS. SUBSYSTEM TESTS 1-6 MAY BE RUN IN AUTOMATIC OR DUMP MODE, AND THE HEAD ALIGNMENT AID MAY ONLY BE RUN IN DUMP MODE.

4.3.1 AUTOMATIC MODE

THE PROGRAM PROVIDES FOR AUTOMATIC APT/ACT OPERATION. IN THIS MODE, PARAMETERS REVERT TO DEFAULT VALUES, AND ALL DRIVES WHICH ARE PRESENT AND READY ARE TESTED.

4.3.2 DUMP MODE

IN DUMP MODE, PARAMETERS MAY BE INPUT, OR DEFAULTED, AND SUBSYSTEM TESTS 1-6 OR HEAD ALIGNMENT AID MAY BE RUN.

4.3.3 APT ETABLE DEFINITIONS

THE FOLLOWING DEFINITIONS ARE VALID FOR SPECIFYING APT ENVIRONMENTAL TABLE (ETABLE) ENTRIES, VIA THE APT UTILITY PROGRAM :

1. SOFTWARE ENVIRONMENT

- =1 IF APT SCRIPT MODE
- =0 IF STANDALONE MODE

2. ENVIRONMENT MODE BYTE

- BIT 7 = 1 ETABLE DOES SIZING
- = 0 PROGRAM DOES SIZING
- BIT 6 = 1 SPOOL MESSAGES TO APT IF SCRIPT MODE
- = 0 DON'T SPOOL TO APT
- BIT 5 = 1 SUPPRESS CONSOLE OUTPUT
- = 0 ALLOW CONSOLE OUTPUT
- BITS 4-0 NOT USED

3. SWITCH 1 (SOFTWARE SWITCH REGISTER)

IF ENVIRONMENT MODE BIT 7 (SIZING BIT) IS SET TO 1, THE SOFTWARE SWITCH REGISTER WILL BE USED, INSTEAD OF THE HARDWARE CONSOLE SWITCH REGISTER. REGARDLESS OF WHICH ONE IS USED, ALL BITS DEFINED IN SECTION 6.2 (SWITCH REGISTER OPTIONS) MAY USED WHEN RUNNING IN STANDALONE MODE. IN APT SCRIPT MODE, HOWEVER, BIT 14 (LOOP ON TEST) MUST ALWAYS BE SET TO 0.

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
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372

- 4. SWITCH 2 (USER SWITCH REGISTER)
NOT USED
- 5. MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
NOT USED
- 6. INTERRUPT VECTOR 1
USED WHEN ENVIRONMENT MODE BIT 7 = 1. DEFAULT = 210
- 7. BUS PRIORITY 1
USED WHEN ENVIRONMENT MODE BIT 7 = 1. DEFAULT = 5
- 8. INTERRUPT VECTOR 2
NOT USED
- 9. BUS PRIORITY 2
NOT USED
- 10. BASE ADDRESS
USED WHEN ENVIRONMENT MODE BIT 7 = 1. DEFAULT = 177440
- 11. DEVICE MAP
USED WHEN ENVIRONMENT MODE BIT 7 = 1. EACH BIT SET TO 1 IN BITS
0-7 WILL SELECT THE CORRESPONDING DRIVE TO BE TESTED.
BITS 8-15 ARE NOT USED.

4.4 DUAL-ACCESS

THIS PROGRAM DOES NOT UTILIZE DUAL-ACCESS. FOR THE PURPOSES OF ALL TESTS (1-6), AND THE HEAD ALIGNMENT AID (SECTION 11), THE OPERATOR MUST GUARANTEE THAT THERE IS NO INTERFERENCE FROM THE UNUSED PORT. IF FAILURES ARE ENCOUNTERED IN TESTS 1-6 DUE TO INTERFERENCE FROM THE OTHER PORT, THE OPERATOR IS ADVISED TO SWITCH THAT PORT OFF-LINE AND RE-RUN THE TESTS.

4.5 MEMORY MANAGEMENT

THIS PROGRAM SUPPORTS MEMORY MANAGEMENT OPTIONS KT11C,D AND PDP11/70, FOR THE PURPOSES OF TESTS 2-4 ONLY. IN THESE TESTS (SEE SECTION 9.7) DIRECT ACCESS TO ALL OF PHYSICAL MEMORY ABOVE THE PROGRAM IS EXERCISED. FOR THE DURATION OF THESE 3 TESTS, MEMORY MANAGEMENT IS ENABLED. IN ALL OTHER TESTS, AND DURING THE USE OF THE HEAD ALIGNMENT AID, MEMORY MANAGEMENT IS DISABLED.

4.6 MEMORY PARITY CHECK

IF THE MEMORY PARITY CHECK OPTION IS AVAILABLE ON THE SYSTEM, ALL

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
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

TESTING IS DONE WITH PARITY CHECK ENABLED. PARITY ERRORS ARE REPORTED, AND TESTING IS ABORTED, UNLESS THE BYTE LABELED MEMABT WAS PREVIOUSLY LOADED WITH A NON-ZERO VALUE.

4.7 BAD SECTORS

ACCORDING TO A CONTROL SWITCH OPTION (SEE SECTION 8.2.4.2) THE BAD SECTOR FILE ON EACH DRIVE TO BE TESTED IS TYPED AT THE CONSOLE PRIOR TO TESTING. ALL DATA ERRORS OCCURRING IN THE PROGRAM ARE MASKED OUT IF THEY OCCUR ON SECTORS DESIGNATED AS BAD, (BY EITHER FACTORY OR SOFTWARE).

4.8 EXECUTION TIME

EXECUTION TIME IS DEPENDENT UPON PARAMETERS INPUT BY THE OPERATOR (SUCH AS STALL TIME OR ITERATION COUNT), AND TO A GREAT DEGREE, UPON THE AMOUNT OF MEMORY. HOWEVER, THE "AVERAGE TIME" REQUIRED TO RUN A QUICK VERIFICATION (FIRST PASS) IS 2 MINUTES (FOR A 64K SYSTEM). A SUBSEQUENT PASS WITH PARAMETERS DEFAULTED REQUIRES 3 MINUTES PER DRIVE (ON A 64-K SYSTEM).

5.0 PROGRAM LOADING

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

6.0 STARTING PROCEDURE

6.1 STARTING ADDRESSES

- 200 NORMAL STARTING ADDRESS OF TESTS 1-6 (PARAMETERS DEFAULTED)
- 204 SELECT OPERATING PARAMETERS, RK06 UNIBUS ADDR. AND INTERRUPT VECTOR FOR TESTS 1-6
- 224 HEAD ALIGNMENT AID START ADDRESS

NOTE

FOR HEAD ALIGNMENT AID OPERATING INFORMATION PLEASE SKIP TO SECTION 11. THE SECTIONS WHICH IMMEDIATELY FOLLOW APPLY ONLY TO THE SUBSYSTEM TESTS (1-6).

463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483

6.2 SWITCH REGISTER OPTIONS USED

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

BIT	OPTION
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR REPORTS
12	REPORT DESCRIPTION ONLY, ON ERRORS
11	INHIBIT ITERATIONS
10	BELL ON ERROR
09	LOOP ON ERROR
08	APPLY RANDOM STALL BETWEEN OPERATIONS
06	REPORT ONE ERROR PER TRANSFER IN TESTS 2-4
01	INHIBIT WRITES IN TEST 1
00	REPORT ALL SOFTWARE COMPARE ERRORS IN TESTS 2-4.

NOTE

FOR ADDITIONAL PROGRAM CONTROL OPTIONS, SEE DESCRIPTION OF CONTROL SWITCH WORD (CS), SECTION 8.2.4.2 .

7.0 OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 5.0)
2. LOAD A FORMATTED PACK ON EACH DRIVE TO BE TESTED.
3. BRING DRIVE (S) TO ONLINE STATE, WRITE ENABLED, AND LOCKED ON PORT.
4. LOAD THE DESIRED STARTING ADDRESS (SEE SECTION 6.1)
5. SET SWITCHES IF DESIRED (SEE SECTION 6.2)
6. PRESS START OR GIVE APPROPRIATE MONITOR START COMMAND.

8.0 PROGRAM ACTION

8.1 DESCRIPTION OF OPERATING PARAMETERS

AFTER THE PROGRAM IS STARTED, IT TYPES ITS IDENTIFICATION AS FOLLOWS: "DZR6N-C- RK11/RK06 SUBSYSTEM VERIFICATION:PART 2", FOLLOWED BY: "LAST PHYS MEM ADRS=XXXXXXXX". THEN, EITHER THE TESTS BEGIN EXECUTION

484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538

WITH DEFAULT PARAMETERS, OR AN OPERATOR INTERACTIVE MODE IS ENTERED (SEE STARTING ADDRESSES, SECTION 6.1). IN THIS MODE OPERATING PARAMETERS MAY BE LISTED OR ALTERED BY VARIOUS MEANS. ALL PARAMETERS ARE TREATED AS OCTAL NUMBERS, AND THE PARAMETER LIST IS SHARED AMONG THE TESTS.

THE FOLLOWING IS THE LIST OF OPERATING PARAMETERS, (WHICH APPLY TO THE CURRENT SELECTION OF DRIVES AND TESTS). AFTER EACH, IS INDICATED THE VALID RANGE OF VALUES FOR THAT PARAMETER (IN OCTAL), AND ITS DEFAULT VALUE. ALL PARAMETERS ARE ENTERED AS OCTAL NUMBERS, AND THE PARAMETER MNEMONICS SHOWN ARE THOSE USED BY THE PROGRAM.

- FC FIRST CYLINDER ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF CYLINDERS, FC=0-631, DEFAULT=0
- LC LAST CYLINDER ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF CYLINDERS, LC=0-632, DEFAULT=632
- FT FIRST TRACK ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF TRACKS, FT=0-2, DEFAULT=0
- LT LAST TRACK ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF TRACKS, LT=0-2, DEFAULT=2
- S0 FIRST SECTOR ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF SECTORS USED ON ANY DRIVE WHICH HAS A 20-SECTOR FORMAT CARTRIDGE. S0=0-23, DEFAULT=0.
- S1 LAST SECTOR ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF SECTORS USED ON ANY DRIVE WHICH HAS A 20-SECTOR FORMAT CARTRIDGE. S1=0-23, DEFAULT=23.
- S2 FIRST SECTOR ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF SECTORS, USED ON ANY DRIVE WHICH HAS A 22-SECTOR FORMATTED CARTRIDGE, S2=0-25, DEFAULT=0.
- S3 LAST SECTOR ADDRESS IN TESTS WHICH ARE DONE OVER A RANGE OF SECTORS, USED ON ANY DRIVE WHICH HAS A 22-SECTOR FORMATTED CARTRIDGE. S3=0-25, DEFAULT = 25.
- PT DATA PATTERN SELECT WORD (SEE SECTION 8.2.4.1 FOR DETAILS) PT=0-177777, DEFAULT=0
- CS CONTROL SWITCH WORD (SEE SECTION 8.2.4.2 FOR DETAILS) CS=0-000062, DEFAULT=0
- ST NUMBER OF UNIT STALL TIMES WITH WHICH TO STALL (DELAY) BETWEEN RK06 COMMANDS

8.2 SELECTION OF OPERATING PARAMETERS (ADDRESS 204 START)

539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594

8.2.1 DRIVE SELECTION

AFTER AN ADDRESS 204 START, THE PROGRAM INDICATES INPUT MODE BY TYPING "PARAMETER INPUT MODE". THEN, THE RK611 REGISTER ADDRESS, RK06 VECTOR ADDRESS AND RK06 PRIORITY ARE OPENED FOR POSSIBLE MODIFICATION AS FOLLOWS:

```

RK06 BUS ADR = 177440 NEW=(TYPE NEW VALUE HERE)
RK06 VEC ADR = 210 NEW=(TYPE NEW VALUE HERE)
RK06 PRIORITY= 5 NEW=(TYPE NEW VALUE HERE)

```

NEXT THE PROGRAM TYPES THE NUMBER(S) OF THE CURRENT DRIVES UNDER TEST, FOLLOWED BY A (*) ON THE NEXT LINE TO REQUEST NEW DRIVE NUMBERS. FOR EXAMPLE:

```

PARAMETER INPUT MODE

DRIVE(S)=0,1,2,4,7
*(INPUT, IF ANY, GOES HERE)

```

THE OPERATOR TYPES THE NEW NUMBERS (SEPARATED BY COMMAS) PLUS <CR> OR SIMPLY <CR> TO LEAVE THEM UNCHANGED. IF HE ENTERS DRIVE NUMBERS, THESE DRIVES ARE THEN CHECKED FOR VALID STATUS BY THE PROGRAM, AND ALL THOSE TYPED WHICH ARE ACCEPTABLE WILL NOW BE LISTED AS BEFORE. THE OPERATOR MAY MANUALLY RECONFIGURE THE DRIVES AND TYPE IN NEW DRIVE NUMBERS, AND THESE WILL BE CHECKED AND TYPED, UNTIL THE OPERATOR FINALLY TYPES JUST <CR>.

ON INITIAL ENTRY FROM ADDRESS 204, THE DRIVES WHICH ARE LISTED ARE ALL THOSE ON THE SUBSYSTEM WHICH ARE ON-LINE, READY, WRITE-ENABLED, AND LOADED WITH A CARTRIDGE OTHER THAN AN ALIGNMENT PACK. (THIS IS THE DEFAULT DRIVE SELECTION FOR ADDRESS 200 START). ON ALL OTHER ENTRIES TO THE DRIVE SELECTION ROUTINE (VIA <C> AS DESCRIBED IN SECTIONS 8.2.2.5 AND 8.2.3.6) THE PROGRAM FIRST TYPES :

```

TO TEST ALL DRIVES TYPE "A" <CR>, ELSE <CR>
*(CHARACTER GOES HERE)

```

NEXT, THE DRIVES ARE LISTED AS SHOWN ABOVE. IF (A) IS TYPED, ALL AVAILABLE DRIVES WILL BE LISTED, AND IF NOT, THE PREVIOUS SELECTION OF DRIVES WILL BE LISTED.

8.2.2 TEST SELECTION

WHEN THE OPERATOR FINALLY TYPES JUST <CR> IN RESPONSE TO THE DRIVE REQUEST, THE PROGRAM ENTERS THE TEST INPUT ROUTINE. THE FOLLOWING INSTRUCTIVE LINES ARE TYPED :

```

L = LIST TESTS
C = CHANGE TESTS
I = INPUT PARAMETERS AND RUN TESTS

```

595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650

THEN, THE FOLLOWING REQUEST IS MADE :

ENTER L,C, OR I
* (CHARACTER GOES HERE)

THE OPERATOR TYPES THE DESIRED CHARACTER PLUS <CR>.

8.2.2.1 LIST TESTS, (L)

IF (L) IS TYPED, THE PROGRAM TYPES THE CURRENT DESIRED TEST LIST, AS FOLLOWS :

TEST	ITERATIONS
1	0
2	177777
3	400
4	25
ETC.	

THE ITERATION NUMBER IS THE NUMBER OF TIMES THE TEST WILL BE RUN ON THIS PASS, AND IT MUST BE BETWEEN 0 AND 177777. IF IT IS ZERO, THAT TEST WILL NOT BE RUN. ON THE FIRST TIME THROUGH, ALL TESTS ARE LISTED WITH DEFAULT ITERATIONS (YET TO BE DETERMINED).

WHEN THE LIST IS COMPLETED, THE PROGRAM RETURNS TO TYPE : "ENTER L,C, OR I" AGAIN.

8.2.2.2 CHANGE TESTS, (C)

IF (C) IS TYPED, THE PROGRAM THEN ASKS: "TO DEFAULT TESTS TYPE D<CR>, ELSE <CR>". IF JUST <CR> IS TYPED, THE PROGRAM SEQUENTIALLY OPENS EACH TEST FOR POSSIBLE ALTERATION OF THE ITERATION NUMBER. EACH TEST IS OPENED AND A NEW LINE IS TYPED, GIVING THE TEST NO. AND ITER. NO. IN OCTAL, FOLLOWED BY (*) ON THE SAME LINE. FOR EXAMPLE :

TEST	ITERATIONS
0	0 * (INPUT, IF ANY, GOES HERE)

THE OPERATOR THEN TYPES THE NEW VALUE PLUS <CR>, OR JUST <CR> TO LEAVE IT UNCHANGED. IF THE OPERATOR TYPES A NUMBER FOLLOWED BY (!) (EXCLAMATION POINT), THE NUMBER JUST ENTERED WILL BE LOADED INTO THE ITERATION TABLE FOR ALL REMAINING TESTS. AFTER ALL TESTS HAVE BEEN OPENED, THE PROGRAM RETURNS TO TYPE : "ENTER L,C, OR I" AGAIN.

8.2.2.3 INPUT PARAMETERS AND RUN TESTS, (I)

IF (I) IS TYPED, THE PROGRAM ENTERS THE PARAMETER LIST ALTERATION

ROUTINE, DESCRIBED IN SECTION 8.2.3, IN WHICH OPERATING PARAMETERS MAY BE INPUT, AND TESTING BEGUN.

8.2.2.4 CONTROL Z (↑Z) FUNCTION

IF THE OPERATOR WISHES TO EXIT AT ANY TIME FROM L OR C MODE, AND RETURN TO SELECT A NEW MODE (L, C, OR I), CONTROL Z (↑Z) MAY BE TYPED.

8.2.2.5 CONTROL C (↑C) FUNCTION

IF THE OPERATOR WISHES TO TERMINATE L, C, OR I MODE, AND RETURN TO REQUEST NEW DRIVES AND TESTS, CONTROL C (↑C) MAY BE TYPED.

8.2.3 PARAMETER LIST ALTERATION

THE PROGRAM NEXT TYPES THE FOLLOWING INSTRUCTIVE LINES:

T = TYPE LIST
O = OPEN LIST
S = SET INDIVIDUAL PARAM.
R = RUN TESTS
ENTER T, O, S, OR R
* (CHARACTER GOES HERE)

THE OPERATOR TYPES THE DESIRED CHARACTER, PLUS <CR>.

8.2.3.1 TYPE LIST, (T)

IF (T) IS TYPED, THE PROGRAM TYPES THE CURRENT PARAMETER LIST, AS FOLLOWS:

FC = XXXXXX
LC = XXXXXX
ETC.

WHEN THE LIST IS COMPLETED, THE PROGRAM RETURNS TO TYPE "ENTER T, O, S, OR R" AGAIN.

8.2.3.2 OPEN LIST, (O)

IF (O) IS TYPED THE PROGRAM THEN ASKS: "TO DEFAULT ALL PARAMETERS TYPE D<CR>, ELSE <CR>". IF JUST <CR> IS TYPED, THE PROGRAM SEQUENTIALLY OPENS EACH OPERATING PARAMETER IN THE LIST FOR ALTERATION

6997
6998
6999
7000
7001
7002
7003
7004
7005
7006
7007
7008
7009
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
7044
7045
7046
7047
7048
7049
7050
7051
7052
7053
7054
7055
7056
7057
7058
7059
7060
7061
7062
7063
7064
7065
7066
7067
7068
7069
7070
7071
7072
7073
7074
7075
7076
7077
7078
7079
7080
7081
7082
7083
7084
7085
7086
7087
7088
7089
7090
7091
7092
7093
7094
7095
7096
7097
7098
7099
7100

(IN THE ORDER SHOWN IN SECTION 8.1). EACH PARAMETER IS OPENED AND A NEW LINE IS TYPED, GIVING THE PARAMETER AND ITS CURRENT VALUE (IN OCTAL), FOLLOWED BY (*) ON THE SAME LINE. FOR EXAMPLE:

IC=3 * (INPUT, IF ANY, GOES HERE)

THE OPERATOR THEN TYPES THE NEW VALUE PLUS A <CR> OR JUST <CR> TO LEAVE THAT PARAMETER UNCHANGED. AFTER ALL PARAMETERS HAVE BEEN OPENED FOR ALTERATION, THE PROGRAM RETURNS TO TYPE "ENTER T,O,S, OR R" AGAIN.

8.2.3.3 SET INDIVIDUAL PARAMETER, (S)

IF (S) IS TYPED, THE PROGRAM TYPES THE PROMPTER (>) ON THE NEXT LINE, AND WAITS FOR THE OPERATOR TO SET THE VALUE OF ANY INDIVIDUAL PARAMETER:

ENTER T,O,S, OR R

*S

> (ENTER PARAMETER AND VALUE HERE)

THE OPERATOR TYPES THE PARAMETER AND ITS NEW VALUE FOLLOWED BY <CR>, AND THE PROGRAM GIVES THE PROMPTER (>) AGAIN AS IN THE FOLLOWING EXAMPLE:

> FC = 600

> FS = 12

> IT = 1

> ETC.

TO TERMINATE INDIVIDUAL SETTING MODE, THE OPERATOR TYPES CONTROL Z (↑Z), AND THE PROGRAM RETURNS TO TYPE "ENTER T,O, S, OR R" AGAIN.

8.2.3.4 RUN TESTS, (R)

IF THE OPERATOR TYPES (R), EXECUTION OF TESTS BEGINS, USING ALL THE CURRENT PARAMETERS.

8.2.3.5 CONTROL Z (↑Z) FUNCTION

IF THE OPERATOR WISHES TO EXIT AT ANY TIME FROM T, O, S, OR R MODE (RUNNING TESTS), AND RETURN TO SELECT A NEW MODE (T,O,S, OR R), CONTROL Z (↑Z) MAY BE TYPED.

8.2.3.6 CONTROL C (↑C) FUNCTION

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

763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816

IF THE OPERATOR WISHES TO TERMINATE T, O, OR S MODE, OR STOP TEST EXECUTION (R MODE), CONTROL C (IC) MAY BE TYPED, AND THE PROGRAM WILL RETURN TO REQUEST NEW DRIVE SELECTION (SEE SECTION 8.2.1). (ON AN ADDRESS 200 DEFAULT RUN, TYPING CONTROL C CAUSES THE PROGRAM TO HALT AFTER COMPLETING ANY DATA TRANSFER).

8.2.4 SPECIAL PARAMETER SPECIFICATIONS

8.2.4.1 PT-DATA PATTERN SELECT WORD

THE OPERATOR SPECIFIES A WORD OF SIXTEEN BITS (6 OCT. DIGITS), TO SELECT UP TO ALL SIXTEEN OF THE PATTERNS, LISTED IN SECTION 9.5.1. THE NO. OF ANY BIT SET IN PT CORRESPONDS TO THE NO. OF A PATTERN CHOSEN. IF THE HIGH BIT OF PT (BIT 15) IS SET, THE USER WILL BE ASKED TO TYPE UP TO SIXTEEN DATA WORDS WHICH WILL BE LOADED INTO THE BUFFER FOR PATTERN 15. THE PROGRAM TYPES:

"SELECT USER DEFINED PATTERN 15"

NEXT, EACH OF UP TO 16 WORDS ARE REQUESTED BY THE PROGRAM IN THE SAME MANNER IN WHICH THE OPERATING PARAMETERS ARE OPENED FOR ALTERATION (SEE SECT. 8.2.3.2):

- WORD 0 = (OLD VALUE) * (NEW VALUE GOES HERE)
- WORD 1 = (OLD VALUE) * (NEW VALUE GOES HERE)
- ETC.

FOR EACH WORD, THE USER TYPES THE NEW VALUE PLUS <CR>, OR JUST <CR>, (TO LEAVE IT UNCHANGED). TO INPUT LESS THAN 16 WORDS FOR PATTERN 15, THE OPERATOR MAY TYPE (!) (EXCLAMATION POINT) AND THE PROGRAM WILL FILL UP THE REST OF THE PATTERN 15 BUFFER WITH THE LAST WORD SPECIFIED. NEXT, THE PROGRAM WILL CONTINUE, IN THE PROPER PARAMETER INPUT MODE (O OR S).

8.2.4.2 CS- CONTROL SWITCH WORD

IN ADDITION TO THE OPTIONS PROVIDED BY THE SWITCH REGISTER (HARDWARE OR SOFTWARE), A CONTROL SWITCH WORD (CS) MAY BE SPECIFIED AMONG THE OPERATING PARAMETERS TO PROVIDE THE FOLLOWING OPTIONS:

BIT	OPTION
---	-----
05	DROP DRIVE IF 20 (DEC) ERRORS EXCEEDED
04	TYPE BAD SECTOR FILES (BSF'S) ON FIRST PASS)
01	INHIBIT OFFSET REPORTS IN TEST 1

937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992

9.0 DESCRIPTION OF TESTS

9.1 TEST 1 - OFFSET-TO-FAILURE MEASUREMENT

IN THIS TEST A REPEATED PATTERN 072307(OCT) IS WRITTEN INTO ALL WORDS OF SECTORS FS AND LS ON TRACKS 0-2 OF CYLINDERS FC AND LC. THE SAME SECTORS ON ADJACENT CYLINDERS ARE THEN WRITTEN WITH 167230(OCT) (TO ESTABLISH A KNOWN PATTERN), BUT THEY ARE NOT TESTED. THEN, EACH DESIRED SECTOR IS WRITE CHECKED WITH INCREASING POSITIVE OFFSETS, UNTIL EITHER A "DATA TYPE" ERROR OCCURS, OR THE FULL RANGE OF POSITIVE OFFSETS HAVE BEEN APPLIED. THEN, THE SAME SECTOR IS WRITE CHECKED WITH INCREASING NEGATIVE HEAD OFFSETS, UNTIL FAILURE OR MAXIMUM NEGATIVE OFFSET IS APPLIED. THIS IS DONE FOR ALL THE SECTORS ORIGINALLY WRITTEN, IN THE FOLLOWING ORDER: SECTORS FS AND LS ON CYLINDERS FC AND LC FOR TRACK 0, THEN FOR TRACK 1, AND FINALLY FOR TRACK 2. THE OFFSET-TO-FAILURE RESULTS ARE REPORTED IN ORDER OF INCREASING TRACKS, AS FOLLOWS:

OFFSET-TO-FAILURE MEASUREMENTS:

TRACK	CYLN	SECT	-OFST (UIN)	-OFST (UIN)
X	XXX	XX	XXXX	XXXX
X	XXX	XX	XXXX	XXXX
X	XXX	XX	XXXX	XXXX

ETC.

NOTE

IN AN ADRS 200 DEFAULT RUN, FS DEFAULTS TO 0, AND LS DEFAULTS TO 12(OCT), FOR THIS TEST.

9.2 NPR / MAIN MEMORY TESTS

THIS GROUP OF TESTS EXERCISES ALL MEMORY ABOVE THE PROGRAM, VIA READ/WRITE NPR TRANSFERS WITH THE RK06. THE TESTS ARE DESIGNED TO DIAGNOSE MEMORY TRANSFER FAILURES DURING HEAVY NPR ACTIVITY, IN SYSTEMS PROVIDED WITH MEMORY MANAGEMENT (KT 11 C, D OR PDP 11/70), ALTHOUGH MEMORY MANAGEMENT IS NOT REQUIRED.

NOTE

THERE IS NO PROGRAM RELOCATION (EXCEPT FOR THE XXDP OR ABSOLUTE LOADER, IF

993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047

PRESENT). HENCE, MEMORY IN WHICH THE PROGRAM RESIDES IS NOT TESTED. ALSO, THE UNIBUS I/O ADDRESSES ARE NOT TESTED. IF IT IS DESIRED THAT THE LOADER BE DESTROYED, SO THAT ADDITIONAL MEMORY MAY BE TESTED, LOCATION 170 (KILLDR:) MAY MANUALLY BE SET TO A NON-ZERO VALUE.

THERE ARE 3 MEMORY TESTS PROVIDED. TESTS 2 AND 3 ARE TESTS OF MEMORY ADDRESSING CAPABILITY DURING DISK NPR'S, AND TEST 4 IS AN NPR/MEMORY DATA PATTERN TEST. DURING TESTING, MEMORY MANAGEMENT WILL BE ENABLED IF INSTALLED, AND ALL OPERATIONS WILL BE PERFORMED IN KERNAL MODE, WITH ADDRESS RELOCATION BEING DONE THROUGH MANIPULATION OF KERNAL PAGE ADDRESS REGISTERS.

9.2.1 TEST 2 - NPR/MEMORY WORD ADDRESSING TEST

STARTING AT THE FIRST ADDRESS OF THE NEXT 1 K MEMORY BLOCK BEYOND THE END OF THE READ/WRITE DATA BUFFER (RWBUF), WRITE UNIQUE NUMBERS (STARTING WITH 1) INTO EACH OF UP TO 64K WORDS (DEPENDING ON THE AMOUNT OF PHYSICAL MEMORY). THESE NUMBERS REFLECT UNIQUE ADDRESSES WITHIN THE 64K BLOCK. NEXT WRITE THE 64K WORDS (MAX) ONTO DISK AT FC, FS, FT (SCALED, IF NECESSARY, TO AVOID PACK OVERFLOW). THEN ZERO THE ENTIRE 64K BLOCK IN MEMORY, AND READ THE DATA BACK FROM THE DISK, AT THE SAME PHYSICAL ADDRESSES. COMPARE THE DATA READ TO THE DATA WRITTEN. TYPE THE FAILING VIRTUAL (IF MEM. MANAGEMENT) AND PHYSICAL ADDRESSES, AS WELL AS THE GOOD AND BAD DATA, FOR UP TO THE FIRST 10 (DECIMAL) FAILING LOCATIONS.

IF MEMORY MANAGEMENT IS PROVIDED AND THERE IS ADDITIONAL MEMORY TO TEST, REPEAT THE ABOVE ADDRESSING TEST FOR EACH OF THE REMAINING 64K PHYSICAL MEMORY BLOCKS.

9.2.2 TEST 3 - NPR/MEMORY BLOCK ADDRESSING TEST

IN THIS TEST, MEMORY BLOCK ADDRESSING CAPABILITY DURING RK06 NPR'S IS TESTED.

THE TEST BEGINS BY WRITING A SINGLE NUMBER INTO ALL LOCATIONS OF EACH MEMORY BLOCK OF UP TO 64K WORDS. THIS NUMBER UNIQUELY DEFINES EACH BLOCK. THEN WRITE EACH BLOCK ONTO DISK AT ADDRESS FC, FS, FT (SCALED, IF NECESSARY TO AVOID PACK OVERFLOW OR DESTRUCTION OF THE BAD SECTOR FILE). ZERO THE BLOCK IN MEMORY, AND THEN READ THE DATA BACK INTO THE PROPER PHYSICAL ADDRESSES, DO THIS FOR ALL 64K BLOCKS, AND THEN COMPARE THE CONTENTS OF ALL THE MEMORY TO EXPECTED VALUES. TYPE FAILURES AS DESCRIBED ABOVE FOR THE FIRST 10(DEC) FAILING LOCATIONS IN EACH 64K MEMORY BLOCK.

1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103

9.2.3 TEST 4 - NPR/MEMORY DATA PATTERN TEST

IN THIS TEST, ALL PHYSICAL MEMORY LOCATIONS ABOVE THE PROGRAM (NOT INCLUDING UPPER UNIBUS DEVICE ADDRESSES) ARE EXERCISED WITH UP TO 15 DATA PATTERNS, AS CHOSEN FROM THE FIRST 14 PATTERNS DESCRIBED IN SECTION 8.3, PLUS PATTERN 15 (USER DEFINED PATTERN). THE DATA PATTERNS DESIRED FOR THIS TEST ARE CHOSEN SEPARATELY, HOWEVER, AND THEY DEFAULT TO PATTERNS 8, 9, 10, AND 11.

MEMORY IS TESTED IN BLOCKS OF 64K, STARTING AT THE FIRST ADDRESS OF THE NEXT 1K MEMORY BLOCK BEYOND THE PROGRAM. EACH BLOCK OF UP TO 64K IS LOADED WITH DATA, WRITTEN ONTO DISK AT FC, FS, FT, LOADED WITH ZEROS, AND READ BACK AND COMPARED. THIS IS DONE FOR EACH OF THE DATA PATTERNS CHOSEN, INCLUDING AN OPTIONAL PATTERN CHOSEN BY THE USER.

9.3 TEST 5 - UNIBUS CONTENTION TEST

THE PURPOSE OF THIS TEST IS TO EXERCISE THE NPR CONTROL LOGIC WITHIN THE CONTROLLER IN SUCH A MANNER THAT A CONTENTION FOR MEMORY CYCLES ARISES BETWEEN THE PROCESSOR AND THE RK06.

FIRST, A WRITE DATA IS BEGUN ON CYLINDER FC, (SCALED TO AVOID PACK OVERFLOW) AT SECTOR 0, WITH THE BUS ADDRESS INCREMENT INHIBIT BIT SET IN THE CONTROLLER. USING A WORD COUNT OF 12000(OCT) OR 13000(OCT) (DEPENDING ON THE FORMAT), THE ENTIRE TRACK IS WRITTEN WITH THE FIRST WORD OF USER DEFINED PATTERN 15 (IF SELECTED) OR THE FIRST WORD OF PATTERN 13 (BY DEFAULT). THEN, AN INSTRUCTION LOOP IS ENTERED IN WHICH A NON-EXISTENT UNIBUS DEVICE ADDRESS IS REFERENCED ONCE PER PASS. THIS WILL RESULT IN A PROCESSOR SEIZURE OF THE UNIBUS, FOR 5-20 MICRO-SEC (DEPENDING ON THE PROCESSOR), FOLLOWED BY A NON-EXISTENT MEMORY PROCESSOR TRAP, ONCE PER PASS. THUS, FOR EACH PASS THROUGH THE INSTRUCTION LOOP, THE CONTROLLER IS FORCED TO LOSE FROM ONE TO FOUR NPR BUS CYCLES. THE EXECUTION TIME OF THE LOOP DETERMINES THE REPETITION RATE OF THE NON-EXISTENT MEMORY REFERENCES, AND THIS IS CHOSEN TO ALLOW ENOUGH NPR'S SO THAT THERE SHOULD BE NO DATA LATE ERRORS IN A FAULT-FREE CONTROLLER. THE DATA IS THEN WRITE CHECKED FOR VERIFICATION, AND THE ABOVE TEST IS REPEATED USING A READ DATA COMMAND. NO ERRORS ARE EXPECTED.

THE ABOVE OPERATIONS ARE REPEATED 15 MORE TIMES, USING THE NEXT WORD OF THE PROPER PATTERN (13 OR 15) EACH TIME, TO WRITE AND READ THE ENTIRE TRACK.

9.4 TEST 6 - MULTI-DRIVE INTERFERENCE TEST

THIS TEST PERFORMS MULTI-DRIVE POSITIONING OPERATIONS, WHILE A LARGE DATA TRANSFER IS IN PROGRESS ON THE DRIVE UNDER TEST, FOR THE PURPOSE OF DETECTING PROBLEMS OF CONCURRENT DRIVE OPERATION. THE TEST IS RUN ONLY IF THERE IS MORE THAN ONE DRIVE ON THE SUBSYSTEM.

1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159

THE TEST PROCEEDS AS FOLLOWS: IT IS FIRST DETERMINED WHICH DRIVE(S) (BESIDE THE ONE UNDER TEST) EXIST AND ARE OPERATIONAL. THESE DRIVES ARE THEN CLEARED AND A SEEK TO CYL 0 IS DONE ON EACH. NEXT, A SEEK IS DONE TO CYLINDER FC (SCALED, IF NECESSARY) ON THE DRIVE UNDER TEST. A SEEK IS NOW BEGUN ON EACH OF THE OTHER DRIVES, EACH TO A DIFFERENT PSEUDO-RANDOM CYLINDER ADDRESS. THIS SHOULD REQUIRE FROM 10-67 MILLI-SEC TO COMPLETE. AS SOON AS THESE SEEKS ARE BEGUN, A WRITE DATA IS BEGUN ON THE DRIVE UNDER TEST, AT THE CURRENT CYLINDER (FC), SECTOR 0, TRACK 0, WITH THE BUS ADDRESS INCREMENT INHIBIT BIT SET IN THE CONTROLLER, AND WITH A WORD COUNT OF 15,616 (DEC) WORDS IF 20 SECTOR FORMAT, OR 17,152 (DEC) WORDS IF 22 SECTOR FORMAT. THIS WORD COUNT AMOUNTS TO A FULL CYLINDER + 1 SECTOR OF DATA. THIS TRANSFER REQUIRES FROM 101-126 MILLI-SEC TO COMPLETE, AND IN THAT TIME, ALL THE OTHER DRIVES SHOULD HAVE COMPLETED POSITIONING. NEXT, A WRITE CHECK IS DONE TO VERIFY THE DATA WRITTEN ON THE DRIVE UNDER TEST, AND A CHECK IS MADE TO INSURE THAT ALL OTHER DRIVES WHICH PERFORMED SEEKS WERE ABLE TO COMPLETE THEM SUCCESSFULLY, BY CHECKING ATTENTION BITS AND COMPARING DRIVE CYLINDER ADDRESSES TO EXPECTED VALUES.

10.0 ERROR REPORTING

10.1 COMMON ERRORS

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

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

1160	
1161	
1162	
1163	
1164	
1165	
1166	
1167	
1168	
1169	
1170	
1171	
1172	
1173	
1174	
1175	
1176	
1177	
1178	
1179	
1180	
1181	
1182	
1183	
1184	
1185	
1186	
1187	
1188	
1189	
1190	
1191	
1192	
1193	
1194	
1195	
1196	
1197	
1198	
1199	
1200	
1201	
1202	
1203	
1204	
1205	
1206	
1207	
1208	
1209	
1210	
1211	
1212	
1213	
1214	
1215	

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

10.2 ERROR HANDLING

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

10.3 ERROR PRINTOUT EXAMPLES

EXAMPLE 1:

** WRITE CHECK ERROR
 TEST 2

PREVIOUS COMMAND:

DRIVE	CMND	CYLNR	TRACK	SECTOR	WD CNT
000000	000121	000016	000001	000000	175000

1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271

HI BA LO BA
000000 061566

CURRENT COMMAND:
ERR PC DRIVE CMND CYLNDR TRACK SECTOR WD CNT
041154 000000 000131 000016 000001 000006 175000
HI BA LO BA
000000 061566

PACK ADDRESS OF ERROR(S):
CYLNDR TRACK SECTOR
000016 000001 000006

EXAMPLE 2:

** DATA MISCOMPARE
TEST 2

PREVIOUS COMMAND:
DRIVE CMND CYLNDR TRACK SECTOR WD CNT
000000 000131 000016 000001 000000 175000
HI BA LO BA
000000 074000

CURRENT COMMAND:
ERR PC DRIVE CMND CYLNDR TRACK SECTOR WD CNT
033156 000000 000121 000016 000001 000000 175000
HI BA LO BA
000000 074000

PACK ADDRESS OF ERROR(S):
CYLNDR TRACK SECTOR
000016 000001 000000

WD #	GOOD	BAD	HI PHY	LO PHY	VRT AD	KIPAR6
000400	000401	125252	000000	075000	155000	000600
000401	000402	125252	000000	075002	155002	000600
000402	000403	125252	000000	075004	155004	000600
000403	000404	125252	000000	075006	155006	000600
000404	000405	125252	000000	075010	155010	000600
000405	000406	125252	000000	075012	155012	000600
000406	000407	125252	000000	075014	155014	000600
000407	000410	125252	000000	075016	155016	000600
000410	000411	125252	000000	075020	155020	000600
000411	000412	125252	000000	075022	155022	000600

11.0 RK06 HEAD ALIGNMENT AID

THIS PROGRAM IS PROVIDED AS A SOFTWARE AID TO HEAD ALIGNMENT OF THE

1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326

RK06 DRIVE, TO BE USED IN CONJUNCTION WITH THE RK06 FIELD TEST BOX, OR OTHER SUITABLE INDICATOR OF HEAD ALIGNMENT. THE PROGRAM OPERATES IN TWO MODES. THE FIRST IS MANUAL SELECT MODE, WHICH SELECTS A SPECIFIC DRIVE AND HEAD TO BE ALIGNED BY TTY INPUT AT THE CONSOLE. THE SECOND MODE IS AUTO-SELECT MODE, WHICH ALLOWS REMOTE DRIVE AND HEAD SELECTION THROUGH OPERATION OF DRIVE DUAL-PORT SELECT SWITCHES. IN EITHER MODE THE OPERATOR HAS THE OPTION TO EITHER PERFORM HEAD ALIGNMENT, VERIFY ALIGNMENT, OR REQUEST UP TO FIVE MINUTES OF RANDOM SEEK EXERCISES TO BE PERFORMED ON THE SPECIFIED DRIVE (S).

11.1 HARDWARE REQUIREMENTS

TO PERFORM HEAD ALIGNMENT, THE ALIGNMENT INDICATING DEVICE MUST BE CONNECTED TO THE DRIVE VIA THE ALIGNMENT CABLE WHICH ATTACHES TO THE RK06 READ/WRITE MODULE. THE ALIGNMENT INFORMATION FROM THE DRIVE MUST BE DISPLAYED ON A METER OR OTHER INDICATOR. EACH DRIVE TO BE ALIGNED MUST BE LOADED WITH AN RK06 ALIGNMENT CARTRIDGE, AND MUST BE WRITE-PROTECTED. UP TO EIGHT DRIVES PER CONTROLLER MAY BE ALIGNED, AND EACH DRIVE MAY BE OPERATED IN SINGLE-PORT MODE ONLY. FOR THE PURPOSE OF THE HEAD ALIGNMENT AND ALL SWITCH REGISTER BITS (HARDWARE OR SOFTWARE) ARE IGNORED.

11.2 OPERATIONAL MODES

THE PROGRAM COMMUNICATES WITH THE OPERATOR DOING THE ALIGNMENT THROUGH CONSOLE DIALOGUE. WHEN THE PROGRAM IS STARTED AT ADDRESS 224(OCTAL), IT TYPES IDENTIFICATION:

" *** RK06 HEAD ALIGNMENT AID ***
TYPE H(CR) FOR INFORMATION- OTHERWISE, TYPE (CR) "

NEXT, THE PROGRAM REQUESTS THE DESIRED MODE OF OPERATION:

" MANUAL OR AUTO MODE (M OR A) ? "

THE OPERATOR MAKES THE SELECTION, AND ENTERS ONE OF THE FOLLOWING MODES.

11.2.1 MANUAL SELECT MODE

IN THIS MODE, THE DRIVE(S) TO BE ALIGNED AND THE HEAD(S) TO BE SELECTED ARE SPECIFIED BY TTY INPUT. THIS MODE IS USEFUL FOR SYSTEMS WITH FEW DRIVES, WHICH ARE IN CLOSE PROXIMITY TO THE CONSOLE, AS IN TYPICAL FIELD INSTALLATIONS.

THE PROGRAM FIRST ECHOS THE MODE:

1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380

"* MANUAL SELECT MODE *"
AND TYPES "ENTER DRIVE NO. (0-7):".

THE OPERATOR TYPES THE DRIVE NUMBER, THE PROGRAM TYPES THE DRIVE SERIAL NUMBER, AND THEN ASKS "ALIGN, VERIFY, OR EXERCISE (A, V, OR E) ?". THE OPERATOR RESPONDS, AND THE PROGRAM ENTERS THE APPROPRIATE SUB-MODE.

11.2.1.1 MANUAL SELECT ALIGNMENT

ALIGNMENT IN MANUAL MODE PROCEEDS AS FOLLOWS. THE PROGRAM TYPES:

"* MANUAL SELECT ALIGNMENT *," FOLLOWED BY
" ENTER HEAD NO. (0-2):"

THE OPERATOR TYPES THE DESIRED HEAD NUMBER, AND THE PROGRAM UNLOADS THE HEADS ON THE DRIVE, IN ANTICIPATION OF OPERATOR MOUNTING OF THE HEAD ALIGNMENT TOOL. THE PROGRAM TYPES "TYPE<R> WHEN READY:". AFTER THE TOOL HAS BEEN MOUNTED THE OPERATOR TYPES <R>, AND THE PROGRAM LOADS HEADS AND RESPONDS WITH:

"HEADS POSITIONED AT CYLINDER 365(OCT)"
AND "HEAD X SELECTED"

THE POSITIONING TO CYLINDER 365 IS DONE IN SINGLE INCREMENT SEEKS, TO AVOID EXCESSIVE MOMENTUM ON THE POSITIONER, WHILE THE ALIGNMENT TOOL IS INSTALLED. THE OPERATOR MAY NOW PROCEED TO ALIGN THIS HEAD, AND THE PROGRAM LOOPS BACK TO ASK FOR ANOTHER HEAD NUMBER ON THIS DRIVE. IF THE OPERATOR TYPES CONTROL Z (↑Z) THE PROGRAM WILL LOOP BACK TO ASK FOR "ALIGN OR EXERCISE ?" (STILL IN THE SAME MODE). IF CONTROL C (↑C) IS TYPED, THE PROGRAM LOOPS FURTHER BACK TO ASK FOR A NEW DRIVE NUMBER (STILL IN THE SAME MODE). IF CONTROL R (↑R) IS TYPED, THE PROGRAM EXITS FROM THE CURRENT MODE, AND DOES A COMPLETE RESTART.

11.2.1.2 MANUAL SELECT VERIFY

VERIFY MODE IDENTIFIES ITSELF AS FOLLOWS :

"* MANUAL SELECT VERIFY *", FOLLOWED BY
"ENTER HEAD NO. (0-2) :"

THE VERIFY MODE IS IDENTICAL TO THE ALIGNMENT MODE, EXCEPT THAT UNLOADING AND LOADING OF HEADS IS NOT DONE BETWEEN HEAD SELECTIONS. THEREFORE, IN THIS MODE, THE OPERATOR MAY SELECT HEADS FOR QUICK ALIGNMENT VERIFICATION (USING THE ALIGNMENT METER), WITHOUT THE DELAYS OF UNLOADING AND LOADING HEADS. IN THIS MODE, THE OPERATOR WILL NOT BE ASKED TO "TYPE <R> WHEN READY".

11.2.1.3 MANUAL SELECT EXERCISE

WHEN THE EXERCISE SUB-MODE OF MANUAL SELECT MODE IS ENTERED, THE PROGRAM UNLOADS HEADS ON THE SELECTED DRIVE, AND TYPES "TYPE (R) WHEN READY". THE OPERATOR MUST THEN REMOVE THE ALIGNMENT TOOL PRIOR TO THE RANDOM SEEKS WHICH FOLLOW, AND ALLOW EXERCISES TO PROCEED BY TYPING (R). THE PROGRAM THEN LOADS HEADS, AND RESPONDS WITH:

"* SEEK EXERCISES IN PROGRESS ON DRIVE X *".

AT THIS POINT, 7500 RANDOM SEEKS ARE BEGUN ON THE DRIVE, WHICH LASTS FOR ABOUT FIVE MINUTES. UPON COMPLETION OF THESE EXERCISES, THE PROGRAM RETURNS TO ASK FOR "ALIGN OR EXERCISE ?" (STILL IN MANUAL MODE) ON THIS DRIVE. WHILE IN EXERCISE SUB-MODE, THE CHARACTERS (Z), (C), AND (R) PERFORM THE SAME FUNCTIONS AS DESCRIBED IN SECTION 11.2.1.1.

11.2.2 AUTO SELECT MODE

IN THIS MODE, ALL SELECTION OF DRIVES AND HEADS FOR ALIGNMENT IS DONE BY SEQUENTIAL OPERATION OF DRIVE PORT SELECT SWITCHES. THIS ALLOWS COMMUNICATION WITH THE PROGRAM TO BE REMOTE FROM THE CONSOLE. THIS AUTO SELECT MODE IS WELL SUITED TO THE MANUFACTURING TEST ENVIRONMENT, OR TO A FIELD INSTALLATION, WHICH HAS A NUMBER OF REMOTE DRIVES. THE PROGRAM FIRST ECHOS THE MODE:

" * AUTO SELECT MODE * ", AND ASKS
"ALIGN OR EXERCISE (A OR E) ?".

THE OPERATOR RESPONDS, AND THE PROGRAM ENTERS THE APPROPRIATE SUB-MODE.

11.2.2.1 AUTO SELECT ALIGNMENT

ALIGNMENT IN AUTO SELECT MODE PROCEEDS AS FOLLOWS. THE PROGRAM TYPES:

" * AUTO SELECT ALIGNMENT * ".

THE PROGRAM CONSTANTLY SCANS ALL DRIVES 0-7 TO DETERMINE THE EXISTENCE OF EACH. WHENEVER A DRIVE IS DESELECTED BY THE OPERATION OF ITS PORT SELECT SWITCH, THE PROGRAM RECEIVES A NON-EXISTENT DRIVE INDICATION WHEN IT SELECTS THAT DRIVE. THEN, WHEN THE PROGRAM DETERMINES THAT A DRIVE WHICH WAS PREVIOUSLY OFF-LINE (DESELECTED) HAS JUST BECOME ON-LINE (SELECTED), IT PREPARES FOR ALIGNMENT ON THAT DRIVE. IN SUMMARY, THE OPERATOR SELECTS A DRIVE FOR ALIGNMENT BY DEPRESSING AND THEN RELEASING THE PORT SELECT SWITCH (FOR THE PORT IN USE) ON THAT DRIVE.

THE PROGRAM RECOGNIZES THE DRIVE SELECTED, AND CHECKS IT TO BE SURE

THAT IT IS WRITE-LOCKED. THE PROGRAM THEN TYPES:

"DRIVE X SELECTED"

AND THEN IT UNLOADS THE HEADS ON THAT DRIVE, SO THAT THE OPERATOR CAN MOUNT THE HEAD ALIGNMENT TOOL. WHEN IT IS MOUNTED THE OPERATOR MUST DEPRESS AND RELEASE THE PORT SWITCH AGAIN, AND THE PROGRAM LOADS HEADS, SEEKS IN SINGLE INCREMENTS OUT TO THE ALIGNMENT CYLINDER (365 OCT.), AND TYPES:

"HEADS POSITIONED AT CYLINDER 365(OCT.)
HEAD 0 SELECTED".

AFTER THE OPERATOR ALIGNS HEAD 0, HE MAY PROCEED TO HEAD 1, BY DEPRESSING AND RELEASING THE PORT SELECT SWITCH. WHEN HE DOES, THE HEADS WILL BE UNLOADED AGAIN, AND THE OPERATOR MAY MOVE THE ALIGNMENT TOOL TO HEAD 1, AND UPON DEPRESSING AND RELEASING THE PORT SWITCH AGAIN, THE HEADS WILL BE LOADED, THE PROGRAM WILL SEEK TO CYLINDER 365(OCT), AND TYPE:

"HEADS POSITIONED AT CYLINDER 365(OCT)
HEAD 1 SELECTED".

THIS PROCESS MAY BE CONTINUED WITH HEADS BEING SELECTED IN THE ORDER 0, 1, 2, 0, 1, 2, 0, ETC. UNTIL THE OPERATOR HAS COMPLETED THE ALIGNMENT.

WHEN THE OPERATOR COMPLETES ALIGNMENT ON THIS DRIVE, HE SHOULD REMOVE THE ALIGNMENT TOOL, AND SWITCH THE DRIVE ON-LINE. IF HE THEN WISHES TO SELECT ANOTHER DRIVE, HE MUST SWITCH THE DESIRED DRIVE OFF-LINE AND THEN ON-LINE AGAIN TO INITIATE SELECTION. WHEN ALL DESIRED DRIVES HAVE BEEN ALIGNED THE OPERATOR TYPES ↑Z OR ↑R, AS DESCRIBED IN SECTION 11.2.1.1. BEFORE LEAVING ALIGNMENT MODE AND PROCEEDING TO DO SEEK EXERCISES, THE OPERATOR MUST BE SURE TO REMOVE THE ALIGNMENT TOOL(S) FROM ALL DRIVE(S).

NOTE

THE ↑C FUNCTION DOES NOT APPLY IN AUTO MODE.

11.2.2.2 AUTO SELECT VERIFY

VERIFY MODE IDENTIFIES ITSELF AS FOLLOWS :

"* AUTO SELECT VERIFY *"

THE AUTO SELECT VERIFY MODE IS IDENTICAL TO THE AUTO ALIGNMENT MODE,

1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492

EXCEPT THAT UNLOADING AND LOADING OF HEADS IS NOT DONE BETWEEN HEAD SELECTIONS. THEREFORE, IN THIS MODE, THE OPERATOR MAY SELECT HEADS FOR QUICK ALIGNMENT VERIFICATION (USING THE ALIGNMENT METER), WITHOUT THE DELAY OF UNLOADING AND LOADING HEADS.

11.2.2.3 AUTO SELECT EXERCISE

IN AUTO SELECT EXERCISE SUB MODE, RANDOM SEEK EXERCISES ARE PERFORMED UPON ALL DRIVES WHICH ARE ON-LINE. DRIVES ARE EXERCISED SEQUENTIALLY, STARTING WITH DRIVE 0. THE OPERATOR SPECIFIES EITHER LONG EXERCISES (7500 SEEKS- 5 MINUTES) OR SHORT EXERCISES (1500 SEEKS- 1 MINUTE) TO BE PERFORMED UPON EACH DRIVE.

AUTO SELECT EXERCISE MODE IDENTIFIES ITSELF AS FOLLOWS:

" * AUTO SELECT EXERCISES * "
FOLLOWED BY "SHORT OR LONG (S OR L) ?".

THE OPERATOR TYPES HIS RESPONSE, AND THE SELECTED SEEK EXERCISES ARE PERFORMED ON EACH DRIVE. UPON COMPLETION OF THESE EXERCISES, THE PROGRAM RETURNS TO ASK FOR "ALIGN OR EXERCISE?" (STILL IN AUTO MODE). WHILE IN EXERCISE SUB-MODE THE CHARACTERS !Z AND !R PERFORM THE SAME FUNCTIONS AS DESCRIBED IN SECTION 11.2.1.1.

NOTE

BEFORE REQUESTING AUTO-EXERCISE MODE, THE OPERATOR MUST REMOVE THE ALIGNMENT TOOL (WHILE STILL IN AUTO-ALIGNMENT MODE) WHILE THE HEADS ARE UNLOADED.

11.3 ALIGNMENT AID ERROR MESSAGES

1. "DRIVE X NOT READY! PLEASE START IF DESIRED, THEN PRESS 'CONT' ON CPU WHEN READY." - THIS INDICATES THAT A DRIVE SELECTED FOR ALIGNMENT WAS FOUND NOT READY.

2. "DRIVE X NOT WRITE-LOCKED! PLEASE SET WRITE LOCK SWITCH. PRESS 'CONT' ON CPU WHEN READY". - THIS INDICATES THAT A DRIVE SELECTED FOR ALIGNMENT WAS FOUND NOT TO BE WRITE-PROTECTED.

3. "PLEASE LOAD ALIGNMENT CARTRIDGE ON DRIVE X! PRESS 'CONT' ON

1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546

1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572

CPU WHEN READY." - THIS INDICATES THAT A DRIVE SELECTED FOR ALIGNMENT WAS NOT LOADED WITH AN ALIGNMENT CARTRIDGE.

4. "MULTIPLE DRIVES AUTO-SELECTED! PRESS 'CONT' TO RESTART."
- THIS INDICATES THAT IN AUTO ALIGNMENT MODE MORE THAN ONE OF THE DRIVES WERE SELECTED FOR ALIGNMENT SIMULTANEOUSELY.

5. "CANNOT READ BAD SECTOR FILE ON DRIVE X! PRESS 'CONT' ON CPU TO RESTART". - THIS INDICATES THAT A READ ERROR WAS ENCOUNTERED WHILE THE PROGRAM WAS ATTEMPTING TO IDENTIFY THE CARTRIDGE ON DRIVE X AS AN ALIGNMENT CARTRIDGE.

6. "? X" - THIS IS TYPED BY THE PROGRAM IN RESPONSE TO THE INPUT OF AN INVALID PARAMETER, SHOWN HERE AS X. THE OPERATOR SHOULD NOW ENTER THE PROPER VALUE.

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
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627

APPENDIX A

SAMPLE ADDRESS 200 DEFAULT RUN

THE FOLLOWING PRINTOUT WAS THE RESULT OF RUNNING SUBSYSTEM VERIFICATION, PART 2, ON A SINGLE DRIVE (DRIVE 0), FOR THE FIRST PASS OF THE PROGRAM (QUICK-VERIFY PASS). IN THIS MODE, ALL MESSAGES APPEARING HERE WERE OUTPUTS--NO OPERATOR INPUTS ARE PROVIDED ON AN ADDRESS 200 RUN.

DZR6N-C - RK611/RK06 SUBSYSTEM VERIFICATION : PART 2

LAST PHYS MEM ADR = 377776

DRIVE 1 NON-EXISTENT
DRIVE 2 NON-EXISTENT
DRIVE 3 NON-EXISTENT
DRIVE 4 NON-EXISTENT
DRIVE 5 NON-EXISTENT
DRIVE 6 NON-EXISTENT
DRIVE 7 NON-EXISTENT

DRIVE(S) = 0

TESTING DRIVE 0
DRIVE SER. NO. 8
CART. SER. NO. 334

OFFSET-TO-FAILURE MEASUREMENTS:

TRACK	CYLN	SECT	+OFST (UIN)	-OFST (UIN)
0	0	0	NONE	NONE
0	0	12	NONE	1175
0	631	0	NONE	NONE
0	631	12	NONE	NONE
1	0	0	1050	NONE
1	0	12	NONE	1200
1	631	0	NONE	NONE
1	631	12	NONE	NONE
2	0	0	NONE	NONE
2	0	12	NONE	NONE
2	631	0	NONE	NONE
2	631	12	NONE	NONE

END PASS # 1

1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683

APPENDIX B
SAMPLE ADDRESS 204 RUN

THE FOLLOWING PRINTOUT WAS THE RESULT OF RUNNING SUBSYSTEM VERIFICATION, PART, 2, ON A SINGLE DRIVE (DRIVE 0), WITH PARTICULAR TESTS SELECTED TO BE RUN FOR SPECIFIED NUMBERS OF TIMES, WITH SPECIFIED PARAMETERS, AND A SPECIFIED NO. OF PROGRAM PASSES. SPECIFICALLY, IT WAS DESIRED THAT TEST 1 BE RUN ONCE, TEST 4 ONCE, AND TEST 6 ONCE. THESE TESTS WERE SPECIFIED TO RUN FOR 1 PROGRAM PASS, USING PARAMETERS SPECIFIED BY THE OPERATOR. IN THIS CASE, SECTOR ADDRESS LIMITS (S2 AND S3) AND DATA PATTERN (PT) WERE SPECIFIED AS NON-DEFAULT VALUES.

DZR6N-C - RK611/RK06 SUBSYSTEM VERIFICATION:PART 2

LAST PHYS MEM ADR=377776

PARAMETER INPUT MODE

RK06 BUS ADR = 177440 NEW =
RK06 VEC ADR = 210 NEW =
RK06 PRIORITY = 5 NEW =

DRIVE(S) = 0
*

L = LIST TESTS
C = CHANGE TESTS
I = INPUT PARAMETERS AND RUN TESTS

ENTER L,C, OR I
* C
TO DEFAULT TESTS TYPE D <CR>, ELSE <CR>
*

TEST	ITERATIONS
1	1
2	0
3	0
4	1

1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739

S 2 * 0
6 100 * 1

ENTER L,C, OR I
* L

TEST	ITERATIONS
S	0
S	0
S	1
S	0
S	1

ENTER L,C, OR I
* I

T = TYPE PARAMETER LIST
O = OPEN PARAMETER LIST
S = SET INDIVIDUAL PARAM
R = RUN TESTS

ENTER T,O,S, OR R
* T

FC=0
LC=632
FT=0
LT=2
S0=0
S1=23
S2=0
S3=25
PT=0
CS=0
ST=0

ENTER T,O,S, OR R
* S

> S2=4
> SV=20
SV=20?
> S3=20
> PT=100000
TO MODIFY PATTERN 15, TYPE M <CR>, ELSE <CR>
* M

MODIFY USER-DEFINED PATTERN 15:
WORD 00 = 072307 * 123456!
> 12

ENTER T,O,S, OR R
* T
FC=0

1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795

LC=632
TT=0
LT=2
SO=0
S1=23
S2=4
S3=20
PT=100000
CS=0
ST=0

USER-DEFINED PATTERN 15:

WORD 00 = 123456
WORD 01 = 123456
WORD 02 = 123456
WORD 03 = 123456
WORD 04 = 123456
WORD 05 = 123456
WORD 06 = 123456
WORD 07 = 123456
WORD 10 = 123456
WORD 11 = 123456
WORD 12 = 123456
WORD 13 = 123456
WORD 14+Z
= 123456

ENTER T, O, S, OR R
* R

ENTER NO. OF PASSES (1-77777) :
* 1

TESTING DRIVE 0
DRIVE SER. NO. 8
CART. SER. NO. 334

OFFSET-TO-FAILURE			MEASUREMENTS:	
TRACK	CYLN	SECT	+OFST (UIN)	-OFST (UIN)
0	0	4	NONE	NONE
0	0	20	NONE	NONE
0	631	4	NONE	NONE
0	631	20	NONE	NONE
1	0	4	NONE	NONE
1	0	20	NONE	NONE
1	631	4	NONE	NONE
1	631	20	NONE	NONE
2	0	4	NONE	NONE
2	0	20	NONE	NONE
2	631	4	NONE	NONE
2	631	20	NONE	NONE

Handwritten initials

K03

MD-11-DZREN-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZREN.C.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 37

SEQ 0036

1796
1797
1798
1799
1800
1801
1802

END PASS # 1
TO TEST ALL DRIVES TYPE "A" <CR>, ELSE <CR>
*

APPENDIX C

SAMPLE MANUAL MODE HEAD ALIGNMENT AID RUN

THE FOLLOWING PRINTOUT WAS THE RESULT OF RUNNING THE HEAD ALIGNMENT AID, PROVIDED IN SUBSYSTEM VERIFICATION PART 2 (ADDRESS 224 START). MANUAL MODE WAS REQUESTED, AND THE OPERATOR SELECTED DRIVE 0 FOR ALIGNMENT. IN THIS CASE, THE OPERATOR SELECTED HEADS 1, 0, AND 2, IN THAT ORDER, AND THEN REQUESTED RANDOM SEEK EXERCISES TO BE RUN ON THE DRIVE (AFTER REMOVAL OF THE HEAD ALIGNMENT TOOL). THE EXERCISES WOULD HAVE NORMALLY RUN FOR 5 MINUTES BUT THEY WERE PREMATURELY TERMINATED BY THE OPERATOR, BY TYPING (↑Z), IN THIS PARTICULAR RUN. FINALLY, VERIFY MODE WAS REQUESTED, AND HEAD 2 WAS SELECTED BY THE OPERATOR.

DZR6N-C - RK611/RK06 SUBSYSTEM VERIFICATION : PART 2

*** RK06 HEAD ALIGNMENT AID ***
FOR HELP TYPE H, ELSE <CR>

MANUAL OR AUTO MODE (M OR A)?
M

* MANUAL SELECT MODE *

ENTER DRIVE NO. (0-7):

0
DRIVE SER. NO. 8

ALIGN, VERIFY, OR EXERCISE (A, V, OR E)?
A

* MANUAL SELECT ALIGNMENT *

ENTER HEAD NO. (0-2):

1
TYPE <R> WHEN READY:

R
HEADS POSITIONED AT CYLINDER 365 (OCT)
HEAD 1 SELECTED

ENTER HEAD NO. (0-2):

1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858

1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901

0
TYPE <R> WHEN READY:
R
HEADS POSITIONED AT CYLINDER 365 (OCT)
HEAD 0 SELECTED

ENTER HEAD NO. (0-2):
2
TYPE <R> WHEN READY:
R
HEADS POSITIONED AT CYLINDER 365 (OCT)
HEAD 2 SELECTED

ENTER HEAD NO. (0-2):
↑Z
ALIGN, VERIFY, OR EXERCISE (A, V, OR E) ?

E
TYPE <R> WHEN READY:
R

*RANDOM SEEK EXERCISES IN PROGRESS ON DRIVE 0 *
↑Z

ALIGN, VERIFY, OR EXERCISE (A, V, OR E) ?
V

* MANUAL SELECT VERIFY *

ENTER HEAD NO. (0-2) :
2
HEADS POSITIONED AT CYLINDER 365 (OCT)
HEAD 2 SELECTED

ENTER HEAD NO. (0-2) :
↑Z
ALIGN, VERIFY, OR EXERCISE (A, V, OR E) ?

APPENDIX D

SAMPLE AUTO MODE HEAD ALIGNMENT AID RUN

THE FOLLOWING PRINTOUT WAS THE RESULT OF RUNNING THE HEAD ALIGNMENT AID, PROVIDED IN SUBSYSTEM VERIFICATION PART 2 (ADDRESS 224 START). AUTO MODE WAS REQUESTED, AND THE OPERATOR SELECTED DRIVE 0 FOR ALIGNMENT. IN THIS CASE, HEADS 0,1,2, AND 0 (AGAIN) WERE SELECTED FOR ALIGNMENT (IN AUTO MODE, THE HEADS ARE ALWAYS SELECTED IN THE ORDER 0,1,2,0,1,2, ETC.). BETWEEN HEAD SELECTIONS, THE HEADS WERE UNLOADED TO ALLOW REMOVAL OR INSTALLATION OF THE HEAD ALIGNMENT TOOL. NOTE THAT ALL DRIVE AND HEAD SELECTION (IN AUTO MODE) IS DONE BY THE OPERATOR, AT THE DRIVES, BY OPERATION OF PORT SELECT SWITCHES. AFTER ALIGNING HEADS, THE OPERATOR REQUESTED RANDOM SEEK EXERCISES TO BE RUN (ON ALL DRIVES, IN THIS CASE DRIVE 0). SHORT EXERCISES WERE REQUESTED, WHICH RUN FOR 1 MINUTE PER DRIVE. FINALLY, AUTO SELECT VERIFY WAS REQUESTED, AND HEADS 0,1, AND 2 WERE SELECTED SEQUENTIALLY.

DZR6N-C - RK611/RK06 SUBSYSTEM VERIFICATION : PART 2

*** RK06 HEAD ALIGNMENT AID ***
 FOR HELP TYPE H, ELSE <CR>

MANUAL OR AUTO MODE (M OR A)?
 A

* AUTO SELECT MODE *

ALIGN, VERIFY, OR EXERCISE (A, V, OR E) ?
 A

* AUTO SELECT ALIGNMENT *

DRIVE 0 SELECTED

HEADS POSITIONED AT CYLINDER 365 (OCT)
 HEAD 0 SELECTED
 HEADS POSITIONED AT CYLINDER 365 (OCT)
 HEAD 1 SELECTED
 HEADS POSITIONED AT CYLINDER 365 (OCT)
 HEAD 2 SELECTED

1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957

.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

000011	HT=	11	::CODE FOR HORIZONTAL TAB
000012	LF=	12	::CODE FOR LINE FEED
000015	CR=	15	::CODE FOR CARRIAGE RETURN
000200	CRLF=	200	::CODE FOR CARRIAGE RETURN-LINE FEED
177776	PS=	177776	::PROCESSOR STATUS WORD
177774	.EQUIV PS,PSW		
177772	STKLMT=	177774	::STACK LIMIT REGISTER
177570	PIRQ=	177772	::PROGRAM INTERRUPT REQUEST REGISTER
177570	DSWR=	177570	::HARDWARE SWITCH REGISTER
177570	DDISP=	177570	::HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

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

;*PRIORITY LEVEL DEFINITIONS

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

;*SWITCH REGISTER SWITCH DEFINITIONS

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

2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
100
101
102
103
104
105
106
107
108
109
110
111
112


```

2170
2171      177572      SR0= 177572
2172      177574      SR1= 177574
2173      177576      SR2= 177576
2174      172516      SR3= 172516
2175
2176      ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
2177
2178      172300      KIPDR0= 172300
2179      172302      KIPDR1= 172302
2180      172304      KIPDR2= 172304
2181      172306      KIPDR3= 172306
2182      172310      KIPDR4= 172310
2183      172312      KIPDR5= 172312
2184      172314      KIPDR6= 172314
2185      172316      KIPDR7= 172316
2186
2187      ;*KERNEL "I" PAGE ADDRESS REGISTERS
2188
2189      172340      KIPAR0= 172340
2190      172342      KIPAR1= 172342
2191      172344      KIPAR2= 172344
2192      172346      KIPAR3= 172346
2193      172350      KIPAR4= 172350
2194      172352      KIPAR5= 172352
2195      172354      KIPAR6= 172354
2196      172356      KIPAR7= 172356
2197
2198      170200      MAPLO0=170200
2199      170202      MAPH00=170202
2200      172100      MEMCSR=172100      ;MEMORY CSR REG START ADRS
2201      177740      LOERAD=177740      ;11/70 MEM LO ERROR ADRS REG
2202      177742      HIERAD=177742      ;11/70 MEM HI ERROR ADRS REG
2203      177744      MEMSYS=177744      ;11/70 MEM SYSTEM REG
2204      .SBTTL TRAP CATCHER
2205
2206      000000      .=0
2207      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
2208      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
2209      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
2210
2211      000174      000174      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
2212      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
2213
2214      000200      000137      013454      .SBTTL STARTING ADDRESS(ES)
2215      000170      000170      JMP @#DFSTRT      ;;JUMP TO STARTING ADDRESS OF PROGRAM
2216      000170      000000      KILLDR: .WORD 0      ;IF NOT = 0, IT IS OK TO OVERLAY LOADER
2217      000204      000204      .=204
2218      000204      000137      013510      JMP @#PSTART
2219      000224      000224      .=224
2220      000224      000137      023634      JMP @#ASTART
2221      000230      000700      ..LOW: .WORD 700
2222      000232      001100      ..HIGH: .WORD 1100
2223
2224      .SBTTL ACT11 HOOKS
2225
2226      ;*****

```

```

2226 ;HOOKS REQUIRED BY ACT11
2227 $SVPC=. ;SAVE PC
2228 =46
2229 000046 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
2230 023600
2231 000052 .=52
2232 140000 .WORD 140000 ;;2)SET LOC.52 TO 140000
2233 000234 .=$SVPC ;; RESTORE PC
2234 001000 .=1000
2235 .SBTTL APT PARAMETER BLOCK
2236
2237 ;*****
2238 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
2239 ;*****
2240 001000 .SX=. ;;SAVE CURRENT LOCATION
2241 000024 =24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
2242 000200 200 ;;FOR APT START UP
2243 000044 =44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
2244 000044 $APTHDR ;;POINT TO APT HEADER BLOCK
2245 001000 .=.SX ;;RESET LOCATION COUNTER
2246 ;*****
2247 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
2248 ;INTERFACE SPEC.
2249
2250 001000 $APTHD:
2251 001000 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
2252 001002 001320 $MADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
2253 001004 001130 $STMT: .WORD 1130 ;;RUN TIM OF LONGEST TEST
2254 001006 003410 $PASTM: .WORD 3410 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
2255 001010 003410 $UNITM: .WORD 3410 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
2256 001012 000030 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
2257 120210 AVECT1=120210 ;;RKVEC=210, RKPRI=5
2258 177440 ABASE=177440 ;;RKBAS ADRS
000377 ADEVM=000377 ;;SET DEVICES 0-7 IN MAP

```

Address	Value	Label	Description
2259		.SBTTL	COMMON TAGS
2260			
2261		*****	
2262		*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS	
2263		*USED IN THE PROGRAM.	
2264			
2265	001100		
2266	001100	\$CMTAG:	=1100
2267	001100		;; START OF COMMON TAGS
2268	001102	\$STNM:	.WORD 0
2269	001103	\$ERFLG:	.BYTE 00
2270	001104	\$ICNT:	.WORD 00
2271	001106	\$LPADR:	.WORD 00
2272	001110	\$LPERR:	.WORD 00
2273	001112	\$ERTTL:	.WORD 00
2274	001114	\$ITEMB:	.BYTE 00
2275	001115	\$ERMAX:	.BYTE 01
2276	001116	\$ERRPC:	.WORD 00
2277	001120	\$GDADR:	.WORD 00
2278	001122	\$BDADR:	.WORD 00
2279	001124	\$GDDAT:	.WORD 00
2280	001126	\$BDDAT:	.WORD 00
2281	001130		.WORD 00
2282	001132		.WORD 00
2283	001134	\$AUTOB:	.BYTE 00
2284	001135	\$INTAG:	.BYTE 00
2285	001136		.WORD 00
2286	001140	\$SWR:	.WORD DSWR
2287	001142	\$DISPLAY:	.WORD DDISP
2288	001144	\$TKS:	177560
2289	001146	\$TKB:	177562
2290	001150	\$TPS:	177564
2291	001152	\$TPB:	177566
2292	001154	\$NULL:	.BYTE 0
2293	001155	\$FILLS:	.BYTE 2
2294	001156	\$FILLC:	.BYTE 12
2295	001157	\$TPFLG:	.BYTE 0
2296	001160	\$REGAD:	.WORD 0
2297			
2298	001162	\$REG0:	.WORD 0
2299	001164	\$REG1:	.WORD 00
2300	001166	\$REG2:	.WORD 00
2301	001170	\$REG3:	.WORD 00
2302	001172	\$REG4:	.WORD 00
2303	001174	\$REG5:	.WORD 00
2304	001176	\$REG6:	.WORD 00
2305	001200	\$REG7:	.WORD 00
2306	001202	\$REG10:	.WORD 00
2307	001204	\$REG11:	.WORD 00
2308	001206	\$REG12:	.WORD 00
2309	001210	\$REG13:	.WORD 00
2310	001212	\$REG14:	.WORD 00
2311	001214	\$REG15:	.WORD 00
2312	001216	\$REG16:	.WORD 00
2313	001220	\$REG17:	.WORD 00
2314	001222	\$REG20:	.WORD 0

2315 001224 000000
 2316 001226 000000
 2317 001230 000000
 2318 001232 000000
 2319 001234 000000
 2320 001236 000000
 2321 001240 000000
 2322 001242 000000
 2323 001244 000000
 2324 001246 000000
 2325 001250 000000
 2326 001252 000000
 2327 001254 000000
 2328 001256 000000
 2329 001260 000000
 2330 001262 000000
 2331 001264 000000
 2332 001266 000000
 2333 001270 000000
 2334 001272 000000
 2335 001274 000000
 2336 001276 000000
 2337 001300 000000
 2338 001302 000000
 2339 001304 000000
 2340 001306 000000
 2341 001310 177607 000377
 2342 001314 077
 2343 001315 015
 2344 001316 000012
 2345
 2346
 2347
 2348
 2349
 2350 001320
 2351 001320 000000
 2352 001322 000000
 2353 001324 000000
 2354 001326 000000
 2355 001330 000000
 2356 001332 000000
 2357 001334 000000
 2358 001336 000000
 2359 001340
 2360 001340 000
 2361 001341 000
 2362 001342 000000
 2363 001344 000000
 2364 001346 000000
 2365
 2366
 2367
 2368
 2369
 2370

\$REG21: .WORD 000000 :: CONTAINS ((\$REGAD)+42)
 \$REG22: .WORD 000000 :: CONTAINS ((\$REGAD)+44)
 \$REG23: .WORD 000000 :: CONTAINS ((\$REGAD)+46)
 \$REG24: .WORD 000000 :: CONTAINS ((\$REGAD)+50)
 \$REG25: .WORD 000000 :: CONTAINS ((\$REGAD)+52)
 \$REG26: .WORD 000000 :: CONTAINS ((\$REGAD)+54)
 \$REG27: .WORD 000000 :: CONTAINS ((\$REGAD)+56)
 \$REG30: .WORD 000000 :: CONTAINS ((\$REGAD)+60)
 \$REG31: .WORD 000000 :: CONTAINS ((\$REGAD)+62)
 \$REG32: .WORD 000000 :: CONTAINS ((\$REGAD)+64)
 \$REG33: .WORD 000000 :: CONTAINS ((\$REGAD)+66)
 \$REG34: .WORD 000000 :: CONTAINS ((\$REGAD)+70)
 \$REG35: .WORD 000000 :: CONTAINS ((\$REGAD)+72)
 \$REG36: .WORD 000000 :: CONTAINS ((\$REGAD)+74)
 \$REG37: .WORD 000000 :: CONTAINS ((\$REGAD)+76)
 STMP0: .WORD 000000 :: USER DEFINED
 STMP1: .WORD 000000 :: USER DEFINED
 STMP2: .WORD 000000 :: USER DEFINED
 STMP3: .WORD 000000 :: USER DEFINED
 STMP4: .WORD 000000 :: USER DEFINED
 STMP5: .WORD 000000 :: USER DEFINED
 STMP6: .WORD 000000 :: USER DEFINED
 STMP7: .WORD 000000 :: USER DEFINED
 STMP10: .WORD 000000 :: USER DEFINED
 \$TIMES: 0 :: MAX. NUMBER OF ITERATIONS
 \$ESCAPE: 0 :: ESCAPE ON ERROR ADDRESS
 \$BELL: .ASCIZ <207><377><377> :: CODE FOR BELL
 \$QUES: .ASCII /?/ :: QUESTION MARK
 \$CRLF: .ASCII <15> :: CARRIAGE RETURN
 \$LF: .ASCIZ <12> :: LINE FEED
 ::*****
 .SBTTL APT MAILBOX-ETABLE
 ::*****
 .EVEN
 \$MAIL: :: APT MAILBOX
 \$MSGTY: .WORD AMSGTY :: MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL :: FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN :: TEST NUMBER
 \$PASS: .WORD APASS :: PASS COUNT
 \$DEVCT: .WORD ADEVCT :: DEVICE COUNT
 \$UNIT: .WORD AUNIT :: I/O UNIT NUMBER
 \$MSGAD: .WORD AMSGAD :: MESSAGE ADDRESS
 \$MSGLG: .WORD AMSGLG :: MESSAGE LENGTH
 \$ETABLE: :: APT ENVIRONMENT TABLE
 \$ENV: .BYTE AENV :: ENVIRONMENT BYTE
 \$ENVM: .BYTE AENVM :: ENVIRONMENT MODE BITS
 \$SWREG: .WORD ASWREG :: APT SWITCH REGISTER
 \$USWR: .WORD AUSWR :: USER SWITCHES
 \$CPUOP: .WORD ACPUOP :: CPU TYPE, OPTIONS
 ::*
 BIT 15-11=CPU TYPE
 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
 11/70=06, PDQ=07, Q=10
 BIT 10=REAL TIME CLOCK
 BIT 9=FLOATING POINT PROCESSOR
 BIT 8=MEMORY MANAGEMENT

J04

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 49
APT MAILBOX-ETABLE

SEQ 0048

2371	001350	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
2372	001351	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
2373			::*		MEM. TYPE BYTE -- (HIGH BYTE)
2374			::*		900 NSEC CORE=001
2375			::*		300 NSEC BIPOLAR=002
2376			::*		500 NSEC MOS=003
2377	001352	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
2378			::*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
2379	001354	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
2380	001355	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
2381	001356	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
2382	001360	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S. BYTE
2383	001361	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
2384	001362	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
2385	001364	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S. BYTE
2386	001365	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
2387	001366	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
2388	001370	120210	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
2389	001372	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
2390	001374	177440	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
2391	001376	000377	\$DEVN: .WORD	ADEVN	::DEVICE MAP
2392	001400		\$ETEND:		
2393			.MEXIT		

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

2394					
2395					
2396					
2397					
2398					
2399					
2400					
2401					
2402					
2403					
2404					
2405					
2406					
2407					
2408	001400				
2409					
2410					
2411	001400	056711			:ERROR 1 ;UNIBUS PARITY ERROR
2412	001402	062004			EM1
2413	001404	063570			DH100
2414	001406	063704			DT100
2415					DF01
2416					
2417	001410	056735			:ERROR 2 ;NON-EXISTANT MEMORY
2418	001412	062004			EM2
2419	001414	063570			DH100
2420	001416	063704			DT100
2421					DF01
2422					
2423	001420	056761			:ERROR 3 ;NON-EXISTANT DRIVE
2424	001422	062004			EM3
2425	001424	063570			DH100
2426	001426	063704			DT100
2427					DF01
2428					
2429	001430	057004			:ERROR 4 ;UNIT FIELD ERROR
2430	001432	062004			EM4
2431	001434	063570			DH100
2432	001436	063704			DT100
2433					DF01
2434					
2435	001440	057025			:ERROR 5 ;SUBSYSTEM TIMEOUT
2436	001442	062004			EM5
2437	001444	063570			DH100
2438	001446	063734			DT100
2439					DF02
2440					
2441	001450	057044			:ERROR 6 ;D TO C PARITY ERROR
2442	001452	062004			EM6
2443	001454	063570			DH100
2444	001456	063770			DT100
2445					DF03
2446					
2447	001460	057070			:ERROR 7 ;DRIVE DETECTED PARITY ERROR
2448	001462	062004			EM7
2449	001464	063570			DH100
					DT100

L04

2450	001466	063770	DF03	
2451				
2452			:ERROR 10	
2453	001470	057124	EM10	;AC LOW
2454	001472	062004	DH100	
2455	001474	063570	DT100	
2456	001476	063734	DF02	
2457				
2458			:ERROR 11	
2459	001500	057133	EM11	;SPEED LOSS
2460	001502	062004	DH100	
2461	001504	063570	DT100	
2462	001506	063734	DF02	
2463				
2464			:ERROR 12	
2465	001510	057146	EM12	;ILLEGAL FUNCTION
2466	001512	062004	DH100	
2467	001514	063570	DT100	
2468	001516	063734	DF02	
2469				
2470			:ERROR 13	
2471	001520	057167	EM13	;PROGRAMMING ERROR
2472	001522	062004	DH100	
2473	001524	063570	DT100	
2474	001526	063704	DF01	
2475				
2476			:ERROR 14	;NON-EXISTANT FUNCTION
2477	001530	057211	EM14	
2478	001532	062004	DH100	
2479	001534	063570	DT100	
2480	001536	063734	DF02	
2481				
2482			:ERROR 15	
2483	001540	057237	EM15	;DRIVE TYPE ERROR
2484	001542	062004	DH100	
2485	001544	063570	DT100	
2486	001546	063734	DF02	
2487				
2488			:ERROR 16	
2489	001550	057260	EM16	;FORMAT ERROR
2490	001552	062004	DH100	
2491	001554	063570	DT100	
2492	001556	063734	DF02	
2493				
2494			:ERROR 17	
2495	001560	057275	EM17	;WRITE LOCK ERROR
2496	001562	062004	DH100	
2497	001564	063570	DT100	
2498	001566	063734	DF02	
2499				
2500			:ERROR 20	
2501	001570	057316	EM20	;DRIVE UNSAFE
2502	001572	062004	DH100	
2503	001574	063570	DT100	
2504	001576	063734	DF02	
2505				

M04

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 52

SEQ 0051

ERROR POINTER TABLE

2506			:ERROR 21	
2507	001600	057333	EM21	;SEEK INCOMPLETE
2508	001602	062004	DH100	
2509	001604	063570	DT100	
2510	001606	063734	DF02	
2511			:ERROR 22	
2512			EM22	;CYLINDER OVERFLOW
2513	001610	057353	DH100	
2514	001612	062004	DT100	
2515	001614	063570	DF02	
2516	001616	063734		
2517			:ERROR 23	
2518			EM23	;ILLEGAL CYLINDER
2519	001620	057375	DH100	
2520	001622	062004	DT100	
2521	001624	063570	DF02	
2522	001626	063734		
2523			:ERROR 24	
2524			EM24	;DRIVE OFF TRACK
2525	001630	057426	DH100	
2526	001632	062004	DT100	
2527	001634	063570	DF02	
2528	001636	063734		
2529			:ERROR 25	
2530			EM25	;DRIVE TIMING ERROR
2531	001640	057446	DH100	
2532	001642	062004	DT100	
2533	001644	063570	DF02	
2534	001646	063734		
2535			:ERROR 26	
2536			EM26	;DATA LATE
2537	001650	057471	DH100	
2538	001652	062004	DT100	
2539	001654	063570	DF02	
2540	001656	063734		
2541			:ERROR 27	
2542			EM27	;CONTROLLER TIMEOUT
2543	001660	057503	DH100	
2544	001662	062004	DT100	
2545	001664	063570	DF02	
2546	001666	063734		
2547			:ERROR 30	
2548			EM30	;OPERATION INCOMPLETE
2549	001670	057526	DH100	
2550	001672	062004	DT100	
2551	001674	063570	DF05	
2552	001676	064044		
2553			:ERROR 31	
2554			EM31	;HEADER VRC ERROR
2555	001700	057553	DH100	
2556	001702	062004	DT100	
2557	001704	063570	DF05	
2558	001706	064044		
2559			:ERROR 32	
2560			EM32	;DATA CHECK ERROR
2561	001710	057574		

N04

MD-11-DZR&N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR&NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 53

SEQ 0052

ERROR POINTER TABLE

2562	001712	062004	DH100	
2563	001714	063570	DT100	
2564	001716	064100	DF07	
2565				
2566			:ERROR 33	
2567	001720	057615	EM33	;WRITE CHECK ERROR
2568	001722	062004	DH100	
2569	001724	063570	DT100	
2570	001726	064134	DF10	
2571				
2572			:ERROR 34	
2573	001730	057637	EM34	;DATA MISCOMPARE(S)
2574	001732	062004	DH100	
2575	001734	063570	DT100	
2576	001736	064030	DF04	
2577				
2578			:ERROR 35	
2579	001740	057657	EM35	;NO DRIVE RESPONSE-UFE & NXD
2580	001742	062004	DH100	
2581	001744	063570	DT100	
2582	001746	063704	DF01	
2583				
2584			:ERROR 36	
2585	001750	057713	EM36	;DRIVE ERROR WILL NOT CLEAR
2586	001752	000000	0	
2587	001754	000000	0	
2588	001756	000000	0	
2589				
2590			:ERROR 37	
2591	001760	057746	EM37	;DRIVE STATUS CHANGE WILL NOT CLEAR
2592	001762	000000	0	
2593	001764	000000	0	
2594	001766	000000	0	
2595				
2596			:ERROR 40	
2597	001770	060011	EM40	;ATTENTION BUT NO STATUS CHANGE OR FAULT
2598	001772	062004	DH100	
2599	001774	063570	DT100	
2600	001776	063734	DF02	
2601				
2602			:ERROR 41	
2603	002000	060055	EM41	;ATTENTION BUT DRIVE NOT AVAILABLE
2604	002002	062004	DH100	
2605	002004	063570	DT100	
2606	002006	063734	DF02	
2607				
2608			:ERROR 42	
2609	002010	060113	EM42	;ATTENTION WHEN NOT EXPECTED
2610	002012	062004	DH100	
2611	002014	063570	DT100	
2612	002016	063734	DF02	
2613				
2614			:ERROR 43	
2615	002020	060143	EM43	;ERROR WHILE GATHERING DRIVE STATUS
2616	002022	062004	DH100	
2617	002024	063570	DT100	

2618	002026	064200	DF12	
2619			.ERROR 44	
2620	002030	060420	EM63	:CLEAR CONTROLLER DID NOT CLEAR ERROR
2621	002032	062004	DH100	
2622	002034	063570	DT100	
2623	002036	064200	DF12	
2624			.ERROR 45	
2625	002040	060465	EM64	:NO ATTENTION IN ATTENTION SUMMARY REG
2626	002042	062004	DH100	
2627	002044	063570	DT100	
2628	002046	064200	DF12	
2629			.ERROR 46	
2630	002050	060530	EM65	:UNSOLICITED ATTENTION
2631	002052	062004	DH100	
2632	002054	063570	DT100	
2633	002056	064200	DF12	
2634			.ERROR 47	
2635	002060	060556	EM66	:UNEXPECTED DATA TYPE ERROR
2636	002062	062004	DH100	
2637	002064	063570	DT100	
2638	002066	064200	DF12	
2639			.ERROR 50	
2640	002070	060611	EM67	:ATTENTION DID NOT RESET WITH CLEAR
2641	002072	062004	DH100	
2642	002074	063570	DT100	
2643	002076	064200	DF12	
2644			.ERROR 51	
2645	002100	060650	EM70	:SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION
2646	002102	062004	DH100	
2647	002104	063570	DT100	
2648	002106	064200	DF12	
2649			.ERROR 52	
2650	002110	060200	EM52	:MULTIPLE DRIVE SELECT
2651	002112	062004	DH100	
2652	002114	063570	DT100	
2653	002116	064200	DF12	
2654			.ERROR 53	
2655	002120	060226	EM53	:ABREVIATED HCE ERROR
2656	002122	062004	DH100	
2657	002124	063570	DT100	
2658	002126	064230	DF13	
2659			.ERROR 54	
2660	002130	057526	EM30	:OPERATION INCOMPLETE ERROR
2661	002132	062004	DH100	
2662	002134	063570	DT100	
2663	002136	064260	DF14	

ERROR POINTER TABLE

2674			:ERROR 55	
2675	002140	057553	EM31	:ABREVIATED HVRC ERROR
2676	002142	062004	DH100	
2677	002144	063570	DT100	
2678	002146	064230	DF13	
2680			:ERROR 56	
2681	002150	060253	EM56	:2 TIMEOUT ERROR
2682	002152	062004	DH100	
2683	002154	063570	DT100	
2684	002156	064310	DF15	
2687			:ERROR 57	
2688	002160	060253	EM56	:2ND LEVEL IN SUBSYSTEM TIMEOUT
2689	002162	062004	DH100	
2690	002164	063570	DT100	
2691	002166	064354	DF16	
2693			:ERROR 60	
2694	002170	060272	EM60	:ERROR IN RECAL FOR RECOVERY
2695	002172	000000	0	
2696	002174	000000	0	
2697	002176	000000	0	
2698			:ERROR 61	
2699	002200	060326	EM61	:ABORT MESSAGE
2700	002202	000000	0	
2701	002204	000000	0	
2702	002206	000000	0	
2703			:ERROR 62	
2704			EM62	:CYLINDER MISCOMPARE
2705	002210	060374	DH100	
2706	002212	062004	DT100	
2707	002214	063570	DF17	
2708	002216	064420		
2709			:ERROR 63	
2710			0	:DATA ERROR WORDS
2711	002220	000000	0	
2712	002222	000000	DT602	
2713	002224	063662	DF25	
2714	002226	064530		
2715			:ERROR 64	
2716	002230	060420	EM63	:CLEAR CONTROLLER DID NOT CLEAR ERROR
2717	002232	062004	DH100	
2718	002234	063570	DT100	
2719	002236	063734	DF02	
2720			:ERROR 65	
2721			EM64	:NO ATTENTION IN ATTENTION SUMMARY REG
2722	002240	060465	DH100	
2723	002242	062004	DT100	
2724	002244	063570	DF02	
2725	002246	063734		
2726			:ERROR 66	
2727			EM65	:UNSOLICITED ATTENTION
2728				
2729	002250	060530		

ERROR POINTER TABLE

2730	002252	062004	DH100	
2731	002254	063570	DT100	
2732	002256	063734	DF02	
2733				
2734				
2735	002260	060556	:ERROR 67	
2736			EM66	:UNEXPECTED DATA TYPE ERROR
2737	002262	062004	DH100	
2738	002264	063570	DT100	
2739	002266	063734	DF02	
2740				
2741	002270	060611	:ERROR 70	
2742			EM67	:ATTENTION DID NOT RESET WITH CLEAR
2743	002272	062004	DH100	
2744	002274	063570	DT100	
2745	002276	063734	DF02	
2746				
2747	002300	060650	:ERROR 71	
2748			EM70	:SUBSYSTEM CLEAR DID NOT CLEAR ATT
2749	002302	062004	DH100	
2750	002304	063570	DT100	
2751	002306	063734	DF02	
2752				
2753	002310	060717	:ERROR 72	
2754			EM71	:DATA LATE WHEN UNLOADING HEADER
2755	002312	062004	DH100	
2756	002314	063570	DT100	
2757	002316	063734	DF02	
2758				
2759	002320	060757	:ERROR 73	
2760			EM72	:CONTROLLER ERROR DURING DRIVER SERVICE
2761	002322	062004	DH100	
2762	002324	063570	DT100	
2763	002326	063734	DF02	
2764				
2765	002330	061026	:ERROR 74	
2766			EM73	:DRIVE DETECTED PARITY ERROR
2767	002332	062004	DH100	
2768	002334	063570	DT100	
2769	002336	063734	DF02	
2770				
2771	002340	061062	:ERROR 75	
2772			EM74	:UNDEFINED ERROR
2773	002342	062004	DH100	
2774	002344	063570	DT100	
2775	002346	063734	DF02	
2776				
2777	002350	061102	:ERROR 76	
2778			EM75	:MARKING SECTOR BAD MESSAGE
2779	002352	000000	0	
2780	002354	000000	0	
2781	002356	000000	0	
2782				
2783	002360	061132	:ERROR 77	
2784			EM76	:BAD DATA VERIFICATION WITH READ
2785	002362	062771	DH605	
2786	002364	063654	DT601	

ERROR POINTER TABLE

2786	002366	064454	DF21	
2787			:ERROR 100	
2788			EM77	:RETRY SUCCESSFUL MESSAGE
2789	002370	061214	0	
2790	002372	000000	DT601	
2791	002374	063654	DF23	
2792	002376	064514		
2793			:ERROR 101	
2794			EM77	:ANOTHER RETRY SUCCESSFUL MESSAGE
2795	002400	061214	0	
2796	002402	000000	DT601	
2797	002404	063654	DF23	
2798	002406	064514		
2799			:ERROR 102	
2800			EM100	:RETRY UNSUCCESSFUL MESSAGE
2801	002410	061235	0	
2802	002412	000000	DT601	
2803	002414	063654	DF23	
2804	002416	064514		
2805			:ERROR 103	
2806			EM101	:NO VALID HEADERS IN TRACK JUST READ
2807	002420	061260	DH6042	
2808	002422	062753	DT601	
2809	002424	063654	DF24	
2810	002426	064524		
2811			:ERROR 104	
2812			EM102	:BSE ERROR ON SECTOR NOT LISTED AS BAD
2813	002430	061336	DH100	
2814	002432	062004	DT100	
2815	002434	063570	DF02	
2816	002436	063734		
2817			:ERROR 105	
2818			EM103	:TIMED-OUT ON READ HEADER
2819	002440	061410	DH100	
2820	002442	062004	DT100	
2821	002444	063570	DF03	
2822	002446	063770		
2823			:ERROR 106	
2824			EM104	:TIMED-OUT ON SEEK
2825	002450	061436	DH100	
2826	002452	062004	DT100	
2827	002454	063570	DF03	
2828	002456	063770		
2829			:ERROR 107	
2830			EM105	:DRIVE SIEZED BY OTHER PORT
2831	002460	061460	DH100	
2832	002462	062004	DT100	
2833	002464	063570	DF02	
2834	002466	063734		
2835			:ERROR 110	
2836			EM106	: "DATA MISCMPR WHILE BAI SET"
2837	002470	061513	DH100	
2838	002472	062004	DT100	
2839	002474	063570	DF27	
2840	002476	064544		
2841				

2842			:ERROR 111	
2843	002500	061546	EM107	;"NO NEM WHEN EXPECTED"
2844	002502	000000	0	
2845	002504	000000	0	
2846	002506	000000	0	
2847				
2848			:ERROR 112	
2849	002510	061613	EM110	;"INTRPT WHEN CNTRLR NOT READY"
2850	002512	062004	DH100	
2851	002514	063570	DT100	
2852	002516	063704	DF01	
2853				
2854			:ERROR 113	
2855	002520	061646	EM111	;"NO ATT'N ON SEEK"
2856	002522	062004	DH100	
2857	002524	063570	DT100	
2858	002526	063734	DF02	
2859				
2860			:ERROR 114	
2861	002530	061667	EM112	;DRIVE'S CYLINDER INCORRECT
2862	002532	063200	DH702	
2863	002534	063630	DT202	
2864	002536	064534	DF26	
2865				
2866			:ERROR 115	
2867	002540	000000	0	;TYPE ADRS OF DATA MISCOMPARE(S)
2868	002542	000000	0	
2869	002544	063654	DT601	
2870	002546	064160	DF11	
2871				
2872			:ERROR 116	
2873	002550	057637	EM34	;DATA MISCOMPARE (11/70)
2874	002552	062315	DH103	
2875	002554	063700	DT103	
2876	002556	064444	DF20	
2877				
2878			:ERROR 117	
2879	002560	000000	0	;PART OF DATA MISCOMPARE
2880	002562	000000	0	
2881	002564	063570	DT100	
2882	002566	064464	DF22	
2883				
2884			:ERROR 120	
2885	002570	061722	EM113	;ABORT- CAN'T READ BSF
2886	002572	062004	DH100	
2887	002574	063570	DT100	
2888	002576	063704	DF01	
2889				
2890			:ERROR 121	
2891	002600	061750	EM114	;KT11 FAILURE
2892	002602	062004	DH100	
2893	002604	063570	DT100	
2894	002606	064570	DF30	
2895				
2896			:ERROR 122	
2897	002610	061765	EM115	;MEM PARITY ERROR

G05

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 59

SEQ 0058

ERROR POINTER TABLE

2898	002612	062004	DH100
2899	002614	063570	DT100
2900	002616	064614	DF31
2901			
2902			
2903			
2904			
2905			
2906			
2907			
2908			
2909			
2910			

H05

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 60
BIT ASSIGNMENTS IN THE RK611 REGISTERS

SEQ 0059

.SBTTL BIT ASSIGNMENTS IN THE RK611 REGISTERS

RKCS1							
15	14	13	12	11	10	9	8
CERR	DI	DCPAR	CFMT	CTO	CDT	BA17	BA16
R/W	RO	RO	R/W	RO	R/W	R/W	R/W

7	6	5	4	3	2	1	0
RDY	IE	SPARE	F4	F3	F2	F1	GO
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

RKCS2							
15	14	13	12	11	10	9	8
DLT	WCE	UPE	NED	NEM	PGE	MDS	UFE
RO	RO	RO	RO	RO	RO	RO	RO

7	6	5	4	3	2	1	0
OR	IR	CLR	BAI	DESL	DS2	DS1	DS0
RO	RO	WO	R/W	R/W	R/W	R/W	R/W

RKDC (DESIRED CYLINDER)							
15	14	13	12	11	10	9	8
NU	NU	NU	NU	NU	NU	DC9	DC8
READ ONLY							

7	6	5	4	3	2	1	0
DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
READ ONLY							

RKDA (DESIRED TRACK AND SECTOR)							
15	14	13	12	11	10	9	8
NU	UN	UN	UN	UN	TA2	TA1	TA0
READ ONLY							

7	6	5	4	3	2	1	0
UN	UN	UN	SA4	SA3	SA2	SA1	SA0
READ ONLY							

0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100

2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011

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

J05

3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050

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

K05

3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088

:RKECC/PAT							
15	14	13	12	11	10	9	8
UN	!	UN	!	UN	!	UN	!
EPA10! EPA9! EPA8!							
READ ONLY							

7	6	5	4	3	2	1	0
EPA7! EPA6! EPA5! EPA4! EPA3! EPA2! EPA1! EPA0!							
READ ONLY							

:RKECC/POS							
15	14	13	12	11	10	9	8
UN	!	UN	!	UN	!	UN	!
EP012! EP011! EP010! EP09! EP08!							
READ ONLY							

7	6	5	4	3	2	1	0
EP07! EP06! EP05! EP04! EP03! EP02! EP01! EP00!							
READ ONLY							

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

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

L05

3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126

:LINE B MESSAGE 00							
15	14	13	12	11	10	9	8
PAR	UNS	DROT	DCLO	WLE	SKI	PERR	ILF
7	6	5	4	3	2	1	0
FLT	ACLO	IVDA	UN	UN	UN	0	0
:LINE A MESSAGE 01							
15	14	13	12	11	10	9	8
PAR	UNLDG	RTZ	LDG	REV	FWD	SPEED	CART
	HDS		HDS			OK	PRES
7	6	5	4	3	2	1	0
DOOR	BRUSH	HEADS	TRK	UN	DRIVE SELECT CODE		
LTCH	HOME	HOME	FOLOW				
			OK				
:LINE B MESSAGE 01							
15	14	13	12	11	10	9	8
PAR	SERVO	LIMIT	SEEK	PLO	DIBIT	INDEX	MULTI
	BRAKE	ON SK	NO MO	ERR	ERR	ERR	HD SEL
7	6	5	4	3	2	1	0
HEAD	WR GTE	WR CUR	SEC	NU	NU	0	1
FAULT	AND NO	AND NO	ERR				
	XISTON	WR GTE					

M05

0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182

```

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

.SBTTL RK06 CONTROLLER REGISTER DEFINITION

000000
000002
000004
000006
000010

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

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

```

3183      000012      RKDS= 12      ;DRIVE STATUS REGISTER
3184      000014      RKER= 14      ;ERROR REGISTER
3185      000016      RKASOF= 16     ;ATTENTION SUMMARY AND OFFSET REGISTER
3186      000020      RKDC= 20     ;DESIRED CYLINDER REGISTER
3187      000020      RKDCYL= 20   ;DESIRED CYLINDER REGISTER
3188      000024      RKDB= 24     ;DATA BUFFER
3189      000026      RKMR1= 26   ;MAINTENANCE REGISTER 1
3190      000034      RKMR2= 34   ;MAINTENANCE REGISTER 2
3191      000036      RKMR3= 36   ;MAINTENANCE REGISTER 3
3192      000030      RKPOS= 30   ;ECC POSITION INFORMATION
3193      000030      RKECPS= 30  ;ECC POSITION INFORMATION
3194      000032      RKPAT= 32   ;ECC PATTERN INFORMATION
3195      000032      RKECPT= 32  ;ECC PATTERN INFORMATION
3196
3197      .SBTTL DRIVE COMMANDS
3198
3199      000101      SELDRV= 101  ;SELECT DRIVE
3200      000103      PACK= 103   ;PACK ACKNOWLEDGE
3201      000105      CLEAR= 105  ;DRIVE CLEAR
3202      000107      UNLOAD= 107 ;UNLOAD
3203      000111      SRTSPL= 111 ;START SPINDLE
3204      000113      RECAL= 113  ;RECALIBRATE
3205      000115      OFFSET= 115 ;OFFSET
3206      000117      SEEK= 117   ;SEEK
3207      000121      RDDATA= 121 ;READ DATA
3208      000123      WRDATA= 123 ;WRITE DATA
3209      000125      RDHEAD= 125 ;READ HEADER
3210      000127      WRHEAD= 127 ;WRITE HEADER AND DATA
3211      000131      WRTCHK= 131 ;WRITE CHECK
3212
3213      ; THE FOLLOWING ARE NOT DRIVE COMMANDS BUT ARE USED BY THE DRIVER
3214      ; TO SIMULATE A SPECIFIC DESIRED OPERATION
3215
3216      000140      RELEAS= 140 ;RELEASE DRIVE
3217      000141      RDSTAT= 141 ;GET ALL STATUS FROM DRIVE
3218      000164      RDALHD= 164 ;READ ALL HEADERS
3219      000176      CONCLR= 176 ;CONTROLLER CLEAR (BIT 15 OF CS1)
3220      000177      SUBCLR= 177 ;SUBSYSTEM CLEAR (BIT 5 OF CS2)
3221      000300      INTR= 300  ;GENERATE INTERRUPT TO CPU
3222
3223      ; DRIVER ISSUED SERVICE COMMANDS
3224
3225      000001      DR.SEL= 001 ;DRIVE SELECT
3226      000005      DR.CLR= 005 ;DRIVE CLEAR
3227
3228      .SBTTL CONTROL AND STATUS REGISTER 1 BITS
3229
3230      000001      GO= BIT0    ;GO BIT
3231      000100      IE= BIT6    ;INTERRUPT ENABLE
3232      000200      RDY= BIT7    ;CONTROLLER READY
3233      000400      BA16= BIT8    ;BUS ADDRESS BIT 16
3234      001000      BA17= BIT9    ;BUS ADDRESS BIT 17
3235      002000      CDT= BIT10   ;CONTROLLER DRIVE TYPE (0=RK06)
3236      004000      CTO= BIT11   ;CONTROLLER TIMED OUT WAITING FOR
3237      ; DRIVE RESPONSE
3238      010000      CFMT= BIT12  ;CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)
    
```

DZRBNC.F11 05-OCT-76 10:07

CONTROL AND STATUS REGISTER 1 BITS

020000
040000
100000
100000

SPAR= BIT13 :DRIVE BUS PARITY ERROR DETECTED BY CONTROLLER
DI= BIT14 :DRIVE INTERRUPT
CERR= BIT15 :CONTROLLER ERROR
CCLR= BIT15 :CONTROLLER CLEAR

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

000001
000002
000004
000020

B.BA16= BIT0 :BUS ADDRESS BIT 16
B.BA17= BIT1 :BUS ADDRESS BIT 17
B.CDT= BIT2 :CONTROLLER DRIVE TYPE (0=RK06)
B.CFMT= BIT4 :CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)

.SBTTL CONTROL AND STATUS REGISTER 2 BITS

000007
000010
000010
000020
000040
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000

DRVMSK= 7 :MASK FOR DRIVE SELECTION CODE
DESL= BIT3 :DESELECT OR RELEASE DRIVE IN BITS 0-2
RLS= BIT3 :DESELECT OR RELEASE DRIVE IN BITS 0-2
BAI= BIT4 :BUS ADDRESS INCREMENT INHIBIT
CLR= BIT5 :CLEAR CONTROLLER AND ALL DRIVES
SCLR= BIT5 :CLEAR CONTROLLER AND ALL DRIVES
IR= BIT6 :INPUT READY
OR= BIT7 :OUTPUT READY
UFE= BIT8 :UNIT FIELD ERROR
MDS= BIT9 :MULTIPLE DRIVE SELECT
PGE= BIT10 :PROGRAMMING ERROR
NEM= BIT11 :NON-EXISTENT MEMORY
NED= BIT12 :NON-EXISTENT DRIVE
UPE= BIT13 :UNIBUS PARITY ERROR
WCE= BIT14 :WRITE CHECK ERROR
DLT= BIT15 :DATA LATE ERROR

.SBTTL ERROR REGISTER BIT DEFINITION

000001
000002
000004
000004
000010
000020
000040
000100
000200
000400
000400
001000
002000
004000
010000
020000
040000
100000

ILC= BIT0 :ILLEGAL FUNCTION CODE
: *ILF= BIT0 :ILLEGAL FUNCTION CODE
SKI= BIT1 :SEEK INCOMPLETE
ILF= BIT2 :ILLEGAL DRIVE FUNCTION
NXF= BIT2 :ILLEGAL DRIVE FUNCTION
DRPAR= BIT3 :DRIVE DETECTED DRIVE BUS PARITY ERROR
FMTE= BIT4 :FORMAT ERROR
DTYE= BIT5 :DRIVE TYPE ERROR
ECH= BIT6 :ECC HARD
BSE= BIT7 :BAD SECTOR ERROR
HCRC= BIT8 :HEADER CRC ERROR
HVRC= BIT8 :HEADER VRC ERROR
COE= BIT9 :CYLINDER ADDRESS OVERFLOW ERROR
IDAE= BIT10 :INVALID DISK ADDRESS ERROR
WLE= BIT11 :WRITE LOCK ERROR
DTE= BIT12 :DRIVE TIMING ERROR
OPI= BIT13 :OPERATION (SEARCH) INCOMPLETE
UNS= BIT14 :DRIVE UNSAFE
DCK= BIT15 :DATA CHECK

.SBTTL STATUS REGISTER BIT DEFINITION

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099
000100

STATUS REGISTER BIT DEFINITION

000001	DRA=	BIT0	: DRIVE AVAILABLE (CONTROLLER IS SET IF THIS BIT IS RESET)
000004	OFST=	BIT2	: DRIVE OFFSET
000010	ACLO=	BIT3	: AC LOW
000020	SPDLSS=	BIT4	: SPEED LOSS
000020	DCLO=	BIT4	: DC LOW
000040	DROT=	BIT5	: DRIVE OFF TRACK
000100	VV=	BIT6	: VOLUME VALID
000200	DRY=	BIT7	: DRIVE READY
000200	DRDY=	BIT7	: DRIVE READY
000400	DDT=	BIT8	: DRIVE TYPE (0=RK06)
004000	WRL=	BIT11	: WRITE LOCK
020000	PIP=	BIT13	: POSITIONING IN PROGRESS
040000	DSC=	BIT14	: DRIVE STATUS CHANGE
100000	SVAL=	BIT15	: STATUS VALID

.SBTTL MAINTENANCE REGISTER 1 BIT DEFINITION

000017	MESMSK=	17	: MESSAGE MASK
000020	PAT=	BIT4	: FORCE EVEN PARITY ON DRIVE BUS MESSAGE LINES
000040	DMD=	BIT5	: DIAGNOSTIC MODE
000100	MSP=	BIT6	: MAINTENANCE SECTOR PULSE
000200	MIND=	BIT7	: MAINTENANCE INDEX
000400	MCLK=	BIT8	: MAINTENANCE CLOCK
001000	MERD=	BIT9	: MAINTENANCE ENCODED READ DATA
002000	MEWD=	BIT10	: MAINTENANCE ENCODED WRITE DATA
004000	PCA=	BIT11	: PRECOMPENSATION ADVANCE
010000	PCD=	BIT12	: PRECOMPENSATION DELAY
020000	ECCW=	BIT13	: ECC WORD IS BEING READ OR WRITTEN
040000	WRTGAT=	BIT14	: WRITE GATE
100000	RDGATE=	BIT15	: READ GATE

.SBTTL DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE A

000040	S.DRA=	BIT5	: DRIVE AVAILIABLE
000100	S.VV=	BIT6	: VOLUME VALID
000200	S.DRY=	BIT7	: DRIVE READY
000400	S.TYPE=	BIT8	: DRIVE TYPE
001000	S.FORM=	BIT9	: DRIVE FORMAT
002000	S.OFF=	BIT10	: OFFSET
004000	S.WRL=	BIT11	: WRITE LOCK
010000	S.SPIN=	BIT12	: SPINDLE ON
020000	S.PIP=	BIT13	: POSITIONING IN PROGRESS
040000	S.DSC=	BIT14	: DRIVE STATUS CHANGE

.SBTTL DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE B

000040	S.ICYL=	BIT5	: ILLEGAL CYLINDER ADDRESS
000100	S.ACLO=	BIT6	: AC LOW
000200	S.FLT=	BIT7	: DRIVE FAULT
000400	S.ILF=	BIT8	: ILLEGAL FUNCTION
001000	S.PAR=	BIT9	: DRIVE DETECTED DRIVE BUS PARITY ERROR
002000	S.SKI=	BIT10	: SEEK INCOMPLETE
004000	S.WLE=	BIT11	: WRITE LOCK ERROR
010000	S.SPLS=	BIT12	: SPEED LOSS

DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE B

010000
020000
040000

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

.SBTTL DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE A

000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000

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

.SBTTL DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE B

000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000

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

.SBTTL COMMON MASKS

000007
100000
000003
017760
017760
077770
000760
007000

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

.SBTTL PARAMETER BLOCK ALLOCATION

```

*****
: 1  : COMMAND          : DRIVE NO.
: 2  : CYLINDER ADDRESS
: 3  : TRACK          : SECTOR
: 4  : BA16-17,FORMAT,DRV TYPE: OFFSET
: 5  : BUS ADDRESS (LOW 16 BITS)
: 6  : WORD COUNT (2'S COMPLEMENT)
: 7  : PROGRAM DRIVE STATUS INFORMATION
: 8  : COMMAND AND STATUS REGISTER 1
: 9  : COMMAND AND STATUS REGISTER 2
:10  : WORD COUNT REGISTER
:11  : BUS ADDRESS REGISTER
:12  : DESIRED TRACK AND SECTOR
:13  : DESIRED CYLINDER
:14  : ATTENTION SUMMARY AND DRIVE OFFSET
:15  : ERROR REGISTER
:16  : STATUS REGISTER
:17  : MESSAGE LINE A STATUS BYTE 00
:18  : MESSAGE LINE B STATUS BYTE 00
:19  : MESSAGE LINE A STATUS BYTE 01
:20  : MESSAGE LINE B STATUS BYTE 01
:21  : MESSAGE LINE A STATUS BYTE 10
:22  : MESSAGE LINE B STATUS BYTE 10
:23  : MESSAGE LINE A STATUS BYTE 11
:24  : MESSAGE LINE B STATUS BYTE 11
:25  : ECC POSITION INFORMATION
:26  : ECC PATTERN INFORMATION
*****

```

.SBTTL PARAMETERS PASSED TO THE DRIVER

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

```

000000 P.DRVN= 0 ;DRIVE NUMBER
000001 P.CMND= 1 ;COMMAND
000002 P.CYLN= 2 ;CYLINDER ADDRESS
000004 P.SECT= 4 ;SECTOR
000005 P.TRCK= 5 ;TRACK
000006 P.OFST= 6 ;OFFSET
000007 P.CS1H= 7 ;RKCS1 BITS 8-15
000007 P.BA16= 7 ;BUS ADDRESS (BITS 16 AND 17)
000010 P.BA10= 10 ;BUS ADDRESS (BITS 0-15)
000012 P.WC= 12 ;WORD COUNT (2'S COMPLEMENT)
000014 P.PRST= 14 ;PROGRAM DRIVE STATUS INFORMATION

```

.SBTTL PROGRAM DEVICE STATUS REGISTER DEFINITION

```

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

```

000000
000001
000002
000004
000005
000006
000007
000007
000010
000012
000014
000001
000002
000004
000010
000020
000040

3449	000100	CMDT0= BIT6	: NO TERMINATION TO COMMAND FOR AT
3450			: LEAST 1 SECOND
3451	000200	W.WCK= BIT7	: WRITE FOR WRITE WRITE CHECK
3452	000400	NOCHK= BIT8	: NO CHECK, DO NOT SET INTERRUPT ENABLE
3453	001000	PBSVAL= BIT9	: PARAMETER STATUS WORDS VALID
3454			: (SET WHEN ERROR TERMINATION OR
3455			: READ STATUS COMMAND)
3456	002000	DRPDRV= BIT10	: DROP DRIVE FROM TEST SEQUENCE
3457	004000	NODSC= BIT11	: ATTENTION SET BUT DCS AND FAULT RESET
3458	010000	DRVSZD= BIT12	: DRIVE SEIZED BY OTHER PORT
3459	020000	E.UNLD= BIT13	: DRIVE UNLOADED DUE TO ERROR
3460	040000	Q.INIT= BIT14	: PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
3461	100000	DTBAII= BIT15	: INHIBIT BUS ADDRESS INCREMENT

.SBTTL PARAMETERS PASSED FROM DRIVER TO PROGRAM

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

3462	000016	P.CS1= 16	: COMMAND AND STATUS REGISTER 1
3463	000020	P.CS2= 20	: COMMAND AND STATUS REGISTER 2
3464	000022	P.WCR= 22	: WORD COUNT REGISTER
3465	000024	P.BAR= 24	: BUS ADDRESS REGISTER
3466	000026	P.DTS= 26	: DESIRED TRACK SECTOR REGISTER
3467	000030	P.DCYL= 30	: DESIRED CYLINDER REGISTER
3468	000032	P.ASOF= 32	: ATTENTION SUMMARY/OFFSET REGISTER
3469	000034	P.ER= 34	: ERROR REGISTER
3470	000036	P.DS= 36	: STATUS REGISTER
3471	000040	P.A00= 40	: MESSAGE A STATUS BYTE 00
3472	000042	P.B00= 42	: MESSAGE B STATUS BYTE 00
3473	000044	P.A01= 44	: MESSAGE A STATUS BYTE 01
3474	000046	P.B01= 46	: MESSAGE B STATUS BYTE 01
3475	000050	P.A10= 50	: MESSAGE A STATUS BYTE 10
3476	000052	P.B10= 52	: MESSAGE B STATUS BYTE 10
3477	000054	P.A11= 54	: MESSAGE A STATUS BYTE 11
3478	000056	P.B11= 56	: MESSAGE B STATUS BYTE 11
3479	000060	P.EPOS= 60	: ECC POSITION INFORMATION
3480	000062	P.EPAT= 62	: ECC PATTERN INFORMATION

.SBTTL PARAMETER BLOCK 0 FOR DRIVE

3481	002620	000	PARMO: .BYTE 0	: DRIVE NUMBER
3482	002621	000	.BYTE 0	: COMMAND
3483	002622	000000	.WORD 00	: CYLINDER ADDRESS
3484	002624	000	.BYTE 00	: SECTOR ADDRESS
3485	002625	000	.BYTE 00	: TRACK ADDRESS
3486	002626	000	.BYTE 00	: OFFSET VALUE
3487	002627	000	.BYTE 00	: BUS ADDRESS (BITS 16 AND 17)
3488	002630	000000	.WORD 00	: BUS ADDRESS (BITS 0 - 15)
3489	002632	000000	.WORD 00	: WORD COUNT (2'S COMPLEMENT)
3490	002634	000000	.WORD 00	: PROGRAM DRIVE STATUS INFORMATION
3500	002636	000000	.WORD 00	: COMMAND AND STATUS REGISTER 1
3501	002640	000000	.WORD 00	: COMMAND AND STATUS REGISTER 2
3502	002642	000000	.WORD 00	: WORD COUNT REGISTER
3503	002644	000000	.WORD 00	: BUS ADDRESS REGISTER
3504	002646	000000	.WORD 00	: DESIRED TRACK AND SECTOR REGISTER

3505	002650	000000	.WORD	0	: DESIRED CYLINDER REGISTER
3506	002652	000000	.WORD	0	: ATTENTION SUMMARY/OFFSET REGISTER
3507	002654	000000	.WORD	0	: ERROR REGISTER
3508	002656	000000	.WORD	0	: STATUS REGISTER
3509	002660	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 00
3510	002662	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 00
3511	002664	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 01
3512	002666	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 01
3513	002670	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 10
3514	002672	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 10
3515	002674	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 11
3516	002676	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 11
3517	002700	000000	.WORD	0	: ECC POSITION INFORMATION
3518	002702	000000	.WORD	0	: ECC PATTERN INFORMATION

.SBTTL PARAMETER BLOCK 1 FOR DRIVE

3520					
3521					
3522	002704	000	PARM1: .BYTE	0	: DRIVE NUMBER
3523	002705	000	.BYTE	0	: COMMAND
3524	002706	000000	.WORD	0	: CYLINDER ADDRESS
3525	002710	000	.BYTE	0	: SECTOR ADDRESS
3526	002711	000	.BYTE	0	: TRACK ADDRESS
3527	002712	000	.BYTE	0	: OFFSET VALUE
3528	002713	000	.BYTE	0	: BUS ADDRESS (BITS 16 AND 17)
3529	002714	000000	.WORD	0	: BUS ADDRESS (BITS 0 - 15)
3530	002716	000000	.WORD	0	: WORD COUNT (2'S COMPLEMENT)
3531	002720	000000	.WORD	0	: PROGRAM DRIVE STATUS INFORMATION
3532	002722	000000	.WORD	0	: COMMAND AND STATUS REGISTER 1
3533	002724	000000	.WORD	0	: COMMAND AND STATUS REGISTER 2
3534	002726	000000	.WORD	0	: WORD COUNT REGISTER
3535	002730	000000	.WORD	0	: BUS ADDRESS REGISTER
3536	002732	000000	.WORD	0	: DESIRED TRACK AND SECTOR REGISTER
3537	002734	000000	.WORD	0	: DESIRED CYLINDER REGISTER
3538	002736	000000	.WORD	0	: ATTENTION SUMMARY/OFFSET REGISTER
3539	002740	000000	.WORD	0	: ERROR REGISTER
3540	002742	000000	.WORD	0	: STATUS REGISTER
3541	002744	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 00
3542	002746	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 00
3543	002750	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 01
3544	002752	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 01
3545	002754	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 10
3546	002756	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 10
3547	002760	000000	.WORD	0	: MESSAGE LINE A STATUS BYTE 11
3548	002762	000000	.WORD	0	: MESSAGE LINE B STATUS BYTE 11
3549	002764	000000	.WORD	0	: ECC POSITION INFORMATION
3550	002766	000000	.WORD	0	: ECC PATTERN INFORMATION

.SBTTL TEMPORARY CONTROLLER REGISTER STORAGE

3551					
3552					
3553					
3554	002770	000000	T.CS1: .WORD	0	: TEMPORARY STORAGE FOR COMMAND AND STATUS REGISTER 1
3555					
3556	002772	000000	T.CS2: .WORD	0	: TEMPORARY STORAGE FOR COMMAND AND STATUS REGISTER 2
3557					
3558	002774	000000	T.WCR: .WORD	0	: TEMPORARY STORAGE FOR WORD COUNT REGISTER
3559	002776	000000	T.BA: .WORD	0	: TEMPORARY STORAGE FOR BUS ADDRESS REGISTER
3560	003000	000000	T.DA: .WORD	0	: TEMPORARY STORAGE FOR DISK TRACK AND SECTOR

TEMPORARY CONTROLLER REGISTER STORAGE

3561	003002	000000	T.DC:	.WORD	0	: TEMPORARY STORAGE FOR DRIVE CYLINDER
3562	003004	000000	T.ASOF:	.WORD	0	: TEMPORARY STORAGE FOR ATTENTION SUMMARY AND OFFSET
3563						
3564	003006	000000	T.ER:	.WORD	0	: TEMPORARY STORAGE FOR ERROR REGISTER
3565	003010	000000	T.DS:	.WORD	00	: TEMPORARY STORAGE FOR DRIVE STATUS REGISTER
3566	003012	000000	T.MR1:	.WORD	00	: TEMPORARY STORAGE FOR MAINTENANCE REGISTER 1
3567	003014	000000	T.MR2:	.WORD	00	: TEMPORARY STORAGE FOR MAINTENANCE REGISTER 2
3568	003016	000000	T.MR3:	.WORD	00	: TEMPORARY STORAGE FOR MAINTENANCE REGISTER 3
3569	003020	000000	T.POS:	.WORD	00	: TEMPORARY STORAGE FOR ECC POSITION
3570	003022	000000	T.PAT:	.WORD	00	: TEMPORARY STORAGE FOR ECC PATTERN
3571	003024	000000	T.DB:	.WORD	0	: TEMPORARY STORAGE FOR DATA BUFFER REGISTER
3572						
3573			.SBTTL	DRIVER	PARAMETERS	
3574						
3575	003026	177440	RKBAS:	.WORD	177440	: ADDRESS OF RK611 UNIBUS ADDRESS BLOCK
3576	003030	000210	RKVEC:	.WORD	210	: ADDRESS OF R611 VECTOR
3577	003032	000240	RKPRI:	.WORD	PR5	: RK611 INTERRUPT PRIORITY
3578	003034	041220	A.NORM:	ERRFRE		: ADDRESS OF NORMAL RETURN FROM DRIVER
3579	003036	042466	A.ABNL:	ERRHDL		: ADDRESS OF ABNORMAL RETURN FROM DRIVER
3580	003040	041762	A.CONT:	CONERR		: ADDRESS OF CONTROLLER ERROR RETURN
3581	003042	000000	E.CONT:	.WORD	0	: CONTROLLER ERROR STATUS
3582						: THIS LOCATION IS CLEARED WHEN EVERY COMMAND IS INITIATED. IF A CONTROLLER ERROR OCCURS THE FOLLOWING BIT ASSIGNMENT IS USED:
3583						
3584						
3585						
3586						
3587		000001	E.CCLR=	BIT0		: CLEAR CONTROLLER DID NOT CLEAR ERROR
3588		000002	E.NOAT=	BIT1		: NO ATTENTION IN ATTENTION SUMMARY REG
3589		000004	E.UATT=	BIT2		: UNSOLICATED ATTENTION (SEQUENTIAL ONLY)
3590		000010	E.UDAT=	BIT3		: UNEXPECTED DATA TYPE ERROR
3591		000020	E.CLAT=	BIT4		: ATTENTION DID NOT RESET WITH CLEAR
3592		000040	E.SCLR=	BITS		: SUBSYSTEM CLEAR DID NOT CLEAR DRIVE ATTENTION
3593						
3594		000100	E.ILLD=	BIT6		: ILLEGAL DRIVER COMMAND
3595		000400	E.DLT=	BIT8		: DATA LATE WHEN UNLOADING HEADER
3596		001000	E.CERR=	BIT9		: CONTROLLER ERROR DURING DRIVER SERVICING
3597		002000	E.DPAR=	BIT10		: DRIVE DETECTED PARITY ERROR
3598		040000	E.CMTO=	BIT14		: CONTROLLER COMMAND TIME OUT (QUEUED ONLY)
3599		100000	E.MDS=	BIT15		: MULTIPLE DRIVE SELECT
3600						
3601	003044	000000	O.WAIT:	.WORD	0	: PARAMETER BLOCK OF THE DRIVE
3602						: WAITING FOR COMMAND COMPLETION
3603	003046	000400	W.MTIM:	.WORD	400	: LOOP COUNTER FOR MILLISECOND SCAN OF DRIVE
3604	003050	000400	W.MILI:	.WORD	400	: 16 MILLISECOND TIME FOR PROGRAM
3605						
3606						
3607						
3608						
3609						
3610						
3611						
3612						
3613						
3614						
3615						
3616						

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

DRIVER PARAMENTERS

```

3617 003052 000300          W.SEC: .WORD 300          ;SECOND COUNT COUNT FOR ALL COMMANDS
3618                                ; EXCEPT START SPINDLE
3619 003054 003000          W.8SEC: .WORD 3000        ;8 SECOND FOR DRIVE CYCLE DOWN
3620 003056 030000          W.MIN: .WORD 30000       ;MINUTE TIME FOR START SPINDLE
3621 003060 000000          HDR.AD: .WORD 0          ;ADDRESS USED FOR READ ALL HEADERS
3622 003062 000000          HDR.CT: .WORD 0          ;NUMBER OF HEADERS LEFT TO READ FOR READ
3623                                ; ALL HEADERS
3624 003064 000            I.ISRL: .BYTE 0          ; INTERRUPT OR RELEASED COMMAND ISSUED
3625 003065 002 004 010    H.HEAD: .BYTE 2,4,10      ;HEAD DECODES
3626 003070 000            W.TIME: .BYTE 0          ;DRIVES BEING WATCH-DOG TIMED
3627
3628                                .SBTTL INTERRUPT MASKS
3629
3630 003071 000            INTMSK: .BYTE 0          ;INTERRUPT MASKS FOR DRIVE IN PARAMETER BLOCK
3631
3632                                ; INTERRUPT MASK TABLE
3633
3634 003072 001            I.DRV: .BYTE 1          ; INTERRUPT MASK FOR DRIVE 0
3635 003073 002            .BYTE 2          ; INTERRUPT MASK FOR DRIVE 1
3636 003074 004            .BYTE 4          ; INTERRUPT MASK FOR DRIVE 2
3637 003075 010            .BYTE 10         ; INTERRUPT MASK FOR DRIVE 3
3638 003076 020            .BYTE 20         ; INTERRUPT MASK FOR DRIVE 4
3639 003077 040            .BYTE 40         ; INTERRUPT MASK FOR DRIVE 5
3640 003100 100            .BYTE 100        ; INTERRUPT MASK FOR DRIVE 6
3641 003101 200            .BYTE 200        ; INTERRUPT MASK FOR DRIVE 7
3642
3643                                .SBTTL PARAMETER BLOCK TABLE
3644
3645 003102 002620          PBLKT:  PARMD          ;ADDRESS OF PARAMETER BLOCK GIVEN WITH
3646                                ;DRIVE CALL. MUST BE LOADED INTO PBLKT
3647
3648
3649                                .SBTTL TIME FOR WATCH-DOG TIMER
3650
3651 003104 000000          W.DRV: .WORD 0          ;TIME FOR INSTRUCTION IN PARAMETER BLOCK
3652                                .SBTTL PROGRAM SPECIFIC RESERVED LOCATIONS
3653 003106 000            MDFLAG: .BYTE 0          ;FLAG TO INDIC. DEFLT OR PARAM MODE
3654 003107 000            XXDPCH: .BYTE 0          ;XXDP CHAIN MODE FLAG
3655 003110 000            TSTING: .BYTE 0          ;CURRENTLY RUNNING TESTS IF = 1
3656 003111 000            DERCNT: .BYTE 0          ;DATA ERROR COUNT
3657 003112 000            OPCOMP: .BYTE 0          ;OPERATION COMPLETE FLAG
3658 003113 000            DONE: .BYTE 0          ;DONE SWITCH
3659 003114 000            TYPFMT: .BYTE 0          ;DRIVE TYPE & FORMAT CONTROL
3660 003115 000            FORMAT: .BYTE 0          ;DRIVE FORMAT IN BIT 4 OF BYTE
3661 003116 000            ERRCNT: .BYTE 0          ;ERROR COUNT
3662 003117 004            ERRLMT: .BYTE 4          ;ERROR LIMIT
3663 003120 000            DRVERS: .BYTE 0          ;ERROR COUNT FOR CURRENT DRIVE
3664 003121 000            OPCONT: .BYTE 0          ;OPERATION CONTROL SWITCHES
3665 003122 000            PCLKF: .BYTE 0          ;IF BYTE=1, KW11-P CLOCK IS PRESENT
3666 003123 000            DOTIM: .BYTE 0          ;IF BYTE=1, DO TIMING TESTS
3667 003124 000            XOVLAD: .BYTE 0          ;FLAG = 1 IF XXDP IS POSSIBLY OVERLAID
3668 003125 000            XDPSVD: .BYTE 0          ;FLAG = 1 IF XXDP IS SAVED
3669 003126 000            DULACS: .BYTE 0          ;FLAG=1 IF DUAL ACCESS TEST
3670 003127 000            DRNAFG: .PYTE 0          ;=1 INDICATES DRIVE SIEZED BY OTHER PORT
3671 003130 000            REISSU: .BYTE 0          ;DUAL-ACC FLAG TO RE-ISSUE COMMAND
3672 003131 000            WCEFLG: .BYTE 0          ;WRITE CHECK ERROR FLAG

```

3673	003132	000	DLTFLG: .BYTE	0	:DATA LATE ERROR FLAG
3674	003133	000	NORTRY: .BYTE	0	: "NO-RETRY" FLAG
3675	003134	000	UBMPRS: .BYTE	0	: UNIBUS MAP PRESENT IF = 1
3676	003135	000	MEMABT: .BYTE	0	: SET BYTE = 1 FOR NO ABORT
3677					: ON MEMORY PARITY ERRORS
3678	003136	000	HLPOVL: .BYTE	0	: HELP FILE OVERLAID INDICATOR
3679	003137	000	NOTYPE: .BYTE	0	: INDICATES PROGRAM JUST LOADED IF 0
3680		000001	WHDSW=BIT0		: WRITE HEADER & DATA SWITCH
3681		000002	VFHDSW=BIT1		: VERIFY HEADERS SWITCH
3682		000004	WCDASW=BIT2		: WRITE CHECK DATA SWITCH
3683		000010	RCDASW=BIT3		: READ CHECK DATA SWITCH
3684		000020	OREGSW=BIT4		: OFFSET REQUIRED SWITCH
3685					
3686			.EVEN		
3687	003140	177546	LKS: .WORD	177546	: KW11-L CLOCK STATUS REGISTER
3688	003142	172540	PKS: .WORD	172540	: KW11-P CONTROL AND STATUS REGISTER
3689	003144	172542	PKSB: .WORD	172542	: KW11-P COUNT SET BUFFER REGISTER
3690	003146	172544	PKRB: .WORD	172544	: KW11-P COUNTER REGISTER
3691	003150	000100	LCVEC: .WORD	100	: KW11-L VECTOR STORAGE
3692	003152	000104	PCVEC: .WORD	104	: KW11-P VECTOR STORAGE
3693		000114	MEMVEC=114		: MEMORY PARITY TRAP VECTOR
3694	003154	000000	TCONLO: .WORD	0	: LO BITS OF CALIBRATION TIME CONSTANT
3695	003156	000000	TCONHI: .WORD	0	: HI BITS OF CALIB TIME CONST
3696	003160	000000	BLWMIN: .WORD	0	: COUNT OF TIMES BELOW MIN
3697	003162	000000	ABVMX1: .WORD	0	: COUNT OF FORWARD TIMES ABOVE MAX
3698	003164	000000	ABVMX2: .WORD	0	: COUNT OF REVERSE TIMES ABOVE MAX
3699	003166	000000	SUML01: .WORD	0	: LO BITS OF FORWARD TIME SUM
3700	003170	000000	SUMHI1: .WORD	0	: HI BITS OF FORWARD TIME SUM
3701	003172	000000	SUML02: .WORD	0	: LO BITS OF REVERSE TIME SUM
3702	003174	000000	SUMHI2: .WORD	0	: HI BITS OF REVERSE TIME SUM
3703	003176	000000	SAVPAR: .WORD	0	: SAVE WORD FOR PAR CONSTANT
3704	003200	000000	SAVWRD: .WORD	0	: SCRATCH WORD
3705	003202	000010	CYLLST: .BLKW	↑D8	: LIST OF PSEUDO-RAND CYL ADDRESSES
3706	003222	000400	BSSOFT: .BLKW	↑D256	: RECORD OF BAD SECTORS FROM SOFTWARE
3707	004222	000400	BSFACT: .BLKW	↑D256	: RECORD OF BAD SECTORS FROM FACTORY
3708	005222	000006	PRVCMD: .BLKW	6	: PREVIOUS COMMAND STORAGE
3709	005236	000006	COMSTR: .BLKW	6	: CURRENT COMMAND STORAGE
3710	005252	000000	PRMPLO: .WORD	0	: PREV. U.B. MAP REG 0
3711	005254	000000	PRMPHO: .WORD	0	
3712	005256	000000	CRMPLO: .WORD	0	: CURRENT U.B. MAP REG 0
3713	005260	000000	CRMPHO: .WORD	0	
3714	005262	000102	BUFF0: .BLKW	↑D66	: OUTPUT BUFFER 1
3715	005466	000000	LOWOCT: .WORD	0	: LOW 16 BITS OF CONVERTED BINARY NUMBER
3716	005470	000000	HIGOCT: .WORD	0	: HIGH BITS OF CONVERTED BINARY NO.
3717	005472	000000	BUFprt: .WORD	0	: BUFFER POINTER
3718	005474	000000	RECODE: .WORD	0	: RECOVERY CODE WORD
3719	005476	000000	ERRCOM: .WORD	0	: ERROR COMMAND
3720	005500	000000	DRIVE: .WORD	0	: NO. OF DRIVE IN USE
3721	005502	000000	STALLS: .WORD	0	: CURRENT NO. OF UNIT STALLS TO APPLY
3722	005504	000000	CYLNRD: .WORD	0	: CURRENT CYLINDER NUMBER
3723	005506	000000	FS: .WORD	0	: FIRST SECTOR LIMIT
3724	005510	000000	LS: .WORD	0	: LAST SECTOR LIMIT
3725	005512	000000	NCYL1: .WORD	0	: NEXT CYL SCRATCH WORD
3726	005514	000000	NCYL2: .WORD	0	: NEXT CYL SCRATCH WORD
3727	005516	000000	OFINUS: .WORD	0	: OFFSET IN USE
3728	005520	000000	TRACK: .WORD	0	: TRACK IN USE

3729	005522	000000	INTCHR: .WORD	0	:TTY INTERRUPT INPUT WORD
3730	005524	000000	SELECT: .WORD	0	:ANY BIT SET IN LOW BYTE GIVES AUTO-SELECTED DRIVE NO.
3731	005526	000000	ONLINE: .WORD	0	:ANY BIT SET IN LOW BYTE INDICATES A DRIVE WAS ON-LINE
3732	005530	000000	NEWON: .WORD	0	:NEW LOOK AT ONLINE DRIVES
3733	005532	000000	SCRACH: .WORD	0	:ALL-PURPOSE SCRATCH WORD
3734	005534	000000	PATRN: .WORD	0	:DATA PATTERN LIST WORD
3735	005536	000000	XXDPAD: .WORD	0	:STARTING ADDRESS OF XXDP LOADER
3736	005540	000002	XDPSAV: .BLKW	2	:XXDP PHYSICAL SAVE ADDRESS
3737	005544	000000	PLOFST: .WORD	0	:(+) OFFSET VALUE
3738	005546	000000	NGOFST: .WORD	0	:(-) OFFSET VALUE
3739	005550	000005	SAVPRS: .BLKW	5	:SAVE INITIAL PARAMS FOR XFER
3740	005562	000000	WDSXFR: .WORD	0	:NO. OF WORDS ACTUALLY XFERRED
3741	005564	000000	FINCYL: .WORD	0	:FINAL CYLINDER ADRS
3742	005566	000	FINTRK: .BYTE	0	:FINAL TRACK ADRS
3743	005567	000	FINSEC: .BYTE	0	:FINAL SECTOR ADRS
3744	005570	000000	LASTWC: .WORD	0	:ACTUAL FINAL WC
3745	005572	157776	MAHILM: .WORD	157776	:MA UPPER LIMIT
3746	005574	000077		77	
3747	005576	064640	MA: .WORD	RWBUF	:LO BITS OF PHYS MEM ADRS
3748	005600	000000		0	:HI BITS OF PHYS MEM ADRS
3749	005602	000002	PMA: .BLKW	2	:PARTIAL TRANSFER MEMORY START ADRS

:LIST OF DESIRED DRIVES TO TEST (IF BYTE=0, DON'T TEST DRIVE)

3754	005606		DRVLST:		
3755	005606	001	.BYTE	1	:DRIVE 0
3756	005607	001	.BYTE	1	:DRIVE 1
3757	005610	001	.BYTE	1	:DRIVE 2
3758	005611	001	.BYTE	1	:DRIVE 3
3759	005612	001	.BYTE	1	:DRIVE 4
3760	005613	001	.BYTE	1	:DRIVE 5
3761	005614	001	.BYTE	1	:DRIVE 6
3762	005615	001	.BYTE	1	:DRIVE 7

:LIST OF ITERATION COUNTS FOR DESIRED TESTS (IF=0, TEST NOT RUN)
 :MAXIMUM ALLOWABLE NUMBER OF ITERATIONS = 77776 (OCT)

3768	005616		TSTLST:		
3769	005616	000002	.WORD	2	:TEST 1
3770	005620	000002	.WORD	2	:TEST 2
3771	005622	000002	.WORD	2	:TEST 3
3772	005624	000002	.WORD	2	:TEST 4
3773	005626	000002	.WORD	2	:TEST 5
3774	005630	000100	.WORD	100	:TEST 6

:LIST OF DEFAULT ITERATION COUNTS FOR ALL TESTS

3779	005632		DFLTST:		
3780	005632	000002	.WORD	2	:TEST 1
3781	005634	000002	.WORD	2	:TEST 2
3782	005636	000002	.WORD	2	:TEST 3
3783	005640	000002	.WORD	2	:TEST 4
3784	005642	000002	.WORD	2	:TEST 5

PROGRAM SPECIFIC RESERVED LOCATIONS

3785 005644 000100 .WORD 100 ;TEST 6

3786
 3787
 3788 000006 NMTSTS=<DFLTST-TSTLST>/2 ;TOTAL NO. OF AUTOMATIC TESTS

3789
 3790

3791 :OPERATING PARAMETER LIST

3792 005646 PRMLST:
 3793 005646 000000 FC: .WORD 0 ;FIRST CYLINDER
 3794 005650 000632 LC: .WORD 632 ;LAST CYLINDER
 3795 005652 000000 FT: .WORD 0 ;FIRST TRACK
 3796 005654 000002 LT: .WORD 2 ;LAST TRACK
 3797 005656 000000 SO: .WORD 0 ;FIRST SECTOR IF 20(DEC) SECTOR FMT
 3798 005660 000023 S1: .WORD 23 ;LAST SECTOR IF 20(DEC) SECTOR FMT
 3799 005662 000000 S2: .WORD 0 ;FIRST SECTOR IF 22(DEC) SECTOR FMT
 3800 005664 000025 S3: .WORD 25 ;LAST SECTOR IF 22(DEC) SECTOR FMT
 3801 005666 000000 PT: .WORD 0 ;DATA PATTERN SELECT WORD
 3802 005670 000000 CS: .WORD 0 ;CONTROL SWITCH WORD
 3803 005672 000000 ST: .WORD 0 ;NUMBER OF UNIT STALLS

3804
 3805

3806 :DEFAULT OPERATING PARAMETER VALUES

3807 005674 PRDFLT:
 3808 005674 000000 .WORD 0 ;FC DEFAULT
 3809 005676 000632 .WORD 632 ;LC DEFAULT
 3810 005700 000000 .WORD 0 ;FT DEFAULT
 3811 005702 000002 .WORD 2 ;LT DEFAULT
 3812 005704 000000 .WORD 0 ;SO DEFAULT
 3813 005706 000023 .WORD 23 ;S1 DEFAULT
 3814 005710 000000 .WORD 0 ;S2 DEFAULT
 3815 005712 000025 .WORD 25 ;S3 DEFAULT
 3816 005714 000000 .WORD 0 ;PT DEFAULT
 3817 005716 000000 .WORD 0 ;CS DEFAULT
 3818 005720 000000 .WORD 0 ;ST DEFAULT

3819
 3820

3821 :OPERATING PARAMETER VALUE LOW AND HIGH LIMITS

3822 PRMLIM:
 3823 005722 .WORD 0 ;FC LIMITS
 3824 005722 000000 .WORD 0 ;LC LIMITS
 3825 005724 000631 .WORD 631 ;LC LIMITS
 3826 005726 000000 .WORD 0 ;LC LIMITS
 3827 005730 000632 .WORD 632 ;LC LIMITS
 3828 005732 000000 .WORD 0 ;FT LIMITS
 3829 005734 000002 .WORD 2 ;FT LIMITS
 3830 005736 000000 .WORD 0 ;LT LIMITS
 3831 005740 000002 .WORD 2 ;LT LIMITS
 3832 005742 000000 .WORD 0 ;SO LIMITS
 3833 005744 000023 .WORD 23 ;SO LIMITS
 3834 005746 000000 .WORD 0 ;S1 LIMITS
 3835 005750 000023 .WORD 23 ;S1 LIMITS
 3836 005752 000000 .WORD 0 ;S2 LIMITS
 3837 005754 000025 .WORD 25 ;S2 LIMITS
 3838 005756 000000 .WORD 0 ;S3 LIMITS
 3839 005760 000025 .WORD 25 ;S3 LIMITS
 3840 005762 000000 .WORD 0 ;PT LIMITS

M06

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 78

SEQ 0077

PROGRAM SPECIFIC RESERVED LOCATIONS

3841	005764	177777	.WORD	177777	
3842	005766	000000	.WORD	0	;CS LIMITS
3843	005770	000062	.WORD	000062	
3844	005772	000000	.WORD	0	;ST LIMITS
3845	005774	177777	.WORD	177777	

ASCII PARAMETER MNEMONICS PRMNEM:

3850	005776		.ASCII	/FC/
3851	005776	041506	.ASCII	/LC/
3852	006000	041514	.ASCII	/FT/
3853	006002	052106	.ASCII	/LT/
3854	006004	052114	.ASCII	/SO/
3855	006006	030123	.ASCII	/S1/
3856	006010	030523	.ASCII	/S2/
3857	006012	031123	.ASCII	/S3/
3858	006014	031523	.ASCII	/PT/
3859	006016	052120	.ASCII	/CS/
3860	006020	051503	.ASCII	/ST/
3861	006022	052123		

DATA PATTERN 00 HI-LO FREQ. MIX

3866			PAT00:	
3867	006024			177777
3868	006024	177777		177777
3869	006026	177777		177777
3870	006030	177777		177777
3871	006032	052525		052525
3872	006034	052525		052525
3873	006036	052525		052525
3874	006040	177777		177777
3875	006042	177777		177777
3876	006044	052525		052525
3877	006046	052525		052525
3878	006050	177777		177777
3879	006052	052525		052525
3880	006054	177252		177252
3881	006056	177252		177252
3882	006060	172765		172765
3883	006062	172765		172765

DATA PATTERN 01 HI FREQ. PHASE MIX

3888			PAT01:	
3889	006064			000000
3890	006064	000000		000000
3891	006066	000000		000000
3892	006070	000000		000000
3893	006072	177777		177777
3894	006074	177777		177777
3895	006076	177777		177777
3896	006100	000000		000000

3897	006102	000000	000000
3898	006104	177777	177777
3899	006106	177777	177777
3900	006110	000000	000000
3901	006112	177777	177777
3902	006114	000000	000000
3903	006116	177777	177777
3904	006120	000000	000000
3905	006122	177777	177777
3906			
3907			
3908			
3909			
3910			
3911	006124		
3912	006124	052525	052525
3913	006126	052525	052525
3914	006130	052525	052525
3915	006132	125252	125252
3916	006134	125252	125252
3917	006136	125252	125252
3918	006140	052525	052525
3919	006142	052525	052525
3920	006144	125252	125252
3921	006146	125252	125252
3922	006150	052525	052525
3923	006152	125252	125252
3924	006154	052525	052525
3925	006156	125252	125252
3926	006160	052525	052525
3927	006162	125252	125252
3928			
3929			
3930			
3931			
3932			
3933	006164		
3934	006164	133333	133333
3935	006166	066666	066666
3936	006170	155555	155555
3937	006172	155555	155555
3938	006174	133333	133333
3939	006176	066666	066666
3940	006200	066666	066666
3941	006202	155555	155555
3942	006204	155555	155555
3943	006206	133333	133333
3944	006210	133333	133333
3945	006212	133333	133333
3946	006214	133333	133333
3947	006216	133333	133333
3948	006220	133333	133333
3949	006222	133333	133333
3950			
3951			
3952			

:DATA PATTERN 02
LO FREQ. PHASE MIX
PAT02:

:DATA PATTERN 03
MAX. PRECOMP. PHASE MIX
PAT03:

:DATA PATTERN 04
: ROTATING BOUNDARY PULSE PRECOMP.

PAT04:
121105
150442
064221
132110
055044
026422
013211
105504
042642
021321
110550
044264
022132
011055
104426
042213

006262
006264
006266
006270
006272
006274
006276
006300
006302
006304
006306
006310
006312
006314
006316
006320
006322

:DATA PATTERN 05
: ROTATING CELL PULSE PRECOMP.

PAT05:
026455
113226
045513
122645
151322
064551
132264
055132
026455
113226
045513
122645
151322
064551
132264
055132

006264
006264
006266
006270
006272
006274
006276
006300
006302
006304
006306
006310
006312
006314
006316
006320
006322

:DATA PATTERN 06
: ALL ZEROS

PAT06:
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000
000000

006324
006324
006326
006330
006332
006334
006336
006340
006342
006344
006346

40009 006350 000000
40010 006352 000000
40011 006354 000000
40012 006356 000000
40013 006358 000000
40014 006360 000000
40015 006362 000000

000000
000000
000000
000000
000000
000000
000000

:DATA PATTERN 07
: ALL ONES

PAT07: 177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777
177777

40020 006364 177777
40021 006366 177777
40022 006368 177777
40023 006370 177777
40024 006372 177777
40025 006374 177777
40026 006376 177777
40027 006378 177777
40028 006400 177777
40029 006402 177777
40030 006404 177777
40031 006406 177777
40032 006410 177777
40033 006412 177777
40034 006414 177777
40035 006416 177777
40036 006420 177777
40037 006422 177777

:DATA PATTERN 08
: SHIFTED 1 IN FIELD OF ZEROS

PAT08: 000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000

40042 006424 000001
40043 006426 000002
40044 006428 000004
40045 006430 000010
40046 006432 000020
40047 006434 000040
40048 006436 000100
40049 006440 000200
40050 006444 000400
40051 006446 001000
40052 006450 002000
40053 006452 004000
40054 006454 010000
40055 006456 020000
40056 006460 040000
40057 006462 100000

:DATA PATTERN 09
: SHIFTED 0 IN FIELD OF ONES

PAT09: 177776

40664 006464 177776
40665 006464 177776

4065	006466	177775	177775
4066	006470	177773	177773
4067	006472	177767	177767
4068	006474	177757	177757
4069	006476	177737	177737
4070	006500	177677	177677
4071	006502	177577	177577
4072	006504	177377	177377
4073	006506	176777	176777
4074	006510	175777	175777
4075	006512	173777	173777
4076	006514	167777	167777
4077	006516	157777	157777
4078	006520	137777	137777
4079	006522	077777	077777

:DATA PATTERN 10
ALTERNATING 0-1

4085	006524		PAT10:	052525
4086	006524	052525		052525
4087	006526	052525		052525
4088	006530	052525		052525
4089	006532	052525		052525
4090	006534	052525		052525
4091	006536	052525		052525
4092	006540	052525		052525
4093	006542	052525		052525
4094	006544	052525		052525
4095	006546	052525		052525
4096	006550	052525		052525
4097	006552	052525		052525
4098	006554	052525		052525
4099	006556	052525		052525
4100	006560	052525		052525
4101	006562	052525		052525

:DATA PATTERN 11
ALTERNATING 1-0

4107	006564		PAT11:	125252
4108	006564	125252		125252
4109	006566	125252		125252
4110	006570	125252		125252
4111	006572	125252		125252
4112	006574	125252		125252
4113	006576	125252		125252
4114	006600	125252		125252
4115	006602	125252		125252
4116	006604	125252		125252
4117	006606	125252		125252
4118	006610	125252		125252
4119	006612	125252		125252
4120	006614	125252		125252

4121	006616	125252	125252
4122	006620	125252	125252
4123	006622	125252	125252

```

:DATA PATTERN 12
: SHIFTING ZEROS AND ONES
PAT12:

```

4124	006624	000001	000001
4130	006624	000003	000003
4131	006626	000007	000007
4132	006630	000017	000017
4133	006632	000037	000037
4134	006634	000077	000077
4135	006636	000177	000177
4136	006640	000377	000377
4137	006642	000777	000777
4138	006644	001777	001777
4139	006646	003777	003777
4140	006650	007777	007777
4141	006652	017777	017777
4142	006654	037777	037777
4143	006656	077777	077777
4144	006660	177777	177777

```

:DATA PATTERN 13
: COMPOSITE ROTATING
PAT13:

```

4151	006664	072307	072307
4152	006664	135143	135143
4153	006666	156461	156461
4154	006670	167230	167230
4155	006672	073514	073514
4156	006674	035646	035646
4157	006676	016723	016723
4158	006700	107351	107351
4159	006702	143564	143564
4160	006704	061672	061672
4161	006706	030735	030735
4162	006710	114356	114356
4163	006712	046167	046167
4164	006714	123073	123073
4165	006716	151453	151453
4166	006720	164616	164616

```

:DATA PATTERN 14
: PSEUDO-RANDOM (COMPUTED BY PROGRAM)
PAT14:

```

4173	006724	.WORD	.WORD
4174	006724	000000	000000
4175	006726	000000	000000
4176	006730	000000	000000

4177	006732	000000	.WORD
4178	006734	000000	.WORD
4179	006736	000000	.WORD
4180	006740	000000	.WORD
4181	006742	000000	.WORD
4182	006744	000000	.WORD
4183	006746	000000	.WORD
4184	006750	000000	.WORD
4185	006752	000000	.WORD
4186	006754	000000	.WORD
4187	006756	000000	.WORD
4188	006760	000000	.WORD
4189	006762	000000	.WORD

:USER-DEFINED DATA PATTERN 15
 PAT15:

4193	006764			
4194	006764	072307	072307	:WORD 00
4195	006766	135143	135143	:WORD 01
4196	006770	156461	156461	:WORD 02
4197	006772	167230	167230	:WORD 03
4198	006774	073514	073514	:WORD 04
4199	006776	035646	035646	:WORD 05
4200	007000	016723	016723	:WORD 06
4201	007002	107351	107351	:WORD 07
4202	007004	143564	143564	:WORD 10
4203	007006	061672	061672	:WORD 11
4204	007010	030735	030735	:WORD 12
4205	007012	114356	114356	:WORD 13
4206	007014	046167	046167	:WORD 14
4207	007016	123073	123073	:WORD 15
4208	007020	151453	151453	:WORD 16
4209	007022	164616	164616	:WORD 17

4210				
4211				
4212				
4213	057467	MINL01=↑D24375	:LO BITS OF SPEC'D MIN ROT. LATENCY	
4214	000000	MINHI1=0	:HI BITS OF SPEC'D MIN ROT. LATENCY	
4215	062031	MAXL01=↑D25625	:LO BITS OF SPEC'D MAX ROT. LATENCY	
4216	000000	MAXHI1=0	:HI BITS OF SPEC'D MAX ROT. LATENCY	
4217	017500	MAXL02=↑D8000	:LO BITS OF SPEC'D MAX 1 CYL SEEK TIME	
4218	000000	MAXHI2=0	:HI BITS OF SPEC'D MAX 1 CYL SEEK TIME	
4219	112160	MAXL03=↑D38000	:LO BITS OF SPEC'D MAX AVG SEEK TIME	
4220	000000	MAXHI3=0	:HI BITS OF SPEC'D MAX AVG SEEK TIME	
4221	022370	MAXL04=↑D9464	:LO BITS OF SPEC'D MAX 410 CYL SEEK TIME	
4222	000001	MAXHI4=1	:HI BITS OF SPEC'D MAX 410 CYL SEEK TIME	
4223	002734	RNDSHT=↑D1500	:SHORT RANDOM SEEKS = 1500(10)	
4224	016514	RNDLNG=↑D7500	:LONG RANDOM SEEKS = 7500(10)	
4225	000632	LSTCYL=632	:LAST CYL. = 410(10)	
4226	000002	LSTRK=2	:LAST TRACK = 2	
4227	000365	ALNCYL=365	:ALIGNMENT CYLINDER =245(10)	
4228	000002	BSERR=BIT1	:BSE ERROR	
4229	000004	HVRCER=BIT2	:HVRC ERROR	
4230	000010	OPIERR=BIT3	:OPI ERROR	
4231	000020	DCKERR=BIT4	:DATA CHECK ERROR	
4232	000040	ECCNC=BITS	:ECC NON-CORRECTABLE	

4233	000100			WCERR=BIT6	:WRITE CHECK ERROR
4234	000200			ABORT=BIT7	:ABORT
4235	000400			LEV2ER=BIT8	:LEVEL TWO ERROR
4236	001000			BADSEC=BIT9	:BAD SECTOR FLAG
4237	002000			TWOTOS=BIT10	:TWO TIME OUTS
4238	004000			RCLREQ=BIT11	:RECALIBRATE REQUIRED
4239	010000			DRNAVL=BIT12	:DRIVE NOT RELSD BY OTHER PORT
4240	100000			ANYDER=BIT15	:ANY ERROR DETECTED FLAG
4241					
4242	007024	005015	042115	030455	DZR6N: .ASCIZ <15><12>/MD-11-DZR6N-C/
4243	007032	026461	055104	033122	
4244	007040	026516	000103		
4245	007044	026440	051040	033113	SUBVER: .ASCIZ & - RK611/RK06 SUBSYSTEM VERIFICATION : PART 3
4246	007052	030461	051057	030113	
4247	007060	020066	052523	051502	
4248	007066	051531	042524	020115	
4249	007074	042526	044522	044506	
4250	007102	040503	044524	047117	
4251	007110	035040	050040	051101	
4252	007116	020124	000		
4253	007121	061	005015	000	PART1: .ASCIZ /1/<15><12>
4254	007125	062	005015	000	PART2: .ASCIZ /2/<15><12>
4255	007131	015	005012	040514	LSTMEM: .ASCIZ <15><12><12>/LAST PHYS MEM ADR = /
4256	007136	052123	050040	054510	
4257	007144	020123	042515	020115	
4258	007152	042101	020122	020075	
4259	007160	000			
4260	007161	117	042526	046122	OVLODR: .ASCIZ /OVERLAY LOADER ? (Y OR N) * /
4261	007166	054501	046040	040517	
4262	007174	042504	020122	020077	
4263	007202	054450	047440	020122	
4264	007210	024516	025040	000040	
4265	007216	005015	040520	040522	PRMINP: .ASCIZ <15><12>/PARAMETER INPUT MODE/<15><12><12>
4266	007224	042515	042524	020122	
4267	007232	047111	052520	020124	
4268	007240	047515	042504	005015	
4269	007246	000012			
4270	007250	045522	033060	041040	RKBADR: .ASCIZ /RK06 BUS ADR = /
4271	007256	051525	040440	051104	
4272	007264	036440	000040		
4273	007270	045522	033060	053040	RKVADR: .ASCIZ /RK06 VEC ADR = /
4274	007276	041505	040440	051104	
4275	007304	036440	000040		
4276	007310	045522	033060	050040	RKPRTY: .ASCIZ /RK06 PRIORITY =/
4277	007316	044522	051117	052111	
4278	007324	020131	000075		
4279	007330	053523	020122	020075	SWRMSG: .ASCIZ /SWR = /
4280	007336	000			
4281	007337	040	047040	053505	NEWMSG: .ASCIZ / NEW = /
4282	007344	036440	000040		
4283	007350	005015	051104	053111	DRVSEQ: .ASCIZ <15><12>/DRIVE(S) = /
4284	007356	024105	024523	036440	
4285	007364	000040			
4286	007366	051104	053111	020105	BADDRV: .ASCIZ /DRIVE /
4287	007374	020040	000		
4288	007377	116	047117	042455	NXDRIV: .ASCIZ /NON-EXISTENT/<15><12>

H07

4289	007404	044530	052123	047105	
4290	007412	006524	000012		
4291	007416	047516	020124	042522	NTREDY: .ASCIZ /NOT READY/<15><12>
4292	007424	042101	006531	000012	
4293	007432	051127	052111	026505	WRTLOK: .ASCIZ /WRITE-LOCKED/<15><12>
4294	007440	047514	045503	042105	
4295	007446	005015	000		
4296	007451	114	040517	042504	ALNPAK: .ASCIZ /LOADED WITH ALIGN PACK/<15><12>
4297	007456	020104	044527	044124	
4298	007464	040440	044514	047107	
4299	007472	050040	041501	006513	
4300	007500	000012			
4301	007502	043111	054040	042130	REPLPK: .ASCIZ /IF XXDP PACK ON DRV 0, TYPE "Y <CR>", & REPLACE IT : /
4302	007510	020120	040520	045503	
4303	007516	047440	020116	051104	
4304	007524	020126	026060	052040	
4305	007532	050131	020105	054442	
4306	007540	036040	051103	021076	
4307	007546	020054	020046	051040	
4308	007554	050105	040514	042503	
4309	007562	044440	020124	020072	
4310	007570	000			
4311	007571	015	005012	025052	NODRTS: .ASCII <15><12><12>/** NO DRIVES TO TEST/<15><12>
4312	007576	047040	020117	051104	
4313	007604	053111	051505	052040	
4314	007612	020117	042524	052123	
4315	007620	005015			
4316	007622	051120	051505	020123	CNTRDY: .ASCIZ /PRESS "CONT" WHEN RDY/<15><12>
4317	007630	041442	047117	021124	
4318	007636	053440	042510	020116	
4319	007644	042122	006531	000012	
4320	007652	040510	052114	051040	HLTRQD: .ASCIZ /HALT REQUESTED/<15><12>
4321	007660	050505	042525	052123	
4322	007666	042105	005015	000	
4323	007673	104	044522	042526	DROXDP: .ASCIZ /DRIVE 0 IS LOAD MEDIUM/<15><12>
4324	007700	030040	044440	020123	
4325	007706	047514	042101	046440	
4326	007714	042105	052511	006515	
4327	007722	000012			
4328	007724	005015	052012	020117	ALDRVS: .ASCIZ <15><12><12>/TO TEST ALL DRIVES TYPE "A" <CR>, ELSE <CR>/<15><12>/** /
4329	007732	042524	052123	040440	
4330	007740	046114	042040	044522	
4331	007746	042526	020123	054524	
4332	007754	042520	021040	021101	
4333	007762	036040	051103	026076	
4334	007770	042440	051514	020105	
4335	007776	041474	037122	005015	
4336	010004	020052	000		
4337	010007	015	046012	036440	TSTMDS: .ASCII <15><12>/L = LIST TESTS/<15><12>
4338	010014	046040	051511	020124	
4339	010022	042524	052123	006523	
4340	010030	012			
4341	010031	103	036440	041440	.ASCII /C = CHANGE TESTS/<15><12>
4342	010036	040510	043516	020105	
4343	010044	042524	052123	006523	
4344	010052	012			

4345	010053	111	036440	044440	.ASCIZ /I = INPUT PARAMS, RUN TESTS/<15><12>
4346	010060	050116	052125	050040	
4347	010066	051101	046501	026123	
4348	010074	051040	047125	052040	
4349	010102	051505	051524	005015	
4350	010110	000			
4351	010111	015	042412	052116	ENTLCI: .ASCIZ <15><12>/ENTER L,C, OR I/<15><12>/* /
4352	010116	051105	046040	041454	
4353	010124	020054	051117	044440	
4354	010132	005015	020052	000	
4355	010137	124	020117	042504	DFTEST: .ASCIZ /TO DEFAULT TESTS TYPE D <CR>, ELSE <CR>/<15><12>/* /
4356	010144	040506	046125	020124	
4357	010152	042524	052123	020123	
4358	010160	054524	042520	042040	
4359	010166	036040	051103	026076	
4360	010174	042440	051514	020105	
4361	010202	041474	037122	005015	
4362	010210	020052	000		
4363	010213	015	052012	051505	TLSTHD: .ASCIZ <15><12>/TEST ITERATIONS/<15><12>
4364	010220	020124	020040	020040	
4365	010226	052111	051105	052101	
4366	010234	047511	051516	005015	
4367	010242	000			
4368	010243	015	052012	036440	EXPLAN: .ASCII <15><12>/T = TYPE PARAM LIST/<15><12>
4369	010250	052040	050131	020105	
4370	010256	040520	040522	020115	
4371	010264	044514	052123	005015	
4372	010272	020117	020075	050117	.ASCII /O = OPEN PARAM LIST/<15><12>
4373	010300	047105	050040	051101	
4374	010305	046501	046040	051511	
4375	010314	006524	012		
4376	010317	123	036440	051440	.ASCII /S = SET INDIVIDUAL PARAM/<15><12>
4377	010324	052105	044440	042116	
4378	010332	053111	042111	040525	
4379	010340	020114	040520	040522	
4380	010346	006515	012		
4381	010351	122	036440	051040	.ASCIZ /R = RUN TESTS/<15><12>
4382	010356	047125	052040	051505	
4383	010364	051524	005015	000	
4384	010371	015	042412	052116	PARMDE: .ASCIZ <15><12>/ENTER T,O,S, OR R/<15><12>
4385	010376	051105	052040	047454	
4386	010404	051454	020054	051117	
4387	010412	051040	005015	000	
4388	010417	015	025012	042040	DUACES: .ASCIZ <15><12>/* DUAL-ACCESS DATA TEST */<15><12>
4389	010424	040525	026514	041501	
4390	010432	042503	051523	042040	
4391	010440	052101	020101	042524	
4392	010446	052123	025040	005015	
4393	010454	000			
4394	010455	115	054101	053440	MAWRDC: .ASCIZ /MAX WORD COUNT = /
4395	010462	051117	020104	047503	
4396	010470	047125	020124	020075	
4397	010476	000			
4398	010477	052	020052	051440	SECNL1: .ASCII /** S0>S1/
4399	010504	037060	030523		
4400	010510	047040	052117	040440	NOTALD: .ASCIZ /NOT ALLOWED/<15><12>

4401	010516	046114	053517	042105	
4402	010524	005015	000		
4403	010527	052	020052	051440	SECNL2: .ASCIZ /** S2>S3/
4404	010534	037062	031523	000	
4405	010541	052	020052	043040	TRKNLW: .ASCIZ /** FT>LT/
4406	010546	037124	052114	000	
4407	010553	052	020052	053440	WC2BIG: .ASCIZ /** WC OR MA TOO LARGE/<15><12>
4408	010560	020103	051117	046440	
4409	010566	020101	047524	020117	
4410	010574	040514	043522	006505	
4411	010602	000012			
4412	010604	047524	042040	043105	DFQUES: .ASCIZ /TO DEFAULT ALL PARAMS, TYPE D <CR>, ELSE <CR> /<15><12> /* /
4413	010612	052501	052114	040440	
4414	010620	046114	050040	051101	
4415	010626	046501	026123	052040	
4416	010634	050131	020105	020104	
4417	010642	041474	037122	020054	
4418	010650	046105	042523	036040	
4419	010656	051103	006476	025012	
4420	010664	000040			
4421	010666	005015	051525	051105	PFIFTN: .ASCIZ <15><12>/USER-DEFINED PATTERN 15 : /<15><12>
4422	010674	042055	043105	047111	
4423	010702	042105	050040	052101	
4424	010710	042524	047122	030440	
4425	010716	020065	006472	000012	
4426	010724	005015	047515	044504	SELP15: .ASCIZ <15><12>/MODIFY PATTERN 15 : /<15><12>
4427	010732	054506	050040	052101	
4428	010740	042524	047122	030440	
4429	010746	020065	006472	000012	
4430	010754	047524	046440	042117	MDFY15: .ASCIZ /TO MODIFY PATTERN 15, TYPE M <CR>, ELSE <CR> /<15><12> /* /
4431	010762	043111	020131	040520	
4432	010770	052124	051105	020116	
4433	010776	032461	020054	054524	
4434	011004	042520	046440	036040	
4435	011012	051103	026076	042440	
4436	011020	051514	020105	041474	
4437	011026	037122	005015	020052	
4438	011034	000			
4439	011035	015	042412	052116	ENTPAS: .ASCIZ <15><12>/ENTER NO. OF PASSES (1-77777) : /<15><12> /* /
4440	011042	051105	047040	027117	
4441	011050	047440	020106	040520	
4442	011056	051523	051505	024040	
4443	011064	026461	033467	033467	
4444	011072	024467	035040	005015	
4445	011100	020052	000		
4446	011103	015	025012	020052	DROPDR: .ASCIZ <15><12>/** DROPPING DRIVE - 20 ERRORS/<15><12>
4447	011110	042040	047522	050120	
4448	011116	047111	020107	051104	
4449	011124	053111	020105	020055	
4450	011132	030062	042440	051122	
4451	011140	051117	006523	000012	
4452	011146	005015	052012	051505	TSTDRN: .ASCII <15><12><12>/TESTING DRIVE /
4453	011154	044524	043516	042040	
4454	011162	044522	042526	040	
4455	011167	040	005015	000	DRVNO: .ASCIZ / /<15><12>
4456	011173	104	044522	042526	DRIV: .ASCIZ /DRIVE /

4457	011200	000040			
4458	011202	040503	052122	020056	CART: .ASCIZ /CART. /
4459	011210	000			
4460	011211	123	051105	020056	SERNM: .ASCIZ /SER. NO. /
4461	011216	047516	020056	000040	
4462	011224	005015	040506	052103	FACTBS: .ASCIZ <15><12>/FACTORY /
4463	011232	051117	020131	000	
4464	011237	012	047523	052106	SOFTBS: .ASCII <12>/SOFTWARE /
4465	011244	040527	042522	040	
4466	011251	102	042101	051440	BDSECT: .ASCIZ /BAD SECTORS :/<15><12>
4467	011256	041505	047524	051522	
4468	011264	035040	005015	000	
4469	011271	040	047040	047117	NOFALS: .ASCIZ / NONE/
4470	011276	000105			
4471					
4472					
4473					
4474	011300	005015	047412	043106	:MESSAGES USED IN OFFSET-TO-FAILURE TEST
4475	011306	042523	026524	047524	OFSHED: .ASCII <15><12><12>/OFFSET-TO-FAILURE MEASUREMENTS :/<15><12><12>
4476	011314	043055	044501	052514	
4477	011322	042522	046440	040505	
4478	011330	052523	042522	042515	
4479	011336	052116	020123	006472	
4480	011344	005012			
4481	011346	051124	041501	020113	.ASCII /TRACK CYLN SECT +OFST -OFST/<15><12>
4482	011354	041440	046131	020116	
4483	011362	051440	041505	020124	
4484	011370	025440	043117	052123	
4485	011376	020040	047455	051506	
4486	011404	006524	012		
4487	011407	011	020011	020040	.ASCIZ <011><011>/ (UIN) (UIN)/<15><12>
4488	011414	052450	047111	020051	
4489	011422	024040	044525	024516	
4490	011430	005015	000		
4491					
4492					
4493	011433	000			UNLOD: .BYTE 0 ;DRIVE UNLOADED INDICATOR
4494	011434	000			VERIFY: .BYTE 0 ;VERIFY MODE FLAG
4495	011435	015	005012	025052	IDENT: .ASCIZ <15><12><12>/*** RK06 HEAD ALIGNMENT AID ***/<15><12>
4496	011442	020052	045522	033060	
4497	011450	044040	040505	020104	
4498	011456	046101	043511	046516	
4499	011464	047105	020124	044501	
4500	011472	020104	025052	006452	
4501	011500	000012			
4502	011502	047506	020122	042510	FORHLP: .ASCIZ /FOR HELP TYPE H, ELSE <CR>/<15><12>
4503	011510	050114	052040	050131	
4504	011516	020105	026110	042440	
4505	011524	051514	020105	041474	
4506	011532	037122	005015	000	
4507	011537	012	040515	052516	TYPMOD: .ASCIZ <12>/MANUAL OR AUTO MODE (M OR A)?/<15><12>
4508	011544	046101	047440	020122	
4509	011552	052501	047524	046440	
4510	011560	042117	020105	046450	
4511	011566	047440	020122	024501	
4512	011574	006477	000012		

4513	011600	005012	020052	040515	TYPMAN: .ASCIZ <12><12> /* MANUAL SELECT MODE */ <15><12>
4514	011606	052516	046101	051440	
4515	011614	046105	041505	020124	
4516	011622	047515	042504	025040	
4517	011630	005015	000		
4518	011633	012	047105	042524	ENTDRV: .ASCIZ <12> / ENTER DRIVE NO. (0-7): / <15><12>
4519	011640	020122	051104	053111	
4520	011646	020105	047516	020056	
4521	011654	030050	033455	035051	
4522	011662	005015	000		
4523	011665	052	020052	042040	NOTRDY: .ASCII /** DRIVE /
4524	011672	044522	042526	040	
4525	011677	040	047040	052117	DRNRDY: .ASCII / NOT READY ! START DRIVE IF NECESSARY. / <15><12>
4526	011704	051040	040505	054504	
4527	011712	020440	020040	052123	
4528	011720	051101	020124	051104	
4529	011726	053111	020105	043111	
4530	011734	047040	041505	051505	
4531	011742	040523	054522	006456	
4532	011750	012			
4533	011751	040	020040	053440	.ASCIZ / WHEN DRIVE IS READY, PRESS "CONT" ON CPU. / <15><12>
4534	011756	042510	020116	051104	
4535	011764	053111	020105	051511	
4536	011772	051040	040505	054504	
4537	012000	020054	051120	051505	
4538	012006	020123	041442	047117	
4539	012014	021124	047440	020116	
4540	012022	050103	027125	005015	
4541	012030	000			
4542	012031	052	020052	042040	NONEXD: .ASCII /** DRIVE /
4543	012036	044522	042526	040	
4544	012043	040	047040	047117	DRVNED: .ASCIZ / NON-EXISTENT OR OFF-LINE ! PRESS "CONT" TO RESTART. / <15><12>
4545	012050	042455	044530	052123	
4546	012056	047105	020124	051117	
4547	012064	047440	043106	046055	
4548	012072	047111	020105	020041	
4549	012100	051120	051505	020123	
4550	012106	041442	047117	021124	
4551	012114	052040	020117	042522	
4552	012122	052123	051101	027124	
4553	012130	005015	000		
4554	012133	040	020040	041440	HALTER: .ASCIZ / CPU WILL HALT. / <15><12>
4555	012140	052520	053440	046111	
4556	012146	020114	040510	052114	
4557	012154	006456	000012		
4558	012160	025052	020040	051104	NOTLOK: .ASCII /** DRIVE /
4559	012166	053111	020105		
4560	012172	020040	047516	020124	NODRLK: .ASCII / NOT WRITE-LOCKED ! PLEASE SET WRITE LOCK SWITCH. / <15><12>
4561	012200	051127	052111	026505	
4562	012206	047514	045503	042105	
4563	012214	020440	050040	042514	
4564	012222	051501	020105	042523	
4565	012230	020124	051127	052111	
4566	012236	020105	047514	045503	
4567	012244	051440	044527	041524	
4568	012252	020110	006456	012	

M07

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 91

SEQ 0090

PROGRAM SPECIFIC RESERVED LOCATIONS

4569	012257	040	020040	052040	.ASCIZ / THEN PRESS "CONT" ON CPU./<15><12>
4570	012264	042510	020116	051120	
4571	012272	051505	020123	041442	
4572	012300	047117	021124	047440	
4573	012306	020116	050103	027125	
4574	012314	005015	000		
4575	012317	052	020052	046440	MULSEL: .ASCIZ /** MULTIPLE DRIVES AUTO-SELECTED ! PRESS 'CONT' TO RESTART./<15><12>
4576	012324	046125	044524	046120	
4577	012332	020105	051104	053111	
4578	012340	051505	040440	052125	
4579	012346	026517	042523	042514	
4580	012354	052103	042105	020440	
4581	012362	050040	042522	051523	
4582	012370	023440	047503	052116	
4583	012376	020047	047524	051040	
4584	012404	051505	040524	052122	
4585	012412	006456	000012		
4586	012416	040412	044514	047107	ENTMOD: .ASCIZ <12>/ALIGN,VERIFY, OR EXERCISE (A,V, OR E) ?/<15><12>
4587	012424	053054	051105	043111	
4588	012432	026131	047440	020122	
4589	012440	054105	051105	044503	
4590	012446	042523	024040	026101	
4591	012454	026126	047440	020122	
4592	012462	024505	037440	005015	
4593	012470	000			
4594	012471	012	025012	051040	MANEXR: .ASCII <12><12>/* RANDOM SEEK EXERCISES IN PROGRESS /
4595	012476	047101	047504	020115	
4596	012504	042523	045505	042440	
4597	012512	042530	041522	051511	
4598	012520	051505	044440	020116	
4599	012526	051120	043517	042522	
4600	012534	051523	040		
4601	012537	117	020116	051104	.ASCII /ON DRIVE /
4602	012544	053111	020105		
4603	012550	020040	006452	000012	DRIEXR: .ASCIZ / */<15><12>
4604	012556	005012	020052	040515	MANALN: .ASCIZ <12><12>/* MANUAL SELECT ALIGNMENT */<15><12>
4605	012564	052516	046101	051440	
4606	012572	046105	041505	020124	
4607	012600	046101	043511	046516	
4608	012606	047105	020124	006452	
4609	012614	000012			
4610	012616	005012	020052	040515	MANVRF: .ASCIZ <12><12>/* MANUAL SELECT VERIFY */<15><12>
4611	012624	052516	046101	051440	
4612	012632	046105	041505	020124	
4613	012640	042526	044522	054506	
4614	012646	025040	005015	000	
4615	012653	110	040505	051504	HEDPOS: .ASCIZ /HEADS POSITIONED AT CYLINDER 365 (OCT)/<15><12>
4616	012660	050040	051517	052111	
4617	012666	047511	042516	020104	
4618	012674	052101	041440	046131	
4619	012702	047111	042504	020122	
4620	012710	033063	020065	047450	
4621	012716	052103	006451	000012	
4622	012724	042412	052116	051105	ENTHED: .ASCIZ <12>/ENTER HEAD NO. (0-2):/<15><12>
4623	012732	044040	040505	020104	
4624	012740	047516	020056	030050	

4625	012746	031055	035051	005015	
4626	012754	000			
4627	012755	124	050131	020105	RWNRDY: .ASCIZ /TYPE <R> WHEN READY :/<15><12>
4628	012762	051074	020076	044127	
4629	012770	047105	051040	040505	
4630	012776	054504	035040	005015	
4631	013004	000			
4632	013005	110	040505	020104	HEDSEL: .ASCII /HEAD /
4633	013012	020040	042523	042514	HEADNO: .ASCIZ / SELECTED/<15><12>
4634	013020	052103	042105	005015	
4635	013026	000			
4636	013027	012	025012	040440	TYPAUT: .ASCIZ <12><12> /* AUTO SELECT MODE */<15><12>
4637	013034	052125	020117	042523	
4638	013042	042514	052103	046440	
4639	013050	042117	020105	006452	
4640	013056	000012			
4641	013060	005012	020052	052501	AUTALN: .ASCIZ <12><12> /* AUTO SELECT ALIGNMENT */<15><12>
4642	013066	047524	051440	046105	
4643	013074	041505	020124	046101	
4644	013102	043511	046516	047105	
4645	013110	020124	006452	000012	
4646	013116	005012	020052	052501	AUTVRF: .ASCIZ <12><12> /* AUTO SELECT VERIFY */<15><12>
4647	013124	047524	051440	046105	
4648	013132	041505	020124	042526	
4649	013140	044522	054506	025040	
4650	013146	005015	000		
4651	013151	012	051104	053111	DRISEL: .ASCII <12>/DRIVE /
4652	013156	020105			
4653	013160	020040	042523	042514	DRVSEL: .ASCIZ / SELECTED/<15><12><12>
4654	013166	052103	042105	005015	
4655	013174	000012			
4656	013176	005012	020052	052501	AUTEXR: .ASCII <12><12> /* AUTO SELECT EXERCISES */<15><12>
4657	013204	047524	051440	046105	
4658	013212	041505	020124	054105	
4659	013220	051105	044503	042523	
4660	013226	020123	006452	012	
4661	013233	012	044123	051117	.ASCIZ <12>/SHORT OR LONG (S OR L) ?/<15><12>
4662	013240	020124	051117	046040	
4663	013246	047117	020107	051450	
4664	013254	047440	020122	024514	
4665	013262	037440	005015	000	
4666	013267	105	042530	041522	AUTOEX: .ASCII /EXERCISING DRIVE /
4667	013274	051511	047111	020107	
4668	013302	051104	053111	020105	
4669	013310	006440	000012		AUTODR: .ASCIZ / /<15><12>
4670					
4671	013314	025052	020040	040503	BAD632: .ASCIZ /** CANNOT READ BAD SECTOR TRACK!/<15><12>
4672	013322	047116	052117	051040	
4673	013330	040505	020104	040502	
4674	013336	020104	042523	052103	
4675	013344	051117	052040	040522	
4676	013352	045503	006441	000012	
4677	013360	041536	005015	000	CNTRLC: .ASCIZ /↑C/<15><12>
4678	013365	136	006532	000012	CNTRLZ: .ASCIZ /↑Z/<15><12>
4679	013372	051136	005015	000	CNTRLR: .ASCIZ /↑R/<15><12>
4680	013377	136	006525	000012	CNTRLU: .ASCIZ /↑U/<15><12>

```

4600 013404 043536 005015 000 CNTRLG: .ASCIZ /!G/<15><12>
4601 013404 013404 005012 000 CR2LF: .ASCIZ <15><12><12>
4602 013404 013404 000000 000 BKLSH: .ASCIZ /\
4603 013404 013404 000000 000 COMMA: .ASCIZ /,
4604 013404 013404 000000 040 SPACE6: .ASCIZ / /
4605 013404 013404 000000 040 SPACE4: .ASCIZ / /
4606 013404 013404 000000 040 SPACE3: .ASCIZ / /
4607 013404 013404 000000 040 SPACE2: .ASCIZ / /
4608 013404 000040 000000 040 SPACE1: .ASCIZ / /
4609 013404 000040 000000 040 .EVEN
4610 013430 020040 000075 000040 PRMBUF: .ASCIZ / = /
4611 013434 047527 042122 000040 WORDSP: .ASCIZ /WORD /
4612 013442 036440 000040 000 EQUALS: .ASCIZ / = /
4613 013446 020052 000000 000 PROMPT: .ASCIZ /* /
4614 013451 076 000040 000 PRMPSP: .ASCIZ /) /
4615 013451 076 000040 000 .EVEN
4700 013454 105037 003106 000 DFSTRT: CLRB MDFLAG ;SET FLAG FOR DEFAULT MODE
4701 013460 105037 003126 000 CLRB DULACS ;CLEAR DUAL-ACCESS FLAG
4702 013464 105037 003116 000 CLRB ERRCNT ;CLEAR ERROR COUNT FOR RESTARTS
4703 013470 022737 014532 000042 CMP #DRVTST, @#42 ;SEE IF EOP RETURN ADRS = DRVTST
4704 013476 001003 000000 000 BNE 4$ ;BR IF NOT DRVTST
4705 013500 012737 017456 000042 MOV #NEWPAS, @#42 ;SET RETURN ADRS = NEWPAS
4706 013506 000405 000000 000 4$: BR CMSTRT ;PROCEED
4707 013510 112737 000001 003106 PSTART: MOV #1, MDFLAG ;SET FLAG FOR PARAMETER MODE
4708 013516 105037 003126 000 CLRB DULACS ;CLEAR DUAL-ACCESS TEST FLAG
4709 013522 012737 000340 177776 CMSTRT: MOV #PR7, @#PS ;BLOCK ALL INTERRUPTS
4710 .SBTTL INITIALIZE THE COMMON TAGS
4711 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
4712 MOV #SCMTAG, R6 ;FIRST LOCATION TO BE CLEARED
4713 CLR (R6)+ ;CLEAR MEMORY LOCATION
4714 CMP #SWR, R6 ;;DONE?
4715 BNE .-6 ;LOOP BACK IF NO
4716 MOV #STACK, SP ;SETUP THE STACK POINTER
4717 ;;INITIALIZE A FEW VECTORS
4718 MOV #SCOPE, @#IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
4719 MOV #340, @#IOTVEC+2 ;LEVEL 7
4720 MOV #ERROR, @#EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
4721 MOV #340, @#EMTVEC+2 ;LEVEL 7
4722 MOV #STRAP, @#TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
4723 MOV #340, @#TRAPVEC+2 ;LEVEL 7
4724 MOV #SPWRDN, @#PWRVEC ;POWER FAILURE VECTOR
4725 MOV #340, @#PWRVEC+2 ;LEVEL 7
4726 MOV $ENDCT, $EOPCT ;SETUP END-OF-PROGRAM COUNTER
4727 CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
4728 CLR $ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS
4729 MOV #1, $ERMAX ;ALLOW ONE ERROR PER TEST
4730 MOV #., $LPADR ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
4731 MOV #., $LPERR ;SETUP THE ERROR LOOP ADDRESS

```

```

4737      :: SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
4738      :: EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
4739 013670 013746 000004      MOV      @#ERRVEC, -(SP)      ;; SAVE ERROR VECTOR
4740 013674 012737 013730 000004  MOV      #64$, @#ERRVEC      ;; SET UP ERROR VECTOR
4741 013702 012737 177570 001140  MOV      #DSWR, SWR          ;; SETUP FOR A HARDWARE SWICH REGISTER
4742 013710 012737 177570 001142  MOV      #DDISP, DISPLAY     ;; AND A HARDWARE DISPLAY REGISTER
4743 013716 022777 177777 165214  CMP      #-1, @SWR          ;; TRY TO REFERENCE HARDWARE SWR
4744 013724 001012      BNE      66$                ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
4745      ;; AND THE HARDWARE SWR IS NOT = -1
4746 013726 000403      BR      65$                ;; BRANCH IF NO TIMEOUT
4747 013730 012716 013736      64$:  MOV      #65$, (SP)        ;; SET UP FOR TRAP RETURN
4748 013734 000002      RTI
4749 013736 012737 000176 001140  65$:  MOV      #SWREG, SWR        ;; POINT TO SOFTWARE SWR
4750 013744 012737 000174 001142  MOV      #DISPREG, DISPLAY   ;;
4751 013752 012637 000004  66$:  MOV      (SP)+, @#ERRVEC     ;; RESTORE ERROR VECTOR
4752 013756 005037 001326      CLR      $PASS              ;; CLEAR PASS COUNT
4753 013762 132737 000200 001341  BITB    #APTSIZE, $ENVM     ;; TEST USER SIZE UNDER APT
4754 013770 001403      BEQ      67$                ;; YES, USE NON-APT SWITCH
4755 013772 012737 001342 001140  67$:  MOV      #SSWREG, SWR      ;; NO, USE APT SWITCH REGISTER
4756 014000      RESET                      ;; CLEAR THE UNIBUS
4757 014000 000005      MOV      #6, @#ERRVEC      ;; SET TIME-OUT VECTOR
4758 014002 012737 000006 000004  CLR      @#ERRVEC+2
4759 014010 005037 000006      MOV      #MPERHD, @#MEMVEC  ;; SET MEM PARITY TRAP VECTOR
4760 014014 012737 030114 000114  MOV      #PR7, @#MEMVEC+2   ;; SET TRAP PRIORITY = 7
4761 014022 012737 000340 000116  JSR      PC, ENBCSR         ;; ENABLE MEMORY PARITY CHECK
4762 014030 004737 030030      MOV      #100, $ERMAX      ;; SET MAX ERROR CNT TO 100 FOR $SCOPE
4763 014034 112737 000144 001115  CLRB    DRVERS              ;; CLEAR ERROR COUNT FOR CURRENT DRIVE
4764 014042 105037 003120      MOV      #1, HLPOVL        ;; SET HLP FILE OVLD INDICTR
4765 014046 112737 000001 003136  TYPE    , DZR6N            ;; TYPE PROGRAM I.D. FOR PART 2
4766 014054 104401 007024      TYPE    , SUBVER.
4767 014060 104401 007044      TYPE    , PART2
4768 014064 104401 007125      TSTB    NOTYPE            ;; SEE IF OPERATOR NOTE SHOULD BE TYPED
4769 014070 105737 003137      BNE      39$                ;; BR IF NOT
4770 014074 001004      TYPE    , NOTMSG          ;; TYPE OPERATOR NOTE
4771 014076 104401 064640      INCB    NOTYPE            ;; SET FLAG FOR NEXT TIME
4772 014102 105237 003137      TSTB    DULACS            ;; SEE IF DUAL-ACCESS DATA TEST
4773 014106 105737 003126  39$:  BEQ      40$                ;; BR IF NOT
4774 014112 001402      TYPE    , DUACES          ;; TYPE "* DUAL-ACCESS DATA TEST *"
4775 014114 104401 010417      MOV      #KBDHDL, @#TKVEC  ;; LOAD VECTOR FOR TTY KBD
4776 014120 012737 027142 000060  40$:  MOV      #PR4, @#TKVEC+2   ;; SET KBD PRIORITY = 4
4777 014126 012737 000200 000062  MOV      RKVEC, R1          ;; ADDR. OF RK06 VECTOR STORAGE
4778 014134 013701 003030      MOV      #I, INTR, (R1)+   ;; SET IT TO RK06 HANDLER
4779 014140 012721 045620      MOV      RKPRI, (R1)       ;; SET RK06 PRIORITY
4780 014144 013711 003032      MOV      #KTERHD, @#MMVEC  ;; VECT FOR KT11 FAILURE
4781 014150 012737 027734 000250  MOV      #PR7, @#MMVEC+2   ;; SET PRIOR. = 7 FOR HNDLER
4782 014156 012737 000340 000252  CLRB    TSTING             ;; CLEAR "RUNNING TESTS" FLAG
4783 014164 105037 003110      MOV      #PRO, @#PS        ;; ALLOW ALL INTERRUPTS AGAIN
4784 014170 012737 000000 177776  MOV      #PRVCM, R1        ;; ZERO OUT PREVIOUS COMMAND
4785 014176 012701 005222      MOV      #6, RO            ;; SET COUNTER
4786 014202 012700 000006  42$:  CLR      (R1)+            ;; ZERO A WORD
4787 014206 005021      DEC     RO
4788 014210 005300      BNE     42$                ;; BR IF NOT DONE YET
4789 014212 001375      CLR     STALLS            ;; DON'T ALLOW STALLS YET
4790 014214 005037 005502      JSR     PC, GTSWRG        ;; OPEN SOFTWARE SWR FOR MODIFICATION
4791 014220 004737 027614

```

```

4793 014224 012737 176543 053246 44$: MOV #176543,$HINUM ;INIT. PSEUDO-RANDOM NOS.
4794 014232 012737 123456 053250 MOV #123456,$LONUM
4795 014240 004737 027334 JSR PC,SIZMEM ;SIZE MEMORY, FIX MA LIMIT IN PRMLIM
4796 014244 105737 003125 TSTB XDPSVD ;SEE IF XXDP PREVIOUSLY SAVED
4797 014250 001404 BEQ 9$ ;BR IF NOT
4798 014252 004737 030536 JSR PC,GETXDP ;RESTORE SAVED XXDP
4799 014256 105037 003125 CLRB XDPSVD ;CLEAR THE FLAG
4800 014262 105737 003106 9$: TSTB MDFLAG ;SEE IF DEFAULT MODE
4801 014266 001031 BNE 10$ ;BR IF NOT DEFAULT MODE
4802 014270 013700 001370 MOV $VECT1,RO ;GET APT RK06 VECTOR AND PRTY
4803 014274 013737 001374 003026 MOV $BASE,RKBAS ;GET APT RK06 BASE ADDRESS
4804 014302 132737 000200 001341 BITB #BIT7,$ENVM ;SEE IF NO SIZING
4805 014310 001005 BNE 12$ ;BR IF NO SIZING
4806 014312 012700 120210 MOV #AVECT1,RO ;GET DEFAULT VECTOR AND PRIORITY
4807 014316 012737 177440 003026 MOV #ABASE,RKBAS ;GET DEFAULT BASE ADDRESS
4808 014324 110037 003030 18$: MOVB RO,RKVEC ;STORE VECTOR
4809 014330 105037 003031 CLRB RKVEC+1 ;CLEAR HI BYTE
4810 014334 000300 SWAB RO ;GET PRTY INTO BITS 5-7
4811 014336 042700 177437 BIC #177437,RO ;CLEAR OTHER BITS
4812 014342 010037 003032 MOV RO,RKPRI ;STORE RK06 PRIORITY
4813 014346 000137 015250 JMP ALLDRV ;GO CHECK ALL DRIVES
4814
4815
4816
4817 ;BEGIN PARAMETER INPUT MODE
4818 014352 104401 007216 10$: TYPE ,PRMINP ;TYPE "PARAMETER INPUT MODE"
4819
4820 ;OPEN RK06 UNIBUS ADDRESS FOR MODIFICATION
4821 014356 013746 003026 MOV RKBAS,-(SP) ;PUT OLD VALUE ON STACK
4822 014362 104401 007250 TYPE RKBADR ;TYPE "RK06 BUS ADR = "
4823 014366 004737 027504 JSR PC,GETPRM ;TYPE OLD, GET NEW RKBAS VALUE
4824 014372 012637 003026 MOV (SP)+,RKBAS ;STORE NEW VALUE
4825 ;OPEN RK06 VECTOR ADDRESS FOR MODIFICATION
4826 014376 013746 003030 MOV RKVEC,-(SP) ;PUT OLD VALUE ON STACK
4827 014402 104401 007270 TYPE RKVADR ;TYPE "RK06 VEC ADR = "
4828 014406 004737 027504 JSR PC,GETPRM ;TYPE OLD, GET NEW RKVEC VALUE
4829 014412 012637 003030 MOV (SP)+,RKVEC ;STORE NEW VALUE
4830 ;OPEN RK06 INTERRUPT HANDLER PRIORITY LEVEL FOR MODIFICATION
4831 014416 013746 003032 11$: MOV RKPRI,-(SP) ;GET OLD VALUE OF PRIORITY
4832 014422 006316 ASL (SP) ;GET IT INTO BITS 0-2
4833 014424 006316 ASL (SP)
4834 014426 006316 ASL (SP)
4835 014430 000316 SWAB (SP)
4836 014432 104401 007310 TYPE RKPRTY ;TYPE "RK06 PRIORITY = "
4837 014436 004737 027504 JSR PC,GETPRM ;TYPE OLD, GET NEW RKPRI VALUE
4838 014442 012600 MOV (SP)+,RO
4839 014444 020027 000004 CMP RO,#4 ;SEE IF AT LEAST LEVEL 4
4840 014450 002414 BLT 12$ ;ER IF NEW VALUE TOO SMALL
4841 014452 020027 000007 CMP RO,#7 ;SEE IF LEVEL 7 OR LESS
4842 014456 003011 BGT 12$ ;BR IF NEW VALUE TOO LARGE
4843 014460 000300 SWAB RO ;GET PRIORITY INTO BITS 5-7
4844 014462 006200 ASR RO
4845 014464 006200 ASR RO
4846 014466 006200 ASR RO
4847 014470 010037 003032 MOV RO,RKPRI ;STORE NEW VALUE
4848 014474 104401 001315 TYPE ,SCLF

```


4905	014700	001354				BNE	4\$:BR IF NOT DONE
4906	014702	005700				TST	RO		:SEE IF ANY DRIVES WERE LISTED
4907	014704	001016				BNE	26\$:BR IF YES
4908	014706	104401	011271			TYPE	,NOFALS		:TYPE " NONE"
4909	014712	104401	007571			TYPE	,NODRTS		:TYPE "** NO DRIVES TO TEST"
4910									: "PRESS 'CONT' WHEN RDY"
4911	014716	012703	000401			MOV	#401,R3		:PATTERN TO MARK DRIVES
4912	014722	012701	005606			MOV	#DRVLST,R1		:DRIVE LIST ADDRESS
4913	014726	010321				MOV	R3,(R1)+		:MARK ALL DRIVES IN LIST
4914	014730	010321				MOV	R3,(R1)+		
4915	014732	010321				MOV	R3,(R1)+		
4916	014734	010311				MOV	R3,(R1)		
4917	014736	000137	044476			JMP	HLTPRG		:HALT PROGRAM
4918	014742	104401	001315	26\$:		TYPE	,\$CRLF		:TYPE <CR>,<LF>
4919	014746	105737	003106			TSTB	MDFLAG		:SEE IF DEFAULT MODE
4920	014752	001002				BNE	20\$:BR IF NOT DEFAULT MODE
4921	014754	000137	015674			JMP	LODFLS		:GO LOAD DEFLT ITER. COUNTS
4922	014760	122737	000013	000041	20\$:	CMPB	#13,#41		:SEE IF RK06 IS XXDP MEDIUM
4923	014766	001032				BNE	19\$:BR IF NOT
4924	014770	105737	005606			TSTB	DRVLST		:SEE IF DRIVE 0 LISTED TO TEST
4925	014774	001427				BEQ	19\$:BR IF NOT
4926	014776	104401	007502	24\$:		TYPE	,REPLPK		:TYPE "IF XXDP PACK ON DRV 0, TYPE Y <CR>
4927									: AND REPLACE IT"
4928	015002	004737	032272			JSR	PC,RDCHRS		:READ RESPONSE
4929	015006	014776				24\$:(<C> RETURN
4930	015010	014776				24\$:(<Z> RETURN
4931	015012	014776				24\$:(<U> RETURN
4932	015014	005700				TST	RO		:SEE IF NULL INPUT
4933	015016	001416				BEQ	19\$:BR IF JUST <CR> TYPED
4934	015020	022737	000131	005262		CMP	#'Y,BUFF0		:SEE IF "Y <CR>" TYPED
4935	015026	001363				BNE	24\$:BR IF NOT, TO ASK AGAIN
4936	015030	104401	007622			TYPE	,CNTRDY		:TYPE "PRESS CONT WHEN DRV RDY"
4937	015034	042762	000100	000000		BIC	#IE,RKCS1(R2)		:DISABLE RK06 INTERRUPT
4938	015042	000000				HALT			:HALT FOR PACK CHANGE
4939	015044	004737	030776			JSR	PC,INITSS		:INIT THE SUB-SYS
4940	015050	000137	015342			JMP	CHKLST		:GO CHECK STATUS OF LISTED DRIVES
4941	015054	104401	013446	19\$:		TYPE	,PROMPT		:TYPE ASTERISK AND SPACE
4942									:READ AND CHECK NEW DRIVE NUMBERS
4943	015060	004737	032272			JSR	PC,RDCHRS		:READ IN INPUT STRING
4944	015064	014532				DRVTST			:(<C> RETURN ADDRESS
4945	015066	014532				DRVTST			:(<Z> RETURN ADDRESS
4946	015070	014616				TELDRV			:(<U> OR ERROR RETURN ADDRESS
4947	015072	005700				TST	RO		:SEE IF NULL INPUT
4948	015074	001007				BNE	9\$:BR IF NEW DRIVES WILL BE SELECTED
4949	015076	105737	003126			TSTB	DULACS		:SEE IF DUAL-ACCESS FLAG SET
4950	015102	001002				BNE	22\$:BR IF YES
4951	015104	000137	015406			JMP	ASKTST		:JUMP TO THE TEST INPUT ROUTINE
4952	015110	000137	016152	22\$:		JMP	INPUTP		:GO ASK FOR INPUT PARAMETERS
4953	015114	022700	000017	9\$:		CMP	#15.,RO		:SEE IF TOO MANY CHARACTERS TYPED
4954	015120	002005				BGE	12\$:BR IF NOT TOO MANY
4955	015122	104401	005262	10\$:		TYPE	,BUFF0		:ECHO BAD INPUT
4956	015126	104401	001314			TYPE	,\$QUES		:TYPE <?> AND <CR>,<LF>
4957	015132	000631				BR	TELDRV		:GO TYPE DRIVES AND ASK AGAIN
4958	015134	005001			12\$:	CLR	R1		:CLEAR CHAR. POINTER
4959	015136	126127	005262	000060	14\$:	CMPB	BUFF0(R1),#'0		:SEE IF DRIVE NO. < 0
4960	015144	002766				BLT	10\$:BR TO ECHO BAD INPUT

```

4961 015146 126127 005262 000067      CMPB   BUFF0(R1),#7      ;SEE IF > 7
4962 015154 003362      BGT    10$              ;BR TO ECHO BAD INPUT
4963 015156 005201      INC    R1               ;INCREMENT CHAR POINTER
4964 015160 020100      CMP    R1,RO            ;SEE IF MORE CHARS TO CHECK
4965 015162 001406      BEQ    16$              ;BR IF ALL CHARS CHECKED
4966 015164 126127 005262 000054      CMPB   BUFF0(R1),#',    ;SEE IF THIS IS COMMA
4967 015172 001353      BNE    10$              ;BR IF NOT COMMA
4968 015174 005201      INC    R1               ;INCREMENT CHAR POINTER
4969 015176 000757      BR     14$              ;BR TO CONTINUE CHECKING CHARS
4970                                     ;GET NEW DRIVE NUMBERS INTO LIST
4971 015200 012701 005606      16$:  MOV    #DRVLST,R1    ;GET DRIVE LIST ADDRESS
4972 015204 005021      CLR    (R1)+            ;CLEAR OUT THE DRIVE LIST
4973 015206 005021      CLR    (R1)+
4974 015210 005021      CLR    (R1)+
4975 015212 005011      CLR    (R1)
4976 015214 005000      CLR    RO               ;CLEAR BUFFER POINTER
4977 015216 116001 005262      18$:  MOVB   BUFF0(RO),R1 ;GET A DRIVE NUMBER
4978 015222 042701 000060      BIC    #'0,R1           ;STRIP ASCII BITS
4979 015226 112761 000001 005606      MOVB   #1,DRVLST(R1)    ;MARK THIS DRIVE IN DRIVE LIST
4980 015234 062700 000002      ADD    #2,RO            ;POINT TO NEXT DRIVE NUMBER
4981 015240 105760 005261      TSTB   BUFF0-1(RO)     ;SEE IF NULL CHAR YET
4982 015244 001364      BNE    18$              ;BR IF NOT DONE YET
4983 015246 000435      BR     CHKLST           ;GO CHECK NEW DRIVES FOR VALID STATUS
4984                                     ;COME HERE IF ALL DRIVES REQUESTED
4985 015250 005000      ALLDRV: CLR   RO         ;CLEAR DRIVE NUMBER
4986 015252 013703 001376      MOV    $DEVN,R3         ;GET APT DEVICE MAP
4987 015256 132737 000200 001341      BITB   #BIT7,$ENVM     ;SEE IF NO SIZING
4988 015264 001002      BNE    1$               ;BR IF NO SIZING
4989 015266 012703 000377      MOV    #377,R3         ;SET UP FOR SIZING
4990 015272 012701 000001      1$:   MOV    #BIT0,R1     ;SET BIT POINTER
4991 015276 105060 005606      2$:   CLRB   DRVLST(RO)  ;INITIALIZE DRIVE ENTRY
4992 015302 030103      BIT    R1,R3           ;SEE IF THIS DRIVE IS REQUESTED
4993 015304 001403      BEQ    4$               ;BR IF NOT
4994 015306 112760 000001 005606      MOVB   #1,DRVLST(RO)   ;MARK THIS DRIVE IN DRIVE LIST
4995 015314 006301      4$:   ASL    R1           ;SHIFT BIT POINTER
4996 015316 005200      INC    RO               ;INCREMENT DRIVE NUMBER
4997 015320 022700 000010      CMP    #10,RO          ;SEE IF DONE MARKING ALL DRIVES
4998 015324 001364      BNE    2$               ;BR IF NOT DONE YET
4999 015326 132737 000200 001341      BITB   #BIT7,$ENVM     ;SEE IF PROGRAM SHOULD SIZE
5000 015334 001402      BEQ    CHKLST           ;BR IF YES
5001 015336 000137 014616      JMP    TELDRV           ;GO LIST APT-SELECTED DRIVES
5002                                     ;CHECK ALL DRIVES MARKED IN DRIVE LIST FOR PROPER STATUS
5003 015342 005000      CHKLST: CLR   RO        ;CLEAR DRIVE NUMBER
5004 015344 105760 005606      2$:   TSTB   DRVLST(RO)  ;SEE IF THIS DRIVE IS MARKED IN LIST
5005 015350 001405      BEQ    4$               ;BR IF NOT MARKED
5006 015352 010037 005500      MOV    RO,DRIVE        ;SET DRIVE NUMBER FOR SCNDRV
5007 015356 004737 031116      JSR    PC,SCNDRV       ;CHECK STATUS OF THIS DRIVE
5008                                     ;IF NOT VALID, TYPE MESSAGE
5009                                     ;AND REMOVE IT FROM DRIVE LIST
5010 015362 015400      4$:   6$:   INC    RO           ;ERROR RETURN ADDRESS FOR SCNDRV
5011 015364 005200      INC    RO               ;RETURN HERE IF DRIVE IS USEABLE
5012 015366 022700 000010      CMP    #10,RO          ;SEE IF DONE CHECKING LIST
5013 015372 001364      BNE    2$               ;BR IF NOT DONE YET
5014 015374 000137 014616      JMP    TELDRV           ;GO BACK TO LIST SELECTED DRIVES
5015 015400 105060 005606      6$:   CLRB   DRVLST(RO)  ;REMOVE INVALID DRIVE FROM LIST
5016 015404 000767      BR     4$               ;CONTINUE CHECKING THE LIST

```

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
5051
5052
5053
5054
5055
5056
5057
5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072

015406 104401 010007
015412 104401 010111
015416 004737 032272
015422 014532
015424 015412
015426 015412
015430 005700
015432 001005
015434 104401 005262
015440 104401 001314
015444 000762
015446 022737 000114 005262
015454 001002
015456 000137 015512
015462 022737 000103 005262
015470 001002
015472 000137 015630
015476 022737 000111 005262
015504 001353
015506 000137 016152

015512 104401 010213
015516 005001
015520 004737 027302
015524 004737 032604
015530 104401 001315
015534 005737 005522
015540 001416
015542 122737 000003 005522
015550 001002
015552 000137 014532
015556 122737 000032 005522
015564 001002
015566 000137 015412
015572 004737 027322
015576 062701 000002

```
*****  
:SBTTL TO - DESIRED TEST INPUT ROUTINE  
:THIS ROUTINE ALLOWS INPUT OF DESIRED TESTS AND ITERATION  
:COUNT FOR EACH TEST. THE INPUT IS CHECKED FOR VALIDITY,  
:AND IF VALID, THE ITERATION COUNTS ARE LOADED INTO THE  
:TEST LIST (TSTLIST). AN ITERATION VALUE OF 0 INDICATES  
:THAT THE TEST IS NOT TO BE RUN.  
:THIS ROUTINE REQUESTS "ENTER L,C, OR I". TYPING (L)  
:CAUSES THE CURRENT LIST OF TESTS AND ITERATIONS TO BE  
:TYPED. (C) ALLOWS TESTS AND ITERATIONS TO BE CHANGED,  
:AND (I) ALLOWS THE OPERATOR TO PROCEED WITH SELECTION  
:OF OPERATING PARAMETERS, AND RUN TESTS.  
:TYPING (↑C) CAUSES IMMEDIATE RETURN TO DRVTST. TYPING  
:* (↑Z) CAUSES RETURN TO ASKTMD.  
:*****
```

```
:DETERMINE L,C, OR I MODE  
ASKTST: TYPE ,TSTMS ;ASK FOR TEST INPUT MODE  
ASKTMD: TYPE ,ENTLCI ;TYPE "ENTER L,C, OR I"  
JSR PC,RDCHRS ;READ RESPONSE  
DRVTST ;(↑C) RETURN ADDRESS  
ASKTMD ;(↑Z) RETURN ADDRESS  
ASKTMD ;(↑U) OR ERROR RETURN ADDRESS  
TST R0 ;SEE IF ANY INPUT  
BNE 4$ ;BR IF ANY INPUT  
2$: TYPE ,BUFFO ;ECHO BAD INPUT  
TYPE ,SQUES ;TYPE <?> AND <CR>,<LF>  
BR ASKTMD ;GO ASK AGAIN  
4$: CMP #'L,BUFFO ;SEE IF (L) TYPED  
BNE 6$ ;BR IF NOT (L)  
JMP LSTTST ;JUMP TO LIST TESTS  
6$: CMP #'C,BUFFO ;SEE IF (C) TYPED  
BNE 8$ ;BR IF NOT (C)  
JMP CHGTST ;JUMP TO CHANGE TESTS  
8$: CMP #'I,BUFFO ;SEE IF (I) TYPED  
BNE 2$ ;BR IF NOT (I), TO ECHO BAD INPUT  
JMP INPUTP ;GO INPUT PARAMS AND RUN TESTS
```

```
:LIST CURRENT TESTS AND ITERATION COUNTS  
LSTTST: TYPE ,TLSTHD ;TYPE HEADING "TEST ITERATIONS"  
CLR R1 ;INITIALIZE TEST INDEX  
2$: JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT  
JSR PC,TYPTST ;TYPE CURRENT TEST AND ITERATION NUMBER  
TYPE ,SCRLF ;TYPE <CR>,<LF>  
TST INTCHR ;SEE IF ANY INPUT  
BEQ 8$ ;BR IF NO INPUT  
CMPB #003,INTCHR ;SEE IF (↑C) TYPED  
BNE 4$ ;BR IF NOT (↑C)  
JMP DRVTST ;JUMP TO ASK FOR DRIVES AGAIN  
4$: CMPB #032,INTCHR ;SEE IF (↑Z) TYPED  
BNE 6$ ;BR IF NOT (↑Z)  
JMP ASKTMD ;JUMP TO ASK FOR NEW TEST INPUT MODE  
6$: JSR PC,ECOBAD ;ECHO BAD INPUT  
8$: ADD #2,R1 ;INCREMENT TEST INDEX
```

```

5073 015602 010102          MOV      R1,R2          ;GET COPY OF INDEX
5074 015604 062702 005616  ADD      #TSTLST,R2     ;GET POSITION IN LIST
5075 015610 022702 005632  CMP      #DFLTST,R2     ;SEE IF DONE WITH LIST
5076 015614 001341          BNE      2$             ;BR IF NOT DONE YET
5077 015616 042777 000100 163320  BIC      #BIT6,0$TKS    ;DISABLE KBD INTERRUPT
5078 015624 000137 015412  JMP      ASKTMD         ;GO ASK FOR NEW TEST INPUT MODE
5079
5080          ;CHANGE CURRENT TESTS AND ITERATION COUNTS
5081 015630 104401 010137  CHGTST: TYPE      ,DFTEST ;ASK IF TESTS SHOULD BE DEFAULTED
5082 015634 004737 032272  JSR      PC,RDCHRS     ;READ RESPONSE
5083 015640 014532          DRVTST                ;(+C) RETURN ADDRESS
5084 015642 015412          ASKTMD                ;(+Z) RETURN ADDRESS
5085 015644 015630          CHGTST                ;(+U) OR ERROR RETURN ADDRESS
5086 015646 005700          TST      RO           ;SEE IF NULL INPUT
5087 015650 001426          BEQ      NULINP       ;BR IF NULL INPUT
5088 015652 022737 000104 005262  CMP      #'D,BUFFO     ;SEE IF (D) TYPED
5089 015660 001405          BEQ      LODFLS       ;BR IF DEFAULTS REQUESTED
5090 015662 104401 005262  TYPE      ,BUFFO       ;ECHO BAD INPUT
5091 015666 104401 001314  TYPE      ,SQUES       ;TYPE (<?) AND (<CR>,<LF>)
5092 015672 000756          BR       CHGTST       ;GO ASK AGAIN
5093          ;LOAD DEFAULT ITERATION COUNTS INTO TEST LIST
5094 015674 012700 005632  LODFLS: MOV      #DFLTST,RO ;LOAD DEFAULT LIST ADDRESS
5095 015700 012702 005616  MOV      #TSTLST,R2    ;LOAD TEST LIST ADDRESS
5096 015704 012022 4$:      MOV      (RO)+,(R2)+ ;LOAD A DEFAULT VALUE
5097 015706 022702 005632  CMP      #DFLTST,R2    ;SEE IF DONE YET
5098 015712 001374          BNE      4$           ;BR IF NOT DONE YET
5099 015714 105737 003106  TSTB     MDFLAG        ;SEE IF DEFAULT MODE
5100 015720 001234          BNE      ASKTMD       ;BR IF NOT DEFAULT MODE
5101 015722 000137 016706  JMP      LODFPT        ;GO LOAD DEFAULT PARAMETERS
5102 015726 005001          NULINP: CLR      R1    ;INITIALIZE TEST INDEX
5103 015730 104401 010213  TYPE      ,TLSTHD     ;TYPE HEADING "TEST ITERATIONS"
5104          ;TYPE CURRENT TEST AND ITERATION NUMBER
5105 015734 004737 032604  B$:      JSR      PC,TYPTST ;TYPE A TEST NO. AND ITERATION NO.
5106 015740 104401 013426  TYPE      ,SPACE1     ;TYPE A SPACE
5107 015744 104401 013446  TYPE      ,PROMPT     ;TYPE ASTERISK AND SPACE
5108          ;READ AND CHECK INPUT IF ANY
5109 015750 004737 032272  JSR      PC,RDCHRS    ;READ OCTAL DIGITS TYPED
5110 015754 014532          DRVTST                ;(+C) RETURN ADDRESS
5111 015756 015412          ASKTMD                ;(+Z) RETURN ADDRESS
5112 015760 015734          B$:      ;(+U) OR ERROR RETURN ADDRESS
5113 015762 005700          TST      RO           ;SEE IF ANY INPUT
5114 015764 001012          BNE      12$          ;BR IF ANY INPUT
5115 015766 062701 000002  10$:     ADD      #2,R1        ;INCREMENT INDEX
5116 015772 010102          MOV      R1,R2        ;COPY INDEX
5117 015774 062702 005616  ADD      #TSTLST,R2    ;GET POSITION IN LIST
5118 016000 022702 005632  CMP      #DFLTST,R2    ;SEE IF DONE WITH LIST
5119 016004 001353          BNE      8$           ;BR IF NOT DONE YET
5120 016006 000137 015412  JMP      ASKTMD        ;GO ASK FOR NEW TEST INPUT MODE
5121 016012 022737 000041 005262  12$:     CMP      #'!,BUFFO     ;SEE IF (!) TYPED
5122 016020 001431          BEQ      16$          ;BR TO PROPAGATE CURRENT ITER. NO.
5123 016022 005002          CLR      R2           ;INITIALIZE (!) INDICATOR
5124 016024 122760 000041 005261  CMPB     #'!,BUFFO-1(RO) ;SEE IF LAST CHAR IN BUF IS (!)
5125 016032 001004          BNE      14$          ;BR IF NOT (!)
5126 016034 105060 005261  CLRB     BUFFO-1(RO)   ;INSERT TERMINATOR BYTE
5127 016040 005300          DEC      RO           ;DECREMENT CHAR COUNT
5128 016042 005202          INC      R2           ;SET (!) INDICATOR

```

```

S129 016044 022700 000005      14$:  CMP      #5,R0      ;SEE HOW MANY CHARS NOW
S130 016050 002433              BLT      20$      ;BR IF TOO MANY (MAX ITER. NO. = 77776)
S131 016052 012746 005262      MOV      #BUFFO,-(SP) ;GET BUF ADDR. ON STACK FOR OCTBIN
S132 016056 004737 053014      JSR      PC,OCTBIN  ;CHECK DIGITS AND CONVERT TO BINARY
S133 016062 016140              JSR      20$      ;ERROR RETURN ADDRESS FOR OCTBIN
S134 016064 012600              MOV      (SP)+,R0   ;GET ITERATION NUMBER
S135 016066 020027 077776      CMP      R0,#77776 ;SEE IF > 77776
S136 016072 101022              BHI      20$      ;BR IF TOO BIG
S137 016074 010061 005616      MOV      R0,TSTLST(R1) ;PUT ITERATION NUMBER INTO TEST LIST
S138 016100 005702              TST      R2        ;SEE IF (!) WAS TYPED
S139 016102 001731              BEQ      10$      ;BR IF NOT (!)
S140                                ;PROPAGATE CURRENT ITERATION NO. TO THE END OF TSTLST
S141 016104 010102      16$:  MOV      R1,R2      ;COPY INDEX
S142 016106 062702 005616      ADD      #TSTLST,R2 ;GET POSITION IN LIST
S143 016112 022702 005630      CMP      #DFLTST-2,R2 ;SEE IF AT END OF LIST
S144 016116 001002              BNE      17$      ;BR IF NOT DONE
S145 016120 000137 015412      JMP      ASKTMD    ;GO ASK FOR NEW TEST INPUT MODE
S146 016124 016161 005616 005620 17$:  MOV      TSTLST(R1),TSTLST+2(R1) ;PROPAGATE TO NEXT WORD
S147 016132 062701 000002      ADD      #2,R1     ;INCREMENT INDEX
S148 016136 000762              BR       16$      ;BR TO CONTINUE
S149 016140 104401 005262      20$:  TYPE      ,BUFFO   ;ECHO BAD INPUT
S150 016144 104401 001314      TYPE      ,SQUES   ;TYPE <?> AND <CR>,<LF>
S151 016150 000671              BR       8$       ;GO ASK AGAIN
S152                                ;ASK FOR DESIRED OPERATIONAL PARAMETER INPUT MODE
S153                                INPUTP: TYPE      EXPLAN ;EXPLAIN INPUT MODES
S154 016152 104401 010243      ASKMDE: CLR      STALLS ;INHIBIT STALL BETWEEN OPERATIONS
S155 016156 005037 005502      JSR      PC,ENBCSR ;ENABLE MEMORY PARITY CHECK
S156 016162 004737 030030      TYPE      ,PARMDE  ;ASK FOR DESIRED INPUT MODE
S157 016166 104401 010371      TYPE      ,PROMPT  ;TYPE "*"
S158 016172 104401 013446      JSR      PC,RDCHRS ;READ RESPONSE TO MODE QUESTION
S159 016176 004737 032272      DRVTST              ;(↑C) RETURN ADDRESS
S160 016202 014532      ASKMDE              ;(↑Z) RETURN ADDRESS
S161 016204 016156      ASKMDE              ;(↑U) OR ERROR RETURN ADDRESS
S162 016206 016156      TST      R0        ;SEE IF NULL INPUT
S163 016210 005700      BNE      4$       ;BR IF ANY INPUT
S164 016212 001005      2$:  TYPE      ,BUFFO   ;ECHO BAD INPUT
S165 016214 104401 005262      TYPE      ,SQUES   ;TYPE <?> AND <CR>,<LF>
S166 016220 104401 001314      BR       ASKMDE   ;GO ASK AGAIN
S167 016224 000754              CMP      #'T,BUFFO ;SEE IF (T) TYPED
S168 016226 022737 000124 005262 4$:  BNE      6$       ;BR IF NOT (T)
S169 016234 001002      JMP      TYPLST   ;JUMP TO THE TYPE LIST ROUTINE
S170 016236 000137 016370      CMP      #'O,BUFFO ;SEE IF (O) TYPED
S171 016242 022737 000117 005262 6$:  BNE      8$       ;BR IF NOT (O)
S172 016250 001002      JMP      OPNLST   ;JUMP TO THE OPEN LIST ROUTINE
S173 016252 000137 016642      CMP      #'S,BUFFO ;SEE IF (S) TYPED
S174 016256 022737 000123 005262 8$:  BNE      10$      ;BR IF NOT (S)
S175 016264 001002      JMP      SETPRM   ;JUMP TO THE SET INDIV. PARAM. ROUTINE
S176 016266 000137 017126      CMP      #'R,BUFFO ;SEE IF (R) TYPED
S177 016272 022737 000122 005262 10$: BNE      2$       ;BR IF NOT (R), TO ECHO BAD INPUT
S178 016300 001345      CMP      S0,S1    ;COMPARE 20(DEC) SECTOR LIMITS
S179 016302 023737 005656 005660 BLE      12$      ;BR IF S0 NOT > S1
S180 016310 003403      TYPE      ,SECNL1 ;TYPE "S0>S1 NOT ALLOWED"
S181 016312 104401 010477      BR       ASKMDE   ;GO ASK AGAIN FOR PARAMETERS
S182 016316 000717      CMP      S2,S3    ;COMPARE 22(DEC) SECTOR LIMITS
S183 016320 023737 005662 005664 12$: BLE      14$      ;BR IF S2 NOT > S3
S184 016326 003405

```

5185	016330	104401	010527			TYPE	,SECNL2	:TYPE "S2>S3"
5186	016334	104401	010510			TYPE	,NOTALD	:TYPE "NOT ALLOWED"
5187	016340	000706				BR	ASKMDE	:GO ASK AGAIN FOR PARAMETERS
5188	016342	023737	005652	005654	14\$:	CMP	FT,LT	:COMPARE CHOSEN TRACK LIMITS
5189	016350	003405				BLE	20\$:BR IF FT NOT > LT
5190	016352	104401	010541			TYPE	,TRKNLW	:TYPE "FT>LT"
5191	016356	104401	010510			TYPE	,NOTALD	:TYPE "NOT ALLOWED"
5192	016362	000675				BR	ASKMDE	:GO ASK AGAIN FOR PARAMETERS
5193	016364	000137	017362		20\$:	JMP	RUNTST	:GO RUN THE TESTS

```
.SBTTL TO - TYPE (T) LIST ROUTINE
;*THIS ROUTINE TYPES THE ENTIRE PARAMETER LIST ON THE
;*CONSOLE, LINE BY LINE. EACH LINE HAS THE FOLLOWING
;*FORMAT: XX=YYYYYY, WHERE XX IS A PARAMETER MNEMONIC
;*AND YYYYYY IS THE VALUE OF THE PARAMETER IN OCTAL.
;*TYPING (↑C) CAUSES IMMEDIATE RETURN TO DRVTST,
;*AND (↑Z) CAUSES RETURN TO ASKMDE.
```

5205	016370					TYPLST:		
5206	016370	005001				CLR	R1	:INITIALIZE INDEX
5207	016372	004737	027302			JSR	PC,PREPKB	:PREPARE FOR POSSIBLE KBD INPUT
5208	016376	004737	032640		1\$:	JSR	PC,TYPPRM	:TYPE CURRENT PARAMETER AND VALUE
5209	016402	104401	001315			TYPE	,\$CRLF	:TYPE <CR>, <LF>
5210	016406	005737	005522			TST	INTCHR	:SEE IF ANY INPUT AT KBD
5211	016412	001420				BEQ	8\$:BR IF NO INPUT
5212	016414	122737	000003	005522		CMPB	#003,INTCHR	:SEE IF (↑C) TYPED
5213	016422	001002				BNE	4\$:BR IF NOT (↑C)
5214	016424	000137	014532			JMP	DRVTST	:JUMP TO ASK FOR DRIVE(S) AGAIN
5215	016430	122737	000032	005522	4\$:	CMPB	#032,INTCHR	:SEE IF (↑Z) TYPED
5216	016436	001002				BNE	6\$:BR IF NOT (↑Z)
5217	016440	000137	016156			JMP	ASKMDE	:JUMP TO ASK FOR NEW MODE
5218	016444	004737	027322		6\$:	JSR	PC,ECOBAD	:ECHO BAD INPUT
5219	016450	004737	027302			JSR	PC,PREPKB	:PREPARE FOR POSSIBLE KBD INPUT
5220	016454	022761	040515	005776	8\$:	CMP	#MA,PRMNM(R1)	:SEE IF PARAM. IS (MA)
5221	016462	001002				BNE	10\$:BR IF NOT (MA)
5222	016464	062701	000002			ADD	#2,R1	:INCREMENT PARAMETER INDEX
5223	016470	062701	000002		10\$:	ADD	#2,R1	:INCREMENT PARAMETER INDEX
5224	016474	010102				MOV	R1,R2	:GET COPY OF INDEX
5225	016476	062702	005646			ADD	#PRMLST,R2	:COMPUTE POSITION IN LIST
5226	016502	022702	005674			CMP	#PRDFLT,R2	:SEE IF DONE WITH LIST
5227	016506	001333				BNE	1\$:BR IF NOT DONE YET
5228	016510	032737	100000	005666		BIT	#BIT15,PT	:SEE IF PATTERN 15 SPECIFIED
5229	016516	001005				BNE	12\$:BR IF PATTERN SPECIFIED
5230	016520	042777	000100	162416		BIC	#BIT6,\$STKS	:DISABLE KBD INTERRUPT
5231	016526	000137	016156			JMP	ASKMDE	:JUMP TO ASK FOR NEW MODE
5232						:TYPE OUT USER-DEFINED PATTERN 15		
5233	016532	104401	010666		12\$:	TYPE	,PFIFTN	:TYPE "USER-DEFINED PATTERN 15 :"
5234	016536	005001				CLR	R1	:INITIALIZE WORD INDEX
5235	016540	005737	005522		14\$:	TST	INTCHR	:SEE IF ANY INPUT
5236	016544	001420				BEQ	20\$:BR IF NO INPUT
5237	016546	122737	000003	005522		CMPB	#003,INTCHR	:SEE IF (↑C) TYPED
5238	016554	001002				BNE	16\$:BR IF NOT (↑C)
5239	016556	000137	014532			JMP	DRVTST	:JUMP TO ASK FOR DRIVE(S) AGAIN
5240	016562	122737	000032	005522	16\$:	CMPB	#032,INTCHR	:SEE IF (↑Z) TYPED

5241 016570 001002
 5242 016572 000137 016156
 5243 016576 004737 027322
 5244 016602 004737 027302
 5245 016606 004737 032720
 5246 016612 104401 001315
 5247 016616 062701 000002
 5248 016622 022701 000040
 5249 016626 001344
 5250 016630 042777 000100 162306
 5251 016636 000137 016156

BNE 18\$;BR IF NOT (↑Z)
 JMP ASKMDE ;JUMP TO ASK FOR NEW MODE
 18\$: JSR PC,ECOBAD ;ECHO BAD INPUT
 JSR PC,PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
 20\$: JSR PC,TYPPAT ;TYPE WORD XX = YYYYYY
 TYPE ,SCLRF ;TYPE <CR>, <LF>
 ADD #2,R1 ;INCREMENT INDEX
 CMP #32.,R1 ;SEE IF 16 WORDS TYPED
 BNE 14\$;BR IF NOT DONE YET
 BIC #BIT6,3\$TKS ;DISABLE KBD INTERRUPT
 JMP ASKMDE ;JUMP TO ASK FOR NEW MODE

.SBTTL TO - OPEN (O) LIST ROUTINE
 ;*THIS ROUTINE IS USED TO EITHER REQUEST LOADING OF ALL
 ;*DEFAULT PARAMETERS INTO THE PARAMETER LIST, OR TO
 ;*SEQUENTIALLY OPEN EACH PARAMETER FOR ALTERATION BY
 ;*TTY INPUT. TYPING (↑C) CAUSES IMMEDIATE RETURN TO
 ;*DRVTST, AND (↑Z) CAUSES RETURN TO ASKMDE.

5263 016642
 5264 016642 104401 010604
 5265 016646 004737 032272
 5266 016652 014532
 5267 016654 016156
 5268 016656 016642
 5269 016660 005700
 5270 016662 001433
 5271 016664 022737 000104 005262
 5272 016672 001405
 5273 016674 104401 005262
 5274 016700 104401 001314
 5275 016704 000756
 5276
 5277 016706 012700 005674
 5278 016712 012702 005646
 5279 016716 012022
 5280 016720 022702 005674
 5281 016724 001374
 5282 016726 105737 003106
 5283 016732 001005
 5284 016734 012737 000001 023540
 5285 016742 000137 017444
 5286 016746 000137 016156
 5287 016752 005001
 5288
 5289 016754 004737 032640
 5290 016760 104401 013426
 5291 016764 104401 013446
 5292
 5293 016770 004737 032272
 5294 016774 014532
 5295 016776 016156
 5296 017000 016754

OPNLST: TYPE DFQUES ;ASK IF ALL DEFAULT VALUES DESIRED
 JSR PC,RDCHRS ;READ RESPONSE TO DEFAULT QUESTION
 DRVTST ;(↑C) RETURN ADDRESS
 ASKMDE ;(↑Z) RETURN ADDRESS
 OPNLST ;(↑U) OR ERROR RETURN ADDRESS
 TST RO ;SEE IF NULL INPUT
 BEQ NOTDFT ;BR IF DEFAULTS NOT REQUESTED
 CMP #D,BUFFO ;SEE IF (D) TYPED
 BEQ LODFPT ;BR IF DEFAULTS DESIRED
 TYPE ,BUFFO ;ECHO BAD INPUT
 TYPE ,SQUES
 BR OPNLST ;GO ASK AGAIN
 ;LOAD ALL DEFAULT VALUES INTO PARAMETER LIST
 LODFPT: MOV #PRDFLT,R0 ;LOAD DEFAULT LIST ADDRESS
 MOV #PRMLST,R2 ;LOAD PARAMETER LIST ADDRESS
 4\$: MOV (R0)+,(R2)+ ;LOAD A DEFAULT VALUE
 CMP #PRDFLT,R2 ;SEE IF DONE YET
 BNE 4\$;BR IF MORE VALUES TO LOAD YET
 TSTB MDFLAG ;SEE IF DEFAULT MODE
 BNE 6\$;BR IF NOT DEFAULT MODE
 MOV #1,\$EOPCT ;SET PASS COUNT = 1
 JMP STPASS ;GO RUN DFLT TESTS
 6\$: JMP ASKMDE ;JUMP TO ASK FOR NEW MODE
 NOTDFT: CLR R1 ;INITIALIZE INDEX
 ;TYPE CURRENT PARAMETER AND VALUE
 8\$: JSR PC,TYPPRM ;TYPE PARAMETER AND VALUE
 TYPE ,SPACE1 ;TYPE A SPACE
 TYPE ,PROMPT ;TYPE ASTERISK AND SPACE
 ;READ AND CHECK INPUT, IF ANY
 JSR PC,RDCHRS ;READ OCTAL DIGITS TYPED
 DRVTST ;(↑C) RETURN ADDRESS FOR RDCHRS
 ASKMDE ;(↑Z) RETURN ADDRESS FOR RDCHRS
 8\$;(↑U) OR ERROR RETURN ADDR. FOR RDCHRS


```

5297 017002 005700          TST      RO          ;SEE IF ANY INPUT
5298 017004 001007          BNE     10$         ;BR IF ANY INPUT
5299 017006 022761 040515 005776  CMP     #'MA,PRMNM(R1) ;SEE IF (MA) JUST DEFAULTED
5300 017014 001023          BNE     12$         ;BR IF NOT (MA)
5301 017016 062701 000002  ADD     #2,R1       ;INCREMENT INDEX
5302 017022 000420          BR      12$         ;BR TO MOVE ON TO NEXT PARAMETER
5303 017024 022700 000010 10$:   CMP     #10,RO    ;SEE IF 8 CHARACTERS TYPED
5304 017030 002431          BLT     14$         ;BR IF MORE THAN 8 TYPED
5305 017032 012746 005262  MOV     #BUFF0,-(SP) ;PUT BUFFER ADDR. ON STACK FOR OCTBIN
5306 017036 004737 053014  JSR     PC,OCTBIN   ;CHECK DIGITS AND CONVERT TO BINARY
5307 017042 017114          14$          ;ERROR RETURN ADDRESS FOR OCTBIN
5308 017044 012637 005466  MOV     (SP)+,LOWOCT ;GET LOW BINARY BITS
5309 017050 013737 053146 005470  MOV     $HIOCT,HIGOCT ;GET HIGH BINARY BITS
5310          ;CHECK PARAMETER VALUE AND PUT INTO LIST
5311 017056 004737 033276  JSR     PC,CHKPRM   ;CHECK VALIDITY OF PARAM VALUE
5312 017062 017114          14$          ;ERROR RETURN ADDR. FOR CHKPRM
5313          ;OPEN USER-DEFINED PATTERN 15 FOR MODIFICATION, IF NECESSARY
5314 017064 004737 032752 12$:   JSR     PC,MODP15
5315          ;MOVE ON TO NEXT PARAMETER
5316 017070 062701 000002  ADD     #2,R1       ;INCREMENT THE PARAMETER INDEX
5317 017074 010102          MOV     R1,R2       ;GET COPY OF INDEX
5318 017076 062702 005646  ADD     #PRMLST,R2  ;COMPUTE POSITION IN LIST
5319 017102 022702 005674  CMP     #PRDFLT,R2 ;SEE IF DONE WITH LIST
5320 017106 001322          BNE     8$          ;BR IF NOT DONE YET
5321 017110 000137 016156  JMP     ASKMDE       ;JUMP TO ASK FOR NEW MODE
5322 017114 104401 005262 14$:   TYPE   ,BUFF0     ;ECHO BAD INPUT
5323 017120 104401 001314  TYPE   ,SQUES       ;TYPE <?> AND <CR>,<LF>
5324 017124 000713          BR      8$          ;BR TO ASK AGAIN

```

```

5325
5326
5327
5328          .SBTTL TO - SET (S) INDIVIDUAL PARAM ROUTINE
5329          ;*THIS ROUTINE IS USED TO ALLOW SETTING OF ANY INDIVIDUAL
5330          ;*PARAMETER VALUE BY TTY INPUT. A PROMPTER (>) IS
5331          ;*TYPED AND THE ROUTINE WAITS FOR INPUT OF THE FORM
5332          ;* XX=YYYYYY, WHERE XX IS THE MNEMONIC FOR ANY VALID
5333          ;*PARAMETER IN THE LIST, AND YYYYYY IS THE DESIRED PARAMETER
5334          ;*VALUE IN OCTAL DIGITS. THE PROGRAM THEN ASKS FOR ANOTHER
5335          ;*PARAMETER BY TYPING ">" AGAIN. TYPING (↑) CAUSES
5336          ;*IMMEDIATE RETURN TO DRVTST, AND (↑Z) CAUSES RETURN TO
5337          ;*ASKMDE.

```

```

5338
5339 017126          SETPRM:
5340 017126 104401 013451  TYPE   ,PRMPSP      ;TYPE ">" PROMPTER
5341          ;READ AND CHECK INPUT, IF ANY
5342 017132 004737 032272  JSR     PC,RDCHRS   ;READ INPUT LINE
5343 017136 014532          DRVTST              ;(↑C) RETURN ADDRESS FOR RDCHRS
5344 017140 016156          ASKMDE              ;(↑Z) RETURN ADDRESS FOR RDCHRS
5345 017142 017126          SETPRM              ;(↑U) OR ERROR RETURN ADDR. FOR RDCHRS
5346 017144 020027 000004  CMP     RO,#4        ;SEE IF AT LEAST 4 CHARACTERS TYPED
5347 017150 002005          BGE     6$          ;BR IF AT LEAST 4 TYPED
5348 017152 104401 005262 4$:   TYPE   ,BUFF0     ;ECHO BAD INPUT
5349 017156 104401 001314  TYPE   ,SQUES       ;TYPE <?> AND <CR>,<LF>
5350 017162 000761          BR      SETPRM      ;BR TO ASK FOR INPUT AGAIN
5351 017164 020027 000013 6$:   CMP     RO,#11.    ;SEE IF ELEVEN OR LESS CHARS TYPED
5352 017170 003370          BGT     4$          ;BR IF MORE THAN ELEVEN TYPED

```

```

5353 ;SEE IF THIS MNEMONIC RESIDES IN PARAM MNEMONIC TABLE
5354 017172 005001          CLR      R1          ;INITIALIZE PARAMETER INDEX
5355 017174 023761 005262 005776 9$:  CMP      BUFFD,PRMNEM(R1) ;TRY TO MATCH 2-CHAR MNEMONIC
5356 017202 001411          BEQ      10$          ;BR IF PARAMETER FOUND IN TABLE
5357 017204 062701 000002          ADD      #2,R1          ;INCREMENT INDEX
5358 017210 010102          MOV      R1,R2          ;GET COPY OF INDEX
5359 017212 062702 005646          ADD      #PRMLST,R2      ;SEE IF ALL ENTRIES HAVE BEEN CHECKED
5360 017216 022702 005674          CMP      #PRDFLT,R2
5361 017222 001364          BNE      8$           ;BR IF MORE TO CHECK YET
5362 017224 000752          BR       4$           ;BR TO ECHO BAD INPUT
5363 ;CHECK FOR VALID LINE FORMAT
5364 017226 012702 000002 10$:  MOV      #2,R2          ;INITIALIZE BUFFER POINTER
5365 017232 122762 000075 005262  CMPB     #'=,BUFFD(R2) ;SEE IF NEXT CHAR IS "="
5366 017240 001411          BEQ      12$          ;BR IF IT IS "="
5367 017242 122762 000040 005262  CMPB     #040,BUFFD(R2) ;SEE IF NEXT CHAR IS A SPACE
5368 017250 001340          BNE      4$           ;BR TO ECHO BAD INPUT
5369 017252 005202          INC      R2           ;INCREMENT BUFFER POINTER
5370 017254 122762 000075 005262  CMPB     #'=,BUFFD(R2) ;SEE IF NEXT CHAR IS "="
5371 017262 001333          BNE      4$           ;BR TO ECHO INVALID INPUT
5372 017264 005202 12$:  INC      R2           ;INCREMENT BUFFER POINTER
5373 017266 020200          CMP      R2,RO        ;SEE IF MORE CHARS LEFT IN BUFFER
5374 017270 002330          BGE      4$           ;BR TO ECHO INVALID INPUT
5375 017272 122762 000040 005262  CMPB     #040,BUFFD(R2) ;SEE IF NEXT CHARACTER IS SPACE
5376 017300 001003          BNE      14$          ;BR IF NOT A SPACE
5377 017302 005202          INC      R2           ;INCREMENT BUFFER POINTER
5378 017304 020200          CMP      R2,RO        ;SEE IF MORE CHARS LEFT IN BUFFER
5379 017306 002321          BGE      4$           ;BR IF NONE LEFT
5380 017310 160200 14$:  SUB      R2,RO        ;SUBTRAC POINTER FROM CHAR COUNT
5381 017312 020027 000010          CMP      RO,#10       ;SEE HOW MANY CHARS LEFT
5382 017316 003315          BGT      4$           ;BR IF TOO MANY LEFT
5383 ;CHECK FOR LEGAL PARAMETER VALUE
5384 017320 062702 005262          ADD      #BUFFD,R2    ;GET ADDRESS OF DIGITS
5385 017324 010246          MOV      R2,-(SP)     ;PUT IT ON STACK FOR OCTBIN
5386 017326 004737 053014          JSR      PC,OCTBIN    ;CHECK DIGITS AND CONVERT TO BINARY
5387 017332 017152          4$           ;ERROR RETURN ADDR. FOR OCTBIN
5388 017334 012637 005466          MOV      (SP)+,LOWOCT ;GET LOW BINARY BITS
5389 017340 013737 053146 005470          MOV      $HI0CT,HIGOCT ;GET HIGH BINARY BITS
5390 017346 004737 033276          JSR      PC,CHKPRM    ;CHECK VALIDITY OF PARAMETER VALUE
5391 017352 017152          4$           ;ERROR RETURN ADDR. FOR CHKPRM
5392 ;OPEN USER-DEFINED PATTERN 15 FOR MODIFICATION, IF NECESSARY
5393 017354 004737 032752          JSR      PC,MODP15
5394 017360 000662          BR       SETPRM      ;RETURN TO ASK FOR ANOTHER PARAMETER
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408

```

```

.SBTTL TO - RUN (R) TESTS ROUTINE
; *THIS ROUTINE IS ENTERED TO SET UP THE RUNNING OF TESTS,
; *WHOSE PARAMETERS HAVE BEEN DETERMINED. IT FIRST ASKS
; *FOR THE DESIRED NUMBER OF PROGRAM PASSES, AND STORES
; *THIS IN $EOPCT FOR THE END OF PASS ROUTINE. THEN, THE
; *DRIVE LIST IS SCANNED FOR THE FIRST DESIRED DRIVE, AND
; *THE HEADER ON SECTOR 0 IS READ TO GET THE FORMAT FOR THIS
; *DRIVE. THIS IS STORED IN THE FORMAT BYTE, AND A JUMP IS
; *MADE TO THE FIRST TEST.
; *THE END OF PASS ROUTINE RETURNS TO "NEWDRV" TO SELECT
; *EACH SUCCESSIVE DRIVE FOR TESTING, UNTIL THE ENTIRE PASS

```

;* (ALL DRIVES) IS COMPLETED.

017362			
017363	104401	011035	
017364	004737	032272	
017372	014532		
017374	016156		
017376	017362		
017400	005700		
017402	001403		
017404	022700	000005	
017410	002005		
017412	104401	005262	
017416	104401	001314	
017422	000757		
017424	012746	005262	
017430	004737	053014	
017434	017412		
017436	012637	023540	
017442	001763		
017444	005037	001326	
017450	112737	000001	003110
017456	012737	177777	005500
017464	005037	001330	
017470	012737	000000	177776
017476	012706	001100	
017502	105037	003120	
017506	005037	005522	
017512	005237	005500	
017516	022737	000010	005500
017524	001002		
017526	000137	023506	
017532	013700	005500	
017536	105760	005606	
017542	001752		
017544	105037	001102	
017550	005037	001304	
017554	010037	001332	
017560	004737	030776	
017564	112765	000125	000001
017572	004737	040774	
017576	132765	000020	000007
017604	001404		
017606	142765	000020	000007
017614	000403		
017616	152765	000020	000007
017624	004737	040774	
017630	105037	003115	
017634	013737	005662	005506
017642	013737	005664	005510
017650	005762	000024	
017654	032762	001000	000024

```

RUNTST:
:INPUT THE DESIRED NUMBER OF PROGRAM PASSES
4$: TYPE ENT PAS :ASK FOR NUMBER OF PASSES
JSR PC,ROCHRS :READ RESPONSE
DRVTST :(+C) RETURN ADDRESS
ASKMDE :(+Z) RETURN ADDRESS
4$: :(+U) OR ERROR RETURN ADDRESS
TST RO :SEE IF NULL INPUT
BEQ 6$:GO ASK AGAIN
CMP #5,RO :SEE HOW MANY CHARS TYPED
BGE 8$:BR IF 5 OR LESS
6$: TYPE ,BUFFD :ECHO BAD INPUT
TYPE ,SQUES
BR 4$:GO ASK AGAIN
8$: MOV #BUFFD, -(SP) :GET BUF ADDR ON STACK FOR OCTBIN
JSR PC,OCTBIN :CHECK DIGITS AND CONVERT TO BINARY
6$: :ERROR RETURN ADDRESS FOR OCTBIN
MOV (SP)+, $EOPCT :SET DESIRED NO. OF PASSES (1-77777)
BEQ 6$:BR IF 0, TO ASK AGAIN
:THIS IS THE ACTUAL START OF A RUN WITH X PASSES
$TPASS: CLR $PASS :INIT. THE PASS NUMBER
MOV #1, TSTING :SET "RUNNING TESTS" FLAG
NEWPAS: MOV #177777, DRIVE :INITIALIZE DRIVE NO. TO -1
CLR $DEVCT :INIT APT DEVICE COUNT TO 0
:CHECK DRIVE LIST FOR MORE DRIVES TO TEST ON THIS PASS
NEWDRV: MOV #PRO, $#PS :RE-ESTABLISH PRIORITY 0
MOV #STACK, SP :RESTORE THE STACK
CLRB DRVERS :CLEAR ERROR COUNT FOR CURRENT DRIVE
CLR INTCHR :CLEAR TTY INPUT BUFFER WORD
INC DRIVE :INCREMENT DRIVE NUMBER
CMP #10, DRIVE :SEE IF DONE WITH THIS PASS
BNE 2$:BR IF NOT DONE CHECKING LIST
JMP DUNPAS :JUMP IF DONE WITH THIS PASS
2$: MOV DRIVE, RO :GET NEW DRIVE NUMBER
TSTB DRVLST(RO) :SEE IF THIS DRIVE IS MARKED IN LIST
BEQ NEWDRV :BR IF NOT MARKED, TO CHECK ANOTHER
CLRB $TSTNM :INIT TEST NUMBER TO 0
CLR $TIMES :INIT ITERATION COUNT
MOV RO, $UNIT :SET DRIVE NO. FOR APT
:READ THE HEADER ON SECTOR 0, TRACK 0 CYL 0, AND GET THE DRIVE FORMAT
JSR PC, INITSS :INIT. DRIVER PARAMS AND S.S.
MOV #RDHEAD, P.CMND(R5) :SET READ HEADER COMMAND
JSR PC, DRVCAL :DO READ HEADER
BITB #B.CFMT, P.CS1H(R5) :COMPLEMENT THE FORMAT BIT
BEQ 4$:GIVEN TO THE CONTROLLER.
BICB #B.CFMT, P.CS1H(R5) :AND DO READ HEADER. THIS
BR 6$:WILL CAUSE SECTOR 0 HEADER
4$: BICB #B.CFMT, P.CS1H(R5) :TO BE READ.
6$: JSR PC, DRVCAL :READ HEADER 0
CLRB FORMAT :INITIALIZE FORMAT BYTE TO 0
MOV S2, FS :INIT SECTOR LIMITS FOR 22(DEC) SECTOR FMT
MOV S3, LS
TST RKDB(R2) :POP SILO ONE TIME
BIT #BIT9, RKDB(R2) :TEST FORMAT BIT IN HEADER WORD 2

```

017668	017662	001411			BEG	B\$:BR IF 22 SECTOR FORMAT
017669	017664	152737	000020	003115	BISB	#B.CFMT,FORMAT	:SET 20 SECTOR FORMAT FOR THIS DRIVE
017670	017672	013737	005656	005506	MOV	SO,FS	:INIT SECTOR LIMITS FOR 20(DEC) SECTOR FMT
017671	017700	013737	005660	005510	MOV	SI,LS	
017672	017706	013737	005672	005502	MOV	ST,STALLS	:SET NUMBER OF UNIT STALLS DESIRED
017673	017714	004737	030776		JSR	PC,INITSS	:INIT THE S.S.
017674	017720	004737	037354		JSR	PC,REDBSF	:READ BAD SECTOR FILE FOR THIS DRIVE
017675	017724	113737	005500	011167	MOVB	DRIVE,DRVNO	:GET DRIVE NO.
017676	017732	152737	000060	011167	BISB	#'O,DRVNO	:CONVERT TO ASCII
017677	017740	104401	011146		TYPE	TSTDN	:TYPE "TESTING DRIVE X"
017678	017744	005737	001326		TST	\$PASS	:SEE IF THIS IS FIRST PASS
017679	017750	001012			BNE	10\$:BR IF NOT FIRST PASS
017680	017752	004737	033602		JSR	PC,DRVSER	:TYPE "DRIVE SER. NO. XXX"
017681	017756	004737	033722		JSR	PC,CRTSER	:TYPE "CART. SER. NO. XXXXXXXXXXXX"
017682	017762	032737	000020	005670	BIT	#BIT4,CS	:SEE IF BAD SECTORS SHOULD BE TYPED
017683	017770	001402			BEG	10\$:BR IF NOT
017684	017772	004737	037642		JSR	PC,TYPBSF	:TYPE BAD SECTOR FILES
017685	017776	005037	005522		CLR	INTCHR	:INIT. TTY INPUT CHAR BUFFER

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041

020002 000004
020004 012737 000001 001324
020012 004737 032130
020016 004737 032176
020022 000137 021112
020026 004737 030776
020032 004737 027302
020036 105737 003106
020042 001005
020044 005037 005506
020050 012737 000012 005510
020056 013765 005646 000002 3\$:
020064 113765 005506 000004
020072 105065 000005
020076 013737 005650 005504
020104 022737 000632 005650
020112 001002
020114 005337 005504
020120 012703 072307 4\$:
020124 004737 036050
020130 012703 167230
020134 005365 000002
020140 100402
020142 004737 036050
020146 062765 000002 000002 6\$:
020154 022765 000632 000002

: TEST 1 OFFSET-TO-FAILURE MEASUREMENTS
: *IN THIS TEST A REPEATED PATTERN 072307(OCT) IS WRITTEN INTO ALL
: *WORDS OF SECTORS FS AND LS ON TRACKS 0-2 OF CYLINDERS FC AND LC.
: *THE SAME SECTORS ON ADJACENT CYLINDERS ARE THEN WRITTEN WITH
: * 167230(OCT) (TO ESTABLISH A KNOWN PATTERN) , BUT THEY ARE NOT TESTED.
: *THEN, EACH DESIRED SECTOR IS WRITE CHECKED WITH INCREASING POSITIVE
: *OFFSETS, UNTIL EITHER A "DATA TYPE" ERROR OCCURS, OR THE FULL RANGE
: *OF POSITIVE OFFSETS HAVE BEEN APPLIED. THEN, THE SAME SECTOR IS WRITE
: *CHECKED WITH INCREASING NEGATIVE HEAD OFFSETS, UNTIL FAILURE OR
: *MAXIMUM NEGATIVE OFFSET IS APPLIED. THIS IS DONE FOR ALL THE SECTORS
: *ORIGINALLY WRITTEN, IN THE FOLLOWING ORDER : SECTORS FS AND LS ON
: *CYLINDERS FC AND LC FOR TRACK 0, THEN FOR TRACK 1, AND FINALLY FOR
: *TRACK 2. THE OFFSET-TO-FAILURE RESULTS ARE REPORTED IN ORDER OF
: *INCREASING TRACKS, AS FOLLOWS :
:
: * OFFSET-TO-FAILURE MEASUREMENTS :
:
: * TRACK CYLN SECT +OFST -OFST
: * (UIN) (UIN)
: * X XXX XX XXXX XXXX
: * X XXX XX XXXX XXXX
: * X XXX XX XXXX XXXX
: * ETC.
:
: *NOTE: IN AN ADRS 200 DEFAULT RUN, FS DEFAULTS TO 0, AND LS DEFAULTS
: *TO 10(DEC), FOR THIS TEST.
: *****

TST1: SCOPE
MOV #1, \$TESTN ;SET TEST NUMBER IN APT MAIL BOX
JSR PC, SETUP ;SET UP FOR LOOP ON ERROR
JSR PC, CHKITR ;SEE IF ITER. NO. = 0 FOR THIS TEST
JMP TST2 ;JUMP TO NEXT TEST
;RETURN HERE FROM CHKITR IF TEST SHOULD BE RUN
JSR PC, INITSS ;INITIALIZE DRIVER PARAMS AND SUB-SYS
JSR PC, PREPKB ;PREPARE FOR POSSIBLE KBD INPUT
;INITIALIZE PARAMETERS, WRITE THE SECTORS, AND WRITE CHECK THEM
TSTB MDFLAG ;SEE IF ADRS 200 DEFAULT RUN
BNE 3\$;BR IF NOT
CLR FS ;SET FS = 0
MOV #10, LS ;SET LS = 10(DEC)
3\$: MOV FC, P.CYLN(R5) ;SET CYL = FC
MOVB FS, P.SECT(R5) ;SET SECTOR = FS
CLRB P.TRCK(R5) ;SET TRACK = 0
MOV LC, CYLNDR
CMP #632, LC ;SEE IF LC = 632
BNE 4\$;BR IF NOT 632
DEC CYLNDR ;MAKE IT 631 (PRESERVE BSF)
4\$: MOV #72307, R3 ;GET DATA PATTERN
JSR PC, WRTSEC ;DO WRITE AND WRITE CHECK
MOV #167230, R3 ;DATA FOR AJACENT CYLS
DEC P.CYLN(R5) ;NEXT LOWER CYL
BMI 6\$;BR IF CYL NEGATIVE
39 JSR PC, WRTSEC ;DO WRITE AND WRITE CHECK
40 ADD #2, P.CYLN(R5) ;NEXT HIGHER CYL
41 CMP #632, P.CYLN(R5) ;SEE IF CYL = 632

E09

```

5543 020162 001004          BNE      8$          ;BR IF NOT
5544 020164 122765 000002 000005  CMPB    #2,P.TRCK(R5) ;SEE IF BSF
5545 020172 001402          BEQ     10$         ;BR IF YES, TO PROTECT IT
5546 020174 004737 036050          JSR     PC,WRTSEC   ;DO WRITE AND WRITE CHECK
5547 020200 005365 000002          DEC     P.CYLN(R5) ;RESTORE CYL NO.
5548 020204 105265 000005          INCB   P.TRCK(R5)  ;INCR TRACK
5549 020210 122765 000003 000005  CMPB    #3,P.TRCK(R5) ;TRACK = 3 YET ?
5550 020216 001340          BNE     4$          ;BR IF NOT YET
5551 020220 126537 000004 005510  CMPB    P.SECT(R5),LS ;SEE IF SECTOR = LS
5552 020226 001406          BEQ     14$         ;BR IF YES
5553 020230 113765 005510 000004  MOVB   LS,P.SECT(R5) ;SET SECTOR = LS
5554 020236 105065 000005          CLRB   P.TRCK(R5)  ;SET TRACK = 0
5555 020242 000726          BR      4$          ;BR TO CONTINUE WRITING
5556 020244 026537 000002 005504 14$:  CMP    P.CYLN(R5),CYLNDR ;SEE IF CYL = LC
5557 020252 001407          BEQ     16$         ;BR IF YES
5558 020254 013765 005504 000002  MOV    CYLNDR,P.CYLN(R5) ;SET CYL = LC
5559 020262 113765 005506 000004  MOVB   FS,P.SECT(R5) ;SET SECTOR = FS AGAIN
5560 020270 000762          BR      12$         ;BR TO CONTINUE WRITING
5561 020272 032737 000002 005670 16$:  ;INITIALIZE PARAMETERS FOR OFFSET MEASUREMENTS
5562 020300 001002          BIT     #BIT1,CS   ;SEE IF REPORTS ARE INHIBITED
5563 020302 104401 011300          BNE     18$         ;BR IF INHIBITED
5564 020306 113765 005652 000005 18$:  TYPE   OFSHED      ;TYPE HEADINGS
5565 020314 013765 005646 000002  MOVB   FT,P.TRCK(R5) ;SET TRACK = FT
5566 020322 113765 005506 000004  MOV    FC,P.CYLN(R5) ;SET CYLINDER = FC
5567          MOVB   FS,P.SECT(R5) ;SET SECTOR = FS
5568          ;LOAD THE R/W BUFFER FOR WRITE CHECKS
5569 020330 012700 064640          MOV    #RWBUR,RO   ;SET BUFFER ADDRESS
5570 020334 012701 000400          MOV    #400,R1     ;PREPARE TO LOAD 400(OCT) WORDS
5571 020340 012720 072307 19$:  MOV    #072307,(RO)+ ;LOAD A WORD INTO BUFFER
5572 020344 005301          DEC    R1           ;DECR COUNTER
5573 020346 001374          BNE     19$         ;BR IF NOT DONE YET
5574 020350 005004 20$:  CLR    R4           ;INIT (+) OFFSET VALUE
5575 020352 005000          CLR    RO           ;INIT OFFSET NUMBER TO 0
5576 020354 012737 177777 005544  MOV    #-1,PLOFST   ;INITIALIZE CURRENT FAILING (+) OFFSET
5577 020362 012737 177777 005546  MOV    #-1,NGOFST   ;INITIALIZE CURRENT FAILING (-) OFFSET
5578          ;PERFORM OFFSETS AND WRITE CHECKS
5579 020370 112765 000117 000001 22$:  MOVB   #SEEK,P.CMND(R5) ;SET SEEK COMMAND
5580 020376 004737 040774          JSR    PC,DRVCAL   ;PERFORM SEEK
5581 020402 110065 000006          MOVB   RO,P.OFST(R5) ;SET CURRENT OFFSET
5582 020406 112765 000115 000001  MOVB   #OFFSET,P.CMND(R5) ;SET OFFSET COMMAND
5583 020414 004737 040774          JSR    PC,DRVCAL   ;PERFORM OFFSET
5584 020420 112765 000131 000001  MOVB   #WRTCHK,P.CMND(R5) ;SET WRITE CHECK COMMAND
5585 020426 012737 021056 003036  MOV    #WRCKHD,A.ABNL ;SET SPECIAL ERROR HANDLER ADDRESS
5586 020434 105037 003131          CLRB   WCEFLG      ;CLEAR WRITE CHECK ERROR FLAG
5587 020440 004737 040774          JSR    PC,DRVCAL   ;PERFORM A WRITE CHECK
5588 020444 012737 042466 003036  MOV    #ERRHDL,A.ABNL ;RESTORE ERROR HANDLER ADDRESS
5589 020452 112765 000177 000001  MOVB   #SUBCLR,P.CMND(R5) ;SET SUBSYS CLEAR CMND
5590 020460 004737 040774          JSR    PC,DRVCAL   ;CLEAR THE SUBSYSTEM
5591 020464 105737 003131          TSTB   WCEFLG      ;SEE IF A WRITE CHECK ERROR OCCURRED
5592 020470 001020          BNE     26$         ;BR IF AN ERROR OCCURRED
5593 020472 010001          MOV    RO,R1       ;GET A COPY OF CURRENT OFFSET
5594 020474 005200          INC    RO           ;INCREMENT OFFSET
5595 020476 062704 000031  ADD    #25,R4       ;INCR OFFSET VALUE BY 25(DEC) UIN
5596 020502 042701 177700          BIC    #177700,R1  ;MASK FOR MAGNITUDE BITS
5597 020506 022701 000060          CMP    #60,R1      ;SEE IF MAX OFFSET APPLIED IN THIS DIRECTION
5598 020512 001326          BNE     22$         ;BR IF NOT YET

```

F09

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07 T1

MACY11 27(1006) 05-OCT-76 10:59 PAGE 110
 OFFSET-TO-FAILURE MEASUREMENTS

SEQ 0109

```

5598 020514 032700 000200 BIT #BIT7,R0 ;SEE IF NEG OFFSETS DONE YET
5599 020520 001015 BNE 32$ ;BR IF YES, TO PRINT CURRENT SECTOR RESULT
5600 020522 012700 000200 24$: MOV #200,R0 ;INIT FOR NEG OFFSETS
5601 020526 005004 CLR R4 ;INIT (-) OFFSET VALUE TO 0
5602 020530 000717 BR 22$ ;BR TO APPLY NEG OFFSETS
5603 020532 032700 000200 26$: BIT #BIT7,R0 ;SEE IF NEG OFFSETS DONE YET
5604 020536 001403 BEQ 28$ ;BR IF NOT YET
5605 020540 010437 005546 MOV R4,NGOFST ;SAVE THIS FAILING (-) OFFSET VALUE
5606 020544 000403 BR 32$ ;BR TO PRINT CURRENT SECTOR RESULT
5607 020546 010437 005544 28$: MOV R4,PLOFST ;SAVE THIS FAILING (+) OFFSET VALUE
5608 020552 000763 BR 24$ ;BR TO APPLY NEGATIVE OFFSETS
5609 020554 032737 000002 005670 32$: BIT #BIT1,CS ;SEE IF REPORTS INHIBITED
5610 020562 001051 BNE 43$ ;BR IF INHIBITED
5611 ;TYPE OFFSET RESULTS FOR THIS SECTOR
5612 020564 116501 000005 MOVB P.TRCK(R5),R1 ;GET TRACK NO.
5613 020570 010146 MOV R1,-(SP) ;PUT IT ON STACK
5614 020572 104403 TYPOS ;TYPE IT
5615 020574 003 .BYTE 3 ;3 DIGITS
5616 020575 000 .BYTE 0 ;SUPPRESS
5617 020576 104401 013424 TYPE SPACE3
5618 020602 016546 000002 MOV P.CYLN(R5),-(SP) ;PUT CYL ON STACK
5619 020606 104403 TYPOS ;TYPE IT
5620 020610 004 .BYTE 4 ;4 DIGITS
5621 020611 000 .BYTE 0 ;SUPPRESS
5622 020612 104401 013425 TYPE SPACE2
5623 020616 116501 000004 MOVB P.SECT(R5),R1 ;GET SECTOR
5624 020622 010146 MOV R1,-(SP) ;PUT IT ON STACK
5625 020624 104404 TYPON ;TYPE SECTOR
5626 020626 104401 013426 TYPE SPACE1
5627 020632 005737 005544 TST PLOFST ;SEE IF THERE WAS A FAILING (+) OFFSET
5628 020636 100003 BPL 36$ ;BR IF THERE WAS
5629 020640 104401 011271 TYPE NOFALS ;TYPE "NONE"
5630 020644 000403 BR 38$ ;CONTINUE
5631 020646 013746 005544 36$: MOV PLOFST,-(SP) ;PUT (+) OFFSET VALUE ON STACK
5632 020652 104405 TYPDS ;TYPE IT IN DECIMAL
5633 020654 104401 013426 38$: TYPE SPACE1
5634 020660 005737 005546 TST NGOFST ;SEE IF THERE WAS A FAILING (-) OFFSET
5635 020664 100003 BPL 40$ ;BR IF THERE WAS
5636 020666 104401 011271 TYPE NOFALS ;TYPE "NONE"
5637 020672 000403 BR 42$ ;CONTINUE
5638 020674 013746 005546 40$: MOV NGOFST,-(SP) ;PUT (-) OFFSET VALUE ON STACK
5639 020700 104405 TYPDS ;TYPE IT IN DECIMAL
5640 020702 104401 001315 42$: TYPE $CRLF ;TYPE <CR>,<LF>
5641 ;SET UP TO TEST NEXT SECTOR
5642 020706 126537 000004 005510 43$: CMPB P.SECT(R5),LS ;SEE IF SECTOR = LS
5643 020714 001404 BEQ 44$ ;BR IF YES
5644 020716 113765 005510 000004 MOVB LS,P.SECT(R5) ;SET SECTOR = LS
5645 020724 000611 BR 20$ ;GO TEST THIS SECTOR
5646 020726 026537 000002 005504 44$: CMP P.CYLN(R5),CYLNDR ;SEE IF CYL = LC
5647 020734 001410 BEQ 48$ ;BR IF YES
5648 020736 013765 005504 000002 MOV CYLNDR,P.CYLN(R5) ;SET CYL = LC
5649 020744 113765 005506 000004 46$: MOVB FS,P.SECT(R5) ;SET SECTOR = FS
5650 020752 000137 020350 JMP 20$ ;GO TEST THIS SECTOR
5651 020756 105265 000005 48$: INCB P.TRCK(R5) ;INCREMENT TRACK
5652 020762 122765 000003 000005 CMPB #3,P.TRCK(R5) ;SEE IF TRACK = 3
5653 020770 001404 BEQ 50$ ;BR IF YES
  
```

```

5654 020772 013765 005646 000002      MOV      FC,P.CYLN(R5)      ;SET CYL = FC
5655 021000 000761                    BR          46$             ;GO TEST THIS SECTOR
5656 021002 112765 000113 000001 50$:     MOVB      #RECAL,P.CMND(R5) ;SET RECAL COMMAND
5657 021010 004737 040774                    JSR      PC,DRVCAL         ;DO CLEANUP RECALIBRATE
5658 021014 013737 005662 005506      MOV      S2,FS             ;RESTORE FS,LS FOR 22 SECTORS
5659 021022 013737 005664 005510      MOV      S3,LS
5660 021030 105737 003115                    TSTB     FORMAT           ;SEE IF 22 SECTORS
5661 021034 001406                    BEQ      54$             ;BR IF YES
5662 021036 013737 005656 005506      MOV      S0,FS             ;RESTORE FS,LS FOR 20 SECTORS
5663 021044 013737 005660 005510      MOV      S1,LS
5664 021052                    54$:
5665 021052 000137 021112                    JMP      TST2             ;JUMP TO NEXT TEST
5666
5667                    ;*THIS IS THE ABNORMAL RETURN FROM THE DRIVER, USED WHEN A
5668                    ;* "DATA TYPE" ERROR IS EXPECTED (AND VALID).
5669 021056 032765 040000 000020 WRCKHD: BIT  #WCE,P.CS2(R5) ;SEE IF WRITE CHECK ERROR OCCURRED
5670 021064 001006                    BNE      4$             ;BR IF WCE
5671                    ;CHECK FOR OTHER "DATA TYPE" ERROR
5672 021066 032765 130400 000034      BIT      #HVRC!DTE!OPI!DCK,P.ER(R5)
5673 021074 001002                    BNE      4$             ;BR IF A DATA ERROR OCCURRED
5674 021076 000137 042466                    JMP      ERRHDL          ;GO HANDLE OTHER ERROR
5675 021102 105237 003131 4$:      INCB     WCEFLG          ;SET WRITE CHECK ERROR FLAG
5676 021106 000137 044702                    JMP      RETNML         ;TAKE NORMAL RETURN
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699

```

```

*****
;*TEST 2      NPR/MEMORY WORD ADDRESSING TEST
;*IN THIS TEST, MEMORY WORD ADDRESSING CAPABILITY DURING RK06 NPR'S,
;*IS TESTED.
;*BEFORE TESTING BEGINS, THE XXDP LOADER IS SAVED, IF PRESENT, AT ADRS RWBUF.
;*THEN, STARTING AT THE FIRST ADDRESS OF THE NEXT 1K MEMORY BLOCK
;*BEYOND ADDRESS RWBUF+6000(OCTAL), WRITE UNIQUE NUMBERS
;*(STARTING WITH 1), INTO EACH OF UP TO 64K WORDS (DEPENDING UPON THE
;*AMOUNT OF PHYSICAL MEMORY). THESE NUMBERS REFLECT UNIQUE ADDRESSES
;*WITHIN THE 64K BLOCK. NEXT, WRITE THE 64K WORDS (MAX) ONTO THE DISK
;*AT ADDRESS FC,FT,FS (SCALED, IF NECESSARY, TO AVOID PACK OVERFLOW
;*OR DESTRUCTION OF THE BAD SECTOR FILE). THEN ZERO THE ENTIRE BLOCK
;*IN MEMORY, AND READ THE DATA BACK FROM THE DISK, AT THE SAME PHYSICAL
;*ADDRESSES, COMPARE THE DATA READ TO THE DATA WRITTEN. TYPE THE FAILING
;*VIRTUAL (IF MEM. MGT.) AS WELL AS PHYSICAL ADDRESSES, AND THE GOOD
;*AND BAD DATA, FOR UP TO THE FIRST 10(DEC) FAILING LOCATIONS.
;* IF MEMORY MANAGEMENT IS INSTALLED AND THERE IS ADDITIONAL MEMORY TO
;*EXERCISE, REPEAT THE ABOVE WORD ADDRESSING TEST, FOR EACH OF THE
;*REMAINING 64K MEMORY BLOCKS.
*****

```

```

5700 021112 000004      TST2:   SCOPE
5701 021114 012737 000002 001324      MOV      #2,$TESTN      ;:SET TEST NUMBER IN APT MAIL BOX
5702 021122 004737 032130                    JSR      PC,SETUP       ;:SET UP FOR LOOP ON ERROR
5703 021126 004737 032176                    JSR      PC,CHKITR      ;:SEE IF ITER. NO. = 0 FOR THIS TEST
5704 021132 000137 021422                    JMP      TST3           ;:JUMP TO NEXT TEST
5705                    ;:RETURN HERE FROM CHKITR IF TEST SHOULD BE RUN
5706 021136 004737 030776                    JSR      PC,INITSS     ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
5707 021142 004737 027302                    JSR      PC,PREPKB     ;:PREPARE FOR POSSIBLE KBD INPUT
5708                    ;:SAVE XXDP LOADER, IF PRESENT
5709 021146 004737 030456                    JSR      PC,FNDXDP     ;:COMPUTE STARTING ADR OF XXDP

```


H09

```

5710 021152 012737 064640 005540      MOV      #RWBUFF,XDPSAV      ;SET XXDP SAVE AREA ADR
5711 021160 005037 005542      CLR      XDPSAV+2
5712 021164 004737 030530      JSR      PC,SAVXDP          ;GO SAVE XXDP LOADER
5713 021170 112737 000001 003124      MOV      #1,XOVLAD         ;SET INDICATORS
5714 021176 112737 000001 003125      MOV      #1,XDPSVD
5715 021204 012737 177777 005534      MOV      #-1,PATRN         ;SET FLAG FOR SVPRMS
5716                                     ;INIT PARAMETERS, BEGIN TESTING
5717 021212 004737 037100      JSR      PC,SETUPM         ;INIT PARAMS
5718 021216 004737 033526      4$: JSR      PC,MXWRDC        ;COMPUTE MAX WORD COUNT FOR THIS MA
5719 021222 005701      TST      R1                ;SEE IF ALL MEMORY TESTED YET
5720 021224 100467      BMI      50$               ;BR IF ALL DONE
5721 021226 005065 000012      CLR      P.WC(R5)          ;INIT WORD COUNT TO 65,536(DEC)
5722 021232 005701      TST      R1                ;SEE IF WRD CNT SHOULD BE 65,536
5723 021234 001003      BNE      6$                ;BR IF YES
5724 021236 005400      NEG      R0
5725 021240 010065 000012      MOV      R0,P.WC(R5)       ;SET WORD COUNT
5726 021244 013765 005576 000010 6$: MOV      MA,P.BALO(R5)     ;SET BA BITS 0-15
5727 021252 013701 005600      MOV      MA+2,R1           ;GET MA BITS 16-21
5728 021256 042701 177774      BIC      #177774,R1        ;MASK FOR LO 2 BITS
5729 021262 150165 000007      BISB    R1,P.BAHI(R5)     ;SET BA BITS 16,17
5730 021266 005737 056246      TST      $K11             ;SEE IF MEM MGT PRESENT
5731 021272 100010      BPL      14$              ;BR IF NOT
5732                                     ;SET UP MEM MGT
5733 021274 013737 005576 005602      MOV      MA,PMA           ;SET BUFFER ADDRESS
5734 021302 013737 005600 005604      MOV      MA+2,PMA+2
5735 021310 004737 030334      JSR      PC,PAPRAR        ;PREPARE MEM MGT REG'S AND UNIBUS MAP
5736                                     ;PERFORM THE TESTS
5737 021314 012737 021314 001110 14$: MOV      #,$LPERR        ;SET NEW LOOP ON ERROR ADRS
5738 021322 004737 032130      JSR      PC,SETUP         ;SET UP FOR LOOP ON ERROR
5739 021326 004737 040304      JSR      PC,SVPRMS        ;SAVE PARAMS FOR THIS XFER
5740 021332 004737 036144      JSR      PC,LDMEM1        ;LOAD THIS MEM BLK WITH INCREMENTING DATA
5741 021336 112765 000123 000001      MOV      #WRDATA,P.CMND(R5) ;SET WRITE DATA COMMAND
5742 021344 004737 040416      JSR      PC,TRNSFR        ;WRITE THE DATA
5743 021350 005003      CLR      R3                ;SET DATA WORD = 0
5744 021352 004737 036156      JSR      PC,LDMEM2        ;LOAD MEM BLK WITH ALL ZEROS
5745 021356 112765 000121 000001      MOV      #RDATA,P.CMND(R5) ;SET READ COMMAND
5746 021364 004737 040416      JSR      PC,TRNSFR        ;READ THE DATA FROM DISK
5747 021370 004737 036310      JSR      PC,CKMEM1        ;PERFORM SOFTWARE COMPARE OF DATA
5748 021374 104411      SCOPER                    ;CHECK FOR INTERNAL LOOP ON ERROR
5749                                     ;GET NEW MA FOR NEXT TRANSFER
5750 021376 004737 037306      JSR      PC,INCRMA        ;COMPUTE NEXT MA
5751 021402 000705      BR      4$                ;GO SEE IF MORE MEMORY TO TEST
5752 021404 004737 030536      50$: JSR      PC,GETXDP      ;RESTORE XXDP LOADER
5753 021410 105737 003134      TSTB    UBMPRS           ;SEE IF UNIBUS MAP PRESENT
5754 021414 001402      BEQ      54$              ;BR IF NOT
5755 021416 005037 172516      CLR      @#SR3           ;DISABLE UNIBUS MAP
5756 021422
5757
5758
5759
5760
5761
5762
5763
5764
5765
;*****
; *TEST 3      NPR/MEMORY BLOCK ADDRESSING TEST
; *IN THIS TEST, MEMORY BLOCK ADDRESSING CAPABILITY DURING RK06
; *NPR'S IS TESTED.
; *BEFORE TESTING BEGINS, THE XXDP LOADER IS SAVED, IF PRESENT, AT ADRS RWBUF.
; *MEMORY WILL BE TESTED STARTING AT THE FIRST ADRS OF THE NEXT 1K
    
```

```

5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777 021422 000004
5778 021424 012737 000003 001324
5779 021432 004737 032130
5780 021436 004737 032176
5781 021442 000137 021762
5782
5783 021446 004737 030776
5784 021452 004737 027302
5785
5786 021456 004737 030456
5787 021462 012737 064640 005540
5788 021470 005037 005542
5789 021474 004737 030530
5790 021500 112737 000001 003124
5791 021506 112737 000001 003125
5792 021514 012737 177777 005534
5793
5794 021522 005037 005524
5795 021526 004737 037100
5796 021532 012704 000001
5797 021536 004737 033526
5798 021542 005701
5799 021544 100473
5800 021546 005065 000012
5801 021552 005701
5802 021554 001003
5803 021556 005400
5804 021560 010065 000012
5805 021564 013765 005576 000010 6$:
5806 021572 013701 005600
5807 021576 042701 177774
5808 021602 150165 000007
5809 021606 005737 056246
5810 021612 100010
5811
5812 021614 013737 005576 005602
5813 021622 013737 005600 005604
5814 021630 004737 030334
5815 021634 005737 005524
5816 021640 001026
5817
5818 021642 004737 040304
5819 021646 010403
5820 021650 004737 036156
5821 021654 112765 000123 000001

```

```

;*BLOCK BEYOND RWBUF+6000(OCTAL).
;*THE TEST BEGINS BY WRITING A SINGLE NUMBER INTO ALL LOCATIONS OF
;*EACH MEMORY BLOCK OF UP TO 64K WORDS. THIS NUMBER UNIQUELY DEFINES
;*EACH BLOCK. THEN WRITE EACH BLOCK ONTO DISK AT ADDRESS FC,FS,FT
;*(SCALED, IF NECESSARY TO AVOID PACK OVERFLOW OR DESTRUCTION OF THE BAD
;*SECTOR FILE). ZERO THE BLOCK IN MEMORY, AND THEN READ THE DATA
;*BACK INTO THE PROPER PHYSICAL ADDRESSES. DO THIS FOR ALL 64K BLOCKS,
;*AND THEN COMPARE THE CONTENTS OF ALL THE MEMORY TO EXPECTED VALUES.
;*TYPE FAILURES AS DESCRIBED ABOVE FOR THE FIRST 10(DEC) FAILING
;*LOCATIONS IN EACH 64K MEMORY BLOCK.
;*****
TST3: SCOPE
      MOV #3,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
      JSR PC,SETUP ;:SET UP FOR LOOP ON ERROR
      JSR PC,CHKITR ;:SEE IF ITER. NO. = 0 FOR THIS TEST
      JMP TST4 ;:JUMP TO NEXT TEST
;RETURN HERE FROM CHKITR IF TEST SHOULD BE RUN
      JSR PC,INITSS ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
      JSR PC,PREPKB ;:PREPARE FOR POSSIBLE KBD INPUT
;SAVE XXDP LOADER, IF PRESENT
      JSR PC,FNDXDP ;:COMPUTE STARTING ADRS OF XXDP
      MOV #RWBUF,XDPSAV ;:SET XXDP SAVE AREA ADDRESS
      CLR XDPSAV+2
      JSR PC,SAVXDP ;:GO SAVE XXDP LOADER
      MOV #1,XOVLAD ;:SET INDICATORS
      MOV #1,XDPSVD
      MOV #-1,PATRN ;:SET FLAG FOR SVPRMS
;INIT PARAMETERS, WRITE AND READ BACK ALL MEMORY BLOCKS
      CLR SELECT ;:INIT COMPARE FLAG
2$: JSR PC,SETUPM ;:INIT PARAMS
      MOV #1,R4 ;:INIT DATA WORD
4$: JSR PC,MXWRDC ;:COMPUTE MAX WRD CNT FOR THIS MA
      TST R1 ;:SEE IF ALL MEMORY TESTED YET
      BMI 30$ ;:BR IF DONE WRITING AND READING
      CLR P.WC(R5) ;:INIT WRD CNT TO 65,536(DEC)
      TST R1 ;:SEE IF WRD CNT SHOULD BE 65536
      BNE 6$ ;:BR IF YES
      NEG R0
      MOV R0,P.WC(R5) ;:SET WORD COUNT
6$: MOV MA,P.BALO(R5) ;:SET BA BITS 0-15
      MOV MA+2,R1 ;:GET MA BITS 16-21
      BIC #177774,R1 ;:MASK FOR LO 2 BITS
      BISB R1,P.BAHI(R5) ;:SET BA BITS 16,17
      TST $K11 ;:SEE IF MEM MGT PRESENT
      BPL 14$ ;:BR IF NO
;SET UP MEMORY MANAGEMENT
      MOV MA,PMA ;:SET BUFFER ADDRESS
      MOV MA+2,PMA+2
      JSR PC,PREPAR ;:PREPARE MEM MGT REG'S AND UNIBUS MAP
14$: TST SELECT ;:SEE IF COMPARES SHOULD BE DONE YET
      BNE 20$ ;:BR IF YES
;WRITE AND READ THIS MEM BLK ON RK06
16$: JSR PC,SVPRMS ;:SAVE PARAMS FOR THIS XFER
      MOV R4,R3 ;:GET DATA WORD
      JSR PC,LDMEM2 ;:LOAD MEMORY BLK WITH THIS WORD
      MOV #WRDATA,P.CMND(R5) ;:SET WRITE COMMAND

```

```

5822 021662 004737 040416 JSR PC,TRNSFR ;WRITE THE DATA
5823 021666 005003 CLR R3
5824 021670 004737 036156 JSR PC,LDMEM2 ;ZERO THE BLK IN MEMORY
5825 021674 112765 000121 000001 MOVB #R0DATA,P.CMND(R5) ;SET READ COMMAND
5826 021702 004737 040416 JSR PC,TRNSFR ;READ THIS DATA BACK
5827 021706 005204 INC R4 ;INCREMENT THE DATA WORD
5828 021710 004737 037306 JSR PC,INCRMA ;COMPUTE NEW MA TO TRY
5829 021714 000710 BR 4$ ;GO WRITE AND READ NEXT BLK
5830 ;PERFORM SOFTWARE COMPARES OF ALL MEMORY BLOCKS
5831 021716 010403 20$: MOV R4,R3 ;GET DATA WORD
5832 021720 004737 036322 JSR PC,CKMEM2 ;COMPARE THIS BLK TO GOOD DATA WORD
5833 021724 005204 INC R4 ;INCREMENT THE DATA WORD
5834 021726 004737 037306 JSR PC,INCRMA ;GET NEW MA TO TRY
5835 021732 000701 BR 4$ ;GO COMPARE NEXT MEM BLK
5836
5837 021734 005137 005524 30$: COM SELECT ;COMPLEMENT THE FLAG
5838 021740 001401 BEQ 50$ ;BR IF ALL DONE NOW
5839 021742 000671 BR 2$ ;BR IF COMPARES SHOULD BE DONE NOW
5840 021744 004737 030536 50$: JSR PC,GETXDP ;RESTORE SAVED XXDP
5841 021750 105737 003134 TSTB UBMPRS ;SEE IF UNIBUS MAP PRESENT
5842 021754 001402 BEQ 54$ ;BR IF NOT
5843 021756 005037 172516 CLR 0#SR3 ;DISABLE UNIBUS MAP
5844 021762
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
    
```

```

*****
*TEST 4 NPR/MEMORY DATA PATTERN TEST
*BEFORE TESTING, THE XXDP LOADER IS SAVED, IF PRESENT, AT ADRS
*RWBUFF. THEN, ALL PHYSICAL MEMORY LOCATIONS, STARTING AT THE
*FIRST ADRS OF THE NEXT 1K MEMORY BLOCK BEYOND RWBUF+6000(OCT),
* ARE EXERCISED WITH UP TO 16 DATA PATTERNS
* CHOSEN IN PARAMETER PT. (THESE ARE THE SAME PATTERNS
*PROVIDED FOR USE IN THE READ/WRITE DATA TEST). IF PT = 0, THE
*DATA PATTERNS WILL DEFAULT TO PATS 08,09,10, AND 11.
*MEMORY IS EXERCISED IN BLOCKS OF UP TO 64K WORDS, STARTING AT
*THE FIRST ADDRESS OF THE NEXT 1K MEMORY BLOCK BEYOND THE BUFFER.
*EACH BLOCK IS LOADED WITH DATA, AND WRITTEN ONTO DISK AT ADDRESS
*FC,FT,FS (SCALED TO AVOID PACK OVERFLOW OR DESTRUCTION OF THE BAD
*SECTOR FILE). THEN THE MEMORY BLOCK IS ZEROED AND READ BACK AND
*COMPARED. THIS IS DONE FOR EACH OF THE DATA PATTERNS CHOSEN,
*POSSIBLY INCLUDING THE USER-DEFINED PATTERN 15, IF SPECIFIED.
*****
    
```

```

5865 021762 000004 TST4: SCOPE
5866 021764 012737 000004 001324 MOV #4,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
5867 021772 004737 032130 JSR PC,SETUP ;:SET UP FOR LOOP ON ERROR
5868 021776 004737 032176 JSR PC,CHKITR ;:SEE IF ITER. NO. = 0 FOR THIS TEST
5869 022002 000137 022332 JMP TST5 ;:JUMP TO NEXT TEST
5870 ;RETURN HERE FROM CHKITR IF TEST SHOULD BE RUN
5871 022006 004737 030776 JSR PC,INITSS ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
5872 022012 004737 027302 JSR PC,PREPKB ;:PREPARE FOR POSSIBLE KBD INPUT
5873 ;SAVE XXDP LOADER, IF PRESENT
5874 022016 004737 030456 JSR PC,FNDXDP ;:COMPUTE STARTING ADR OF XXDP
5875 022022 012737 064640 005540 MOV #RWBUFF,XDPSAV ;:SET XXDP SAVE AREA ADR
5876 022030 005037 005542 CLR XDPSAV+2
5877 022034 004737 030530 JSR PC,SAVXDP ;:GO SAVE XXDP LOADER
    
```

```

5878 022040 112737 000001 003124      MOVB      #1,XOVLAD      ;SET INDICATORS
5879 022046 112737 000001 003125      MOVB      #1,XDPSVD
5880 022054 004737 034270          JSR      PC,LODP14      ;GENERATE PSEUDO-RAND PAT 14
5881 022060 013737 005666 005534      MOV      PT,PATRN      ;GET A COPY OF PT
5882 022066 001003          BNE      2$            ;BR IF PT NON-ZERO
5883 022070 012737 007400 005534      MOV      #7400,PATRN   ;SET DEFLT PATRNS = PAT 8,9,10,11
5884          ;INIT PARAMETERS, BEGIN TESTING
5885 022076 004737 037100      2$: JSR      PC,SETUPM    ;INIT PARAMS
5886 022102 004737 033526      4$: JSR      PC,MXWRDC   ;COMPUTE MAX WORD COUNT FOR THIS MA
5887 022106 005701          TST      R1            ;SEE IF ALL MEMORY TESTED YET
5888 022110 100501          BMI      50$          ;BR IF ALL DONE
5889 022112 005065 000012      CLR      P.WC(R5)      ;INIT WORD COUNT TO 65,536(DEC)
5890 022116 005701          TST      R1            ;SEE IF WRD CNT SHOULD BE 65,536
5891 022120 001003          BNE      6$            ;BR IF YES
5892 022122 005400          NEG      R0
5893 022124 010065 000012      MOV      R0,P.WC(R5)   ;SET WORD COUNT
5894 022130 013765 005576 000010 6$: MOV      MA,P.BALO(R5) ;SET BA BITS 0-15
5895 022136 013701 005600      MOV      MA+2,R1       ;GET MA BITS 16-21
5896 022142 042701 177774      BIC      #177774,R1    ;MASK FOR LO 2 BITS
5897 022146 150165 000007      BISB     R1,P.BAHI(R5) ;SET BA BITS 16,17
5898 022152 005737 056246      TST      $KT11        ;SEE IF MEM MGT PRESENT
5899 022156 100010          BPL      14$          ;BR IF NOT
5900          ;SET UP MEM MGT
5901 022160 013737 005576 005602      MOV      MA,PMA        ;SET BUFFER ADDRESS
5902 022166 013737 005600 005604      MOV      MA+2,PMA+2
5903 022174 004737 030334          JSR      PC,PREPAR     ;PREPARE MEM MGT REG'S AND UNIBUS MAP
5904 022200 012704 000001      14$: MOV      #1,R4      ;SET PATTERN BIT POINTER
5905 022204 030437 005534      16$: BIT      R4,PATRN  ;SEE IF THIS PATTERN IS CHOSEN
5906 022210 001003          BNE      20$          ;BR IF YES
5907 022212 006304      18$: ASL      R4        ;SHIFT TO POINT TO NEXT PATTERN
5908 022214 001434          BEQ      24$          ;BR IF NO MORE LEFT TO CHECK
5909 022216 000772          BR       16$          ;GO CHECK FOR ANOTHER PATTERN
5910 022220 010401      20$: MOV      R4,R1     ;GET A COPY OF BIT POINTER
5911          ;PERFORM THE TESTS
5912 022222 012737 022222 001110 22$: MOV      #,$LPERR   ;SET NEW LOOP ON ERROR ADRS
5913 022230 004737 032130          JSR      PC,SETUP     ;SET UP FOR LOOP ON ERROR
5914 022234 004737 040304          JSR      PC,SVPRMS    ;SAVE THE PARAMS FOR THIS XFER
5915 022240 004737 034504          JSR      PC,LODBUF    ;LOAD THIS MEM BLK WITH SELECTED DATA
5916 022244 112765 000123 000001      MOVB     #WRDATA,P.CMND(R5) ;SET WRITE DATA COMMAND
5917 022252 004737 040416          JSR      PC,TRNSFR    ;WRITE THE DATA
5918 022256 005003          CLR      R3           ;SET DATA WORD = 0
5919 022260 004737 036156          JSR      PC,LDMEM2    ;LOAD MEM BLK WITH ALL ZEROS
5920 022264 112765 000121 000001      MOVB     #R0DATA,P.CMND(R5) ;SET READ COMMAND
5921 022272 004737 040416          JSR      PC,TRNSFR    ;READ THE DATA FROM DISK
5922 022276 004737 034732          JSR      PC,CMPBUF    ;PERFORM SOFTWARE COMPARE OF DATA
5923 022302 104411          SCOPER          ;CHECK FOR INTERNAL LOOP ON ERROR
5924 022304 000742          BR       18$          ;CHECK FOR NEXT PATTERN
5925          ;GET NEW MA FOR NEXT TRANSFER
5926 022306 004737 037306      24$: JSR      PC,INCRMA  ;COMPUTE NEXT MA
5927 022312 000673          BR       4$           ;GO SEE IF MORE MEMORY TO TEST
5928 022314 004737 030536      50$: JSR      PC,GETXDP  ;RESTORE XXDP LOADER
5929 022320 105737 003134          TSTB     UBMPRS      ;SEE IF UNIBUS MAP PRESENT
5930 022324 001402          BEQ      54$          ;BR IF NOT PRESENT
5931 022326 005037 172516          CLR      @#SR3       ;DISABLE UNIBUS MAP
5932 022332          54$:
5933
    
```

5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989

```

*****
*TEST 5 UNIBUS CONTENTION TEST
*THE PURPOSE OF THIS TEST IS TO EXERCISE THE NPR CONTROL LOGIC
*WITHIN THE CONTROLLER IN SUCH A MANNER THAT A CONTENTION FOR
*UNIBUS CYCLES ARISES BETWEEN THE PROCESSOR AND THE RK06.
*
*FIRST, A WRITE DATA IS BEGUN AT ADDRESS FC, FT, SECTOR 0, WITH
*THE BUS ADDRESS INHIBIT BIT (BAI) SET IN THE CONTROLLER.
*USING A WORD COUNT OF 12000(OCT) OR 13000(OCT) (DEPENDING ON THE
*FORMAT), THE ENTIRE TRACK IS WRITTEN WITH THE FIRST WORD OF PATTERN
*15 (IF SELECTED) OR THE FIRST WORD OF PAT 13 (BY DEFAULT).
* THEN, AN INSTRUCTION LOOP IS ENTERED IN WHICH A NON-
*EXISTENT UNIBUS DEVICE ADDRESS IS REFERENCED ONCE PER PASS. THIS
*WILL RESULT IN A PROCESSOR SEIZURE OF THE UNIBUS, FOR 5-20 USEC,
*(DEPENDING ON THE PROCESSOR), FOLLOWED BY A NON-EXISTENT MEMORY
*PROCESSOR TRAP, ONCE PER PASS. THUS, FOR EACH PASS THROUGH THE
*INSTRUCTION LOOP THE CONTROLLER IS FORCED TO LOSE FROM ONE
*TO FOUR NPR BUS CYCLES. THE EXECUTION TIME OF THE LOOP DETERMINES THE
*REPETITION RATE OF THE NON-EXISTENT MEMORY REFERENCES, AND THIS
*IS CHOSEN TO ALLOW ENOUGH NPR'S SO THAT THERE SHOULD BE NO
*DATA LATE ERRORS IN A FAULT-FREE CONTROLLER.
*THE DATA IS THEN WRITE CHECKED FOR VERIFICATION, AND THE ABOVE
*TEST IS REPEATED USING A READ DATA COMMAND. NO ERRORS ARE EXPECTED.
*
*THE ABOVE OPERATIONS ARE REPEATED 15 MORE TIMES, USING THE NEXT WORD
*OF THE PROPER PATTERN (13 OR 15) EACH TIME, TO WRITE AND READ
*THE ENTIRE TRACK. IF A DATA LATE ERROR OCCURS ON A GIVEN TRANSFER,
*THAT DATA IS NOT WRITE-CHECKED OR COMPARED.

```

```

022332 000004
022334 012737 000005 001324
022342 034737 032130
022346 004737 032176
022352 000137 022636
022356 004737 030776
022362 004737 027302
022366 013765 005646 000002
022374 113765 005652 000005
022402 105065 000004
022406 012765 165000 000012
022414 105737 003115
022420 001403
022422 012765 166000 000012
022430 052765 100000 000014
022436 012700 006764
022442 005737 005666
022446 100402
022450 012700 006664
022454 012701 000020
022460 012703 000024

```

```

*****
TSTS: SCOPE
      MOV #5, $TESTN ; SET TEST NUMBER IN APT MAIL BOX
      JSR PC, SETUP ; SET UP FOR LOOP ON ERROR
      JSR PC, CHKTR ; SEE IF ITER. NO. = 0 FOR THIS TEST
      JMP TST6 ; JUMP TO NEXT TEST
; RETURN HERE FROM CHKTR IF TEST SHOULD BE RUN
      JSR PC, INITSS ; INITIALIZE DRIVER PARAMS AND SUB-SYS
      JSR PC, PREPK8 ; PREPARE FOR POSSIBLE KBD INPUT
; INITIALIZE PARAMETERS
      MOV FC, P.CYLN(R5) ; SET CYL = FC
      MOV FT, P.TRCK(R5) ; SET TRACK = FT
      CLRB P.SECT(R5) ; SET SECTOR = 0
      MOV #-13000, P.WC(R5) ; INIT WORD COUNT FOR FULL TRACK
      TSTB FORMAT ; DETERMINE THE FORMAT
      BEQ 4$ ; BR IF 22 SECTORS
      MOV #-12000, P.WC(R5) ; SET WORD COUNT FOR FULL TRACK
4$: BIS #DTBAII, P.PRST(R5) ; SET BUS ADDRESS INHIBIT
; WRITE DATA WITH ALLOWABLE UNIBUS CONTENTION, AND WRITE CHECK IT
      MOV #PAT15, RO ; SET DATA PATTERN ADDRESS
      TST PT ; SEE IF USER-DEFINED PATTERN 15 SELECTED
      BMI 14$ ; BR IF YES
      MOV #PAT13, RO ; SET DATA PATTERN 13 ADDRESS
14$: MOV #16., R1 ; SET COUNTER = 16
      MOV #20., R3 ; SET NON-EXISTENT MEMORY TIMER

```

```

5990 022464 010065 000010      16$:  MOV      RO,P.BALO(R5)  ;SET BA BITS
5991 022470 105037 003132      CLR      DLIFLG           ;CLEAR DATA LATE FLAG
5992 022474 112737 000001 003133  MOV      #1,NORTRY       ;SET "NO-RETRY" FLAG
5993 022502 112765 000123 000001  MOV      #WRDATA,P.CMND(R5) ;SET WRITE DATA COMMAND
5994 022510 004737 037204      JSR      PC,REFNEM       ;WRITE DATA DURING NEM REF.
5995 022514 105737 003132      TST      DLIFLG          ;SEE IF DATA LATE ERROR OCCURRED
5996 022520 001011      BNE      18$             ;BR IF NOT
5997 022522 032737 000002 005474  BIT      #BSERR,RECODE   ;SEE IF BAD SECTOR ERROR
5998 022530 001042      BNE      24$             ;BR IF YES
5999 022532 112765 000131 000001  MOV      #WRTCHK,P.CMND(R5) ;SET WRITE CHECK COMMAND
6000 022540 004737 040774      JSR      PC,DRVCAL       ;DO WRITE CHECK
6001      :READ DATA WITH ALLOWABLE UNIBUS CONTENTION, AND COMPARE LAST WORD
6002 022544 012765 064640 000010 18$:  MOV      #RWBUF,P.BALO(R5) ;SET BA = RWBUF ADDRESS
6003 022552 005037 064640      CLR      RWBUF           ;CLEAR THE BUFFER WORD
6004 022556 112765 000121 000001  MOV      #RDDATA,P.CMND(R5) ;SET READ COMMAND
6005 022564 004737 037204      JSR      PC,REFNEM       ;READ DATA DURING NEM REF.
6006 022570 105737 003132      TST      DLIFLG          ;SEE IF DATA LATE ERROR OCCURRED
6007 022574 001014      BNE      20$             ;BR IF YES
6008 022576 022037 064640      CMP      (RO)+,RWBUF     ;COMPARE LAST DATA WORD
6009 022602 001411      BEQ      20$             ;BR IF EQUAL
6010 022604 004737 042206      JSR      PC,REPSUP       ;GATHER STATUS FOR PRINTOUT
6011 022610 016037 177776 001174  MOV      -2(RO),SREG5    ;GOOD DATA WORD
6012 022616 013737 064640 001176  MOV      RWBUF,SREG6     ;BAD DATA WORD
6013 022624 104110      ERROR   110             ;"READ ERROR WHILE BAI SET"
6014      :SET UP FOR NEXT PASS THROUGH LOOP
6015 022626 005301 20$:  DEC      R1              ;SEE IF ALL DONE YET
6016 022630 001402      BEQ      24$             ;BR IF YES
6017 022632 000137 022464      JMP      16$             ;JUMP TO CONTINUE TESTING
6018 022636 24$:

```

```

6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045

```

```

*****
*TEST 6 MULTI-DRIVE INTERFERENCE TEST
*THIS TEST PERFORMS MULTI-DRIVE POSITIONING OPERATIONS, WHILE A
*LARGE DATA TRANSFER IS IN PROGRESS ON THE DRIVE UNDER TEST. FOR
*THE PURPOSE OF DETECTING PROBLEMS OF CONCURRENT DRIVE OPERATION.
*THE TEST IS RUN ONLY IF THERE IS MORE THAN 1 DRIVE ON THE SUBSYSTEM.
*
*THE TEST PROCEEDS AS FOLLOWS : IT IS FIRST DETERMINED WHICH DRIVE(S)
*(BESIDE THE ONE UNDER TEST) EXIST AND ARE OPERATIONAL. THESE DRIVES
*ARE THEN CLEARED AND A SEEK TO CYL 0 IS DONE ON EACH.NEXT, A SEEK IS
*DONE TO CYLINDER FC (SCALED, IF NECESSARY,) ON THE DRIVE UNDER
*TEST. A SEEK IS NOW BEGUN ON EACH OF THE OTHER DRIVES, EACH
*TO A DIFFERENT PSEUDO-RANDOM CYLINDER ADDRESS. THIS SHOULD REQUIRE
*FROM 10-67 MILLI-SEC TO COMPLETE. AS SOON AS THESE SEEKS ARE BEGUN,
*A WRITE DATA COMMAND IS BEGUN ON THE DRIVE UNDER TEST AT THE CURRENT
*CYLINDER (FC), SECTOR 0, TRACK 0, WITH THE BUS ADDRESS INCREMENT
*INHIBIT BIT SET IN THE CONTROLLER, AND WITH A WORD COUNT OF 15,616
*(DEC) WORDS IF 20 SECTOR FORMAT OR 17,152(DEC) WORDS IF 22 SECTOR
*FORMAT. THIS WORD COUNT AMOUNTS TO A FULL CYLINDER + 1 SECTOR OF DATA.
*THIS TRANSFER REQUIRES FROM 101-126 MILLI-SEC TO COMPLETE, AND IN
*THAT TIME, ALL THE OTHER DRIVES SHOULD HAVE COMPLETED POSITIONING.
*NEXT, A WRITE CHECK IS DONE TO VERIFY THE DATA WRITTEN ON THE DRIVE
*UNDER TEST, AND A CHECK IS MADE, TO INSURE THAT ALL OTHER DRIVES WHICH
*PERFORMED SEEKS WERE ABLE TO COMPLETE THEM SUCCESSFULLY, BY CHECKING

```

```

6046 ;*ATTENTION BITS AND COMPARING DRIVE CYLINDER ADDRESSES TO EXPECTED
6047 ;*VALUES.
6048 ;*****
6049 022636 000004 TST6: SCOPE
6050 022640 012737 000006 001324 MOV #6,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
6051 022646 004737 032130 JSR PC,SETUP ;:SET UP FOR LOOP ON ERROR
6052 022652 004737 032176 JSR PC,CHKITR ;:SEE IF ITER. NO. = 0 FOR THIS TEST
6053 022656 000137 023474 JMP $EOP ;:JUMP TO END-OF-PASS ROUTINE
6054 ;RETURN HERE FROM CHKITR IF TEST SHOULD BE RUN
6055 022662 004737 030776 JSR PC,INITSS ;:INITIALIZE DRIVER PARAMS AND SUB-SYS
6056 022666 004737 027302 JSR PC,PREPKB ;:PREPARE FOR POSSIBLE KBD INPUT
6057 ;CHECK DRVLST FOR OTHER AVAILABLE DRIVES, AND RECALIBRATE THEM
6058 022672 112765 000117 000001 MOVB #SEEK,P.CMND(R5) ;:SET SEEK COMMAND
6059 022700 005001 CLR R1 ;:INIT MULTIPLE-DRIVES FLAG
6060 022702 005000 CLR R0 ;:INIT DRIVE NO. TO 0
6061 022704 005065 000002 CLR P.CYLN(R5) ;:SET CYL = 0
6062 022710 022700 000010 4$: CMP #10,R0 ;:SEE IF ALL DRIVES CHECKED YET
6063 022714 001415 BEQ 8$ ;:BR IF ALL DONE
6064 022716 105760 005606 TSTB DRVLST(R0) ;:SEE IF THIS DRIVE IS MARKED IN LIST
6065 022722 001410 BEQ 6$ ;:BR IF NOT MARKED
6066 022724 120037 005500 CMPB R0,DRIVE ;:SEE IF MARKED DRIVE IS DRIVE UNDER TEST
6067 022730 001405 BEQ 6$ ;:BR IF YES
6068 022732 110065 000000 MOVB R0,P.DRVN(R5) ;:SET DRIVE NO. PARAMETER
6069 022736 004737 040774 JSR PC,DRVCAL ;:SEEK TO CYL 0 ON CURRENT DRIVE
6070 022742 005201 INC R1 ;:SET FLAG TO INDICATE MORE THAN 1 DRIVE
6071 022744 005200 6$: INC R0 ;:INCR DRIVE NO.
6072 022746 000760 BR 4$ ;:KEEP CHECKING DRIVES
6073 022750 005701 8$: TST R1 ;:SEE IF MORE THAN 1 DRIVE PRESENT
6074 022752 001002 BNE 9$ ;:BR IF YES, TO RUN TEST
6075 022754 000137 023474 JMP MULT11 ;:NO, SKIP TEST
6076 ;SEEK TO CYLINDER FC ON DRIVE UNDER TEST, SET PARAMS FOR DATA XFER
6077 022760 004737 030776 9$: JSR PC,INITSS ;:INIT S.S.
6078 022764 013765 005646 000002 MOV FC,P.CYLN(R5) ;:SET CYL = FC
6079 022772 005065 000004 CLR P.SECT(R5) ;:SET TRACK AND SECTOR = 0
6080 022776 012765 136400 000012 MOV #-17152.,P.WC(R5) ;:SET WC FOR 67(DEC) SECTORS
6081 023004 105737 003115 TSTB FORMAT ;:CHECK THE FORMAT
6082 023010 001403 BEQ 12$ ;:BR IF 22-SECTOR FORMAT
6083 023012 012765 141400 000012 MOV #-15616.,P.WC(R5) ;:SET WC FOR 61(DEC) SECTORS
6084 023020 012765 006664 000010 12$: MOV #PAT13,P.BALO(R5) ;:SET BA BITS
6085 023026 052765 100000 000014 BIS #DTBA11,P.PRST(R5) ;:SET BUS ADDRESS INCREMENT INHIBIT
6086 023034 112765 000117 000001 MOVB #SEEK,P.CMND(R5) ;:SET SEEK COMMAND
6087 023042 004737 040774 JSR PC,DRVCAL ;:SEEK TO CYL FC ON DRIVE UNDER TEST
6088 ;DO PSEUDO-RANDOM SEEKS ON ALL OTHER DRIVES
6089 023046 105037 003110 CLR B TSTING ;:INHIBIT ↑C, ↑Z ESCAPE
6090 023052 012701 000007 MOV #7,R1 ;:SET LOOP COUNTER
6091 023056 012700 003202 MOV #CYLLST,R0 ;:ADRS OF RAND CYL LIST
6092 023062 004737 034122 15$: JSR PC,RNDADR ;:GENERATE A PSEUDO-RANDOM CYL NO.
6093 023066 013720 005504 MOV CYLNDR,(R0)+ ;:SET CYL NO. IN TABLE
6094 023072 001002 BNE 16$ ;:BR IF NOT 0
6095 023074 005260 177776 INC -2(R0) ;:MAKE IT NON-ZERO
6096 023100 005301 16$: DEC R1 ;:DECREMENT LOOP COUNTER
6097 023102 001367 BNE 15$ ;:BR IF NOT DONE LOADING LIST YET
6098 023104 012701 003202 MOV #CYLLST,R1 ;:GET ADRS OF RAND CYL TABLE
6099 023110 010537 023160 MOV R5,19$ ;:SET PARAM BLK ADRS FOR DRIVER
6100 023114 005000 CLR R0 ;:INIT DRIVE NO. TO 0
6101 023116 022700 000010 18$: CMP #10,R0 ;:SEE IF ALL DRIVES CHECKED YET

```

B10

```

6102 023122 001421          BEQ      22$          :BR IF YES
6103 023124 105760 005606  TSTB   DRVLST(RO)   :SEE IF THIS DRIVE PRESENT
6104 023130 001414          BEQ      20$          :BR IF NO
6105 023132 120037 005500  CMPB   RO,DRIVE     :SEE IF THIS IS DRIVE UNDER TEST
6106 023136 001411          BEQ      20$          :BR IF YES
6107 023140 110065 000000  MOVB   RO,P.DRVN(R5) :SET DRIVE NO. PARAMETER
6108 023144 012165 000002  MOV    (R1)+,P.CYLN(R5) :SET CYLINDER NO.
6109 023150 004737 041110  JSR    PC,STACMD     :STORE PREV AND CURRENT CMNDS
6110 023154 004737 051120  JSR    PC,C.INIT     :GO START SEEK ON THIS DRIVE
6111 023160 000000          19$: .WORD          :P.B. ADRS GOES HERE
6112 023162 005200          20$: INC      RO      :INCR DRIVE NO.
6113 023164 000754          BR      18$          :BR TO SEEK ON NEXT DRIVE
6114          :WRITE DATA ON DRIVE UNDER TEST, WITH BAI SET
6115 023166 112765 000123 000001  22$: MOVB   #WRDATA,P.CMND(R5) :SET WRITE DATA COMMAND
6116 023174 113765 005500 000000  MOVB   DRIVE,P.DRVN(R5) :SET DRIVE NO.
6117 023202 013765 005646 000002  MOV    FC,P.CYLN(R5)   :SET CYL NO.
6118 023210 012737 023406 003034  MOV    #MULHDL,A.NORM  :SET SPECIAL DRIVER RETURN ADDRESS
6119 023216 004737 040774          JSR    PC,DRVCAL     :WRITE 1 CYL + 1 SECTOR AT CYL FC
6120 023222 032737 000002 005474  BIT    #BSERR,RECODE  :SEE IF BSE ERROR
6121 023230 001121          BNE     MULT11       :BR IF YES
6122 023232 112765 000131 000001  MOVB   #WRTCHK,P.CMND(R5) :SET WRITE CHECK COMMAND
6123 023240 004737 040774          JSR    PC,DRVCAL     :WRITE CHECK THE DATA WRITTEN
6124          :CHECK ATTENTIONS AND CHECK CYLINDER ADDRESSES FROM ALL OTHER DRIVES
6125 023244 112737 000001 003110  MOVB   #1,TSTING      :ALLOW ↑C, ↑Z ESCAPE
6126 023252 112765 000141 000001  MOVB   #RDSTAT,P.CMND(R5) :SET READ DRIVE STATUS COMMAND
6127 023260 012701 003202          MOV    #CYLLST,R1     :ADRS OF RAND CYL LIST
6128 023264 005000          CLR    RO             :INIT DRIVE NO.
6129 023266 022700 000010  26$: CMP    #10,RO      :SEE IF ALL DRIVES CHECKED YET
6130 023272 001500          BEQ    MULT11        :BR IF YES
6131 023274 105760 005606          TSTB   DRVLST(RO)   :SEE IF THIS DRIVE MARKED IN LIST
6132 023300 001440          BEQ    32$          :BR IF NOT
6133 023302 120037 005500          CMPB   RO,DRIVE     :SEE IF THIS IS DRIVE UNDER TEST
6134 023306 001435          BEQ    32$          :BR IF YES
6135 023310 136037 003072 003200  BITB   I.DRV(RO),SAVWRD :SEE IF GOT ATT'N FROM THIS DRIVE
6136 023316 001003          BNE     28$          :BR IF YES
6137 023320 004737 042206          JSR    PC,REPSUP    :PREPARE STATUS FOR PRINTOUT
6138 023324 104113          ERROR  113          :"NO ATTENTION ON SEEK"
6139 023326 110065 000000  28$: MOVB   RO,P.DRVN(R5) :SET THIS DRIVE NO.
6140 023332 004737 040774          JSR    PC,DRVCAL     :READ STATUS OF THIS DRIVE
6141 023336 016503 000052          MOV    P.B10(R5),R3  :GET STATUS BYTE B10
6142 023342 006003          ROR    R3           :GET CYL ADRS RIGHT-JUSTIFIED
6143 023344 006003          ROR    R3
6144 023346 006003          ROR    R3
6145 023350 006003          ROR    R3
6146 023352 042703 177000          BIC    #177000,R3    :CLEAR OFF UNUSED BITS
6147 023356 022103          CMP    (R1)+,R3     :COMPARE TO EXPECTED CYLINDER
6148 023360 001410          BEQ    32$          :BR IF EQUAL
6149 023362 016137 177776 001212  MOV    -2(R1),SREG14 :GOOD CYL NO.
6150 023370 010337 001214          MOV    R3,SREG15    :BAD CYL NO.
6151 023374 004737 042206          JSR    PC,REPSUP    :GATHER STATUS FOR PRINTOUT
6152 023400 104114          ERROR  114          :"DRIVE'S CYLINDER INCORRECT"
6153 023402 005200          32$: INC    RO      :INCREMENT DRIVE NO.
6154 023404 000730          BR     26$          :BR IF NOT DONE WITH ALL DRIVES YET
6155
6156
6157          :THIS IS THE NORMAL RETURN FROM THE DRIVER WHICH IS USED TO CHECK
          :THE RESULTS OF MULTI-DRIVE OPERATIONS.
    
```



```

6158 023406 032737 000200 002770 MULHDL: BIT #RDY,T.CS1 ;SEE IF CNTRLR RDY IS SET
6159 023414 001010 BNE 6$ ;BR IF RDY SET
6160 023416 004737 050520 JSR PC,I.CSTS ;READ CONTROLLER REGISTERS
6161 023422 013765 002770 000016 MOV T.CS1,P.CS1(R5) ;GET CS1 BITS
6162 023430 004737 042206 JSR PC,REPSUP ;PREPARE STATUS FOR PRINTOUT
6163 023434 104112 ERROR 112 ;"INTRPT WHEN CNTRLR NOT RDY"
6164 023436 013737 003004 003200 6$: MOV T.ASOF,SAVWRD ;SAVE ATT'N SUMMARY
6165 023444 000337 003200 SWAB SAVWRD ;GET ATT'N BITS IN BITS 0-7
6166 023450 112765 000177 000001 MOVB #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR COMMAND
6167 023456 012737 041220 003034 MOV #ERRFRE,A.NORM ;RESTORE NORMAL DRIVER RETURN ADDRESS
6168 023464 004737 040774 JSR PC,DRVCAL ;CLEAR SUB-SYS
6169 023470 000137 041220 JMP ERRFRE ;TAKE NORMAL RETURN

```

6170 023474 MULTI1:

6175 .SBTTL END OF PASS ROUTINE

```

6176 *****
6177 ;*INCREMENT THE PASS NUMBER ($PASS)
6178 ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
6179 ;*IF THERES A MONITOR GO TO IT
6180 ;*IF THERE ISN'T JUMP TO NEWPAS

```

```

6181 $EOP:
6182 SCOPE
6183 INC $DEVCT ;INCREMENT DEVICE COUNT FOR APT
6184 JMP NEWDRV ;GO SEE IF MORE DRIVES TO TEST ON THIS PASS
6185 DUNPAS: ;THIS IS TRULY THE END OF A PASS
6186 BIC #BIT6,$STKS ;DISABLE TTY KBD INTERRUPT
6187 CLR $STNM ;ZERO THE TEST NUMBER
6188 CLR $TIMES ;ZERO THE NUMBER OF ITERATIONS
6189 INC $PASS ;INCREMENT THE PASS NUMBER
6190 BIC #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
6191 DEC (PC)+ ;LOOP?
6192 $EOPCT: .WORD 1
6193 BGT $DOAGN ;:YES
6194 MOV (PC)+,$(PC)+ ;:RESTORE COUNTER
6195 $ENDCT: .WORD 1
6196 $EOPCT
6197 TYPE $SENDMG ;:TYPE "END PASS #"
6198 MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
6199 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
6200 TYPE $ENULL ;:TYPE A NULL CHARACTER
6201 $GET42: MOV $42,R0 ;:GET MONITOR ADDRESS
6202 BEQ $DOAGN ;:BRANCH IF NO MONITOR
6203 RESET ;:CLEAR THE WORLD
6204 $ENDAD: JSR PC,(R0) ;:GO TO MONITOR
6205 NOP ;:SAVE ROOM
6206 NOP ;:FOR
6207 NOP ;:ACT11
6208 $DOAGN:
6209 JMP $$(PC)+ ;:RETURN
6210 $RTNAD: .WORD NEWPAS
6211 $ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
6212
6213

```

D10

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2 MACY11 27(1006) 05-OCT-76 10:59 PAGE 121
DZR6NC.P11 05-OCT-76 10:07 END OF PASS ROUTINE

SEQ 0120

0214	023617	015	042412	042116	\$ENDMG: .ASCIZ <15><12>/END PASS #/
0215	023624	050040	051501	020123	
0216	023632	000043			
0217					
0218					

.SBTTL RK06 HEAD ALIGNMENT AID

E10

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 122
 RK06 HEAD ALIGNMENT AID

SEQ 0121

```

6219 023634 012737 000340 177776 ASTART: MOV #PR7, @#PS
6220 .SBTTL INITIALIZE THE COMMON TAGS
6221 ;;CLEAR THE COMMON TAGS (%CMTAG) AREA
6222 023642 012706 001100 MOV #SCMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
6223 023646 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
6224 023650 022706 001140 CMP #SWR, R6 ;;DONE?
6225 023654 001374 BNE -6 ;;LOOP BACK IF NO
6226 023656 012706 001100 MOV #STACK, SP ;;SETUP THE STACK POINTER
6227 ;;INITIALIZE A FEW VECTORS
6228 023662 012737 055504 000020 MOV #SSCOPE, @#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
6229 023670 012737 000340 000022 MOV #340, @#IOTVEC+2 ;;LEVEL 7
6230 023676 012737 055000 000030 MOV #ERROR, @#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
6231 023704 012737 000340 000032 MOV #340, @#EMTVEC+2 ;;LEVEL 7
6232 023712 012737 056612 000034 MOV #STRAP, @#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
6233 023720 012737 000340 000036 MOV #340, @#TRAPVEC+2 ;;LEVEL 7
6234 023726 012737 055350 000024 MOV #SPWRDN, @#PWRVEC ;;POWER FAILURE VECTOR
6235 023734 012737 000340 000026 MOV #340, @#PWRVEC+2 ;;LEVEL 7
6236 023742 013737 023546 023540 MOV SENDCT, $EOPCT ;;SETUP END-OF-PROGRAM COUNTER
6237 023750 005037 001304 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
6238 023754 005037 001306 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
6239 023760 112737 000001 001115 MOVB #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
6240 023766 012737 023766 001106 MOV #., $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
6241 023774 012737 023774 001110 MOV #., $LPERR ;;SETUP THE ERROR LOOP ADDRESS
6242 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
6243 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
6244 024002 013746 000004 MOV @#ERRVEC, -(SP) ;;SAVE ERROR VECTOR
6245 024006 012737 024042 000004 MOV #64$, @#ERRVEC ;;SET UP ERROR VECTOR
6246 024014 012737 177570 001140 MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
6247 024022 012737 177570 001142 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
6248 024030 022777 177777 155102 CMP #-1, @SWR ;;TRY TO REFERENCE HARDWARE SWR
6249 024036 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
6250 ;;AND THE HARDWARE SWR IS NOT = -1
6251 024040 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
6252 024042 012716 024050 64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
6253 024046 000002 RTI
6254 024050 012737 000176 001140 65$: MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
6255 024056 012737 000174 001142 MOV #DISPREG, DISPLAY
6256 024064 012637 000004 66$: MOV (SP)+, @#ERRVEC ;;RESTORE ERROR VECTOR
6257
6258 024070 005037 001326 CLR $PASS ;;CLEAR PASS COUNT
6259 024074 132737 000200 001341 BITB #APTSIZE, $ENVM ;;TEST USER SIZE UNDER APT
6260 024102 001403 BEQ 67$ ;;YES, USE NON-APT SWITCH
6261 024104 012737 001342 001140 MOV #SSWREG, SWR ;;NO, USE APT SWITCH REGISTER
6262 024112 67$:
6263 .SBTTL RK06 HEAD ALIGNMENT AID
6264 024112 000005 RESET ;;RESET UNIBUS
6265 024114 104401 007024 TYPE ,DZR6N ;;TYPE PROGRAM I.D. FOR PART 2
6266 024120 104401 007044 TYPE ,SUBVER
6267 024124 104401 007125 TYPE ,PART2
6268 024130 012737 000176 001140 MOV #SWREG, SWR ;;ADRS OF SOFTWARE SWR
6269 024136 005077 154776 CLR @SWR ;;MAKE SWR BITS ALL 0
6270 024142 012737 027142 000060 MOV #KBDHDL, @#TKVEC ;;LOAD VECTOR FOR TTY KBD
6271 024150 012737 000200 000062 MOV #PR4, @#TKVEC+2 ;;SET KBD PRIORITY = 4
6272 024156 013701 003030 MOV RKVEC, R1 ;;GET ADDRESS OF VECTOR STORAGE
6273 024162 012721 045620 MOV #I.INTR, (R1)+ ;;SET IT TO INTERRUPT HNDLR
6274 024166 012711 000340 MOV #PR7, (R1) ;;SET INTERRUPT HANDLER PR7
  
```

```

6275 024172 012737 000000 177776      MOV      #PRO_2#PS      ;ALLOW ALL INTERRUPTS
6276 024200 012737 000000 005646      MOV      #0,FC        ;SET FIRST CYL = 0
6277 024206 012737 000632 005650      MOV      #LSTCYL,LC   ;SET LAST CYL = 410(10)
6278
6279 024214                                GIVEID:
6280 024214 012737 176543 053246      MOV      #176543,$HINUM ;INITIALIZE PSEUDO-RANDOM NUMBERS
6281 024222 012737 123456 053250      MOV      #123456,$LONUM
6282 024230 105037 003106                                CLR      MDFLAG        ;CLEAR PARAM INP FLAG
6283 024234 105037 003110                                CLR      TSTING        ;CLEAR "RUNNING TESTS" FLAG
6284 024240 005037 005502                                CLR      STALLS        ;DON'T STALL ON OPERATIONS
6285 024244 012701 005222                                MOV      #PRVCMD,R1   ;ZERO OUT PREV COMMAND
6286 024250 012700 000006                                MOV      #6,R0
6287 024254 005021                                42$:  CLR      (R1)+
6288 024256 005300                                DEC      R0
6289 024260 001375                                BNE     42$
6290 024262 105037 003116                                CLR      ERRCNT        ;CLEAR ERROR COUNT FOR RESTARTS
6291 024266 104401 011435                                TYPE    ,IDENT         ;TYPE PROGRAM IDENTIFICATION
6292 024272 105737 003136                                TSTB   HLPOVL         ;SEE IF HELP FILE OVERLAID
6293 024276 001011                                BNE     ASKMOD         ;BR IF YES
6294 024300 104401 011502                                TYPE    ,FORHLP        ;ASK IF HELP DESIRED
6295 024304 104406                                RDCHR   ;READ RESPONSE
6296 024306 112601                                MOV     (SP)+,R1       ;GET CHARACTER
6297 024310 122701 000015                                CMP     #015,R1        ;SEE IF <CR> TYPED
6298 024314 001402                                BEQ     ASKMOD         ;BR IF NO HELP NEEDED
6299 024316 104401 065257                                TYPE    ,HLPFIL        ;HELP REQUESTED- TYPE INFO.
6300
6301
6302                                ;DETERMINE DESIRED OPERATIONAL MODE
6303 024322 104401 011537                                ASKMOD: TYPE    ,TYPMOD ;ASK FOR DESIRED OPERATIONAL MODE
6304 024326 004737 027302                                JSR     PC,PREPKB     ;PREPARE FOR KBD INPUT
6305 024332 005737 005522                                1$:  TST     INTCHR    ;CHECK FOR TTY INPUT
6306 024336 001775                                BEQ     1$            ;BR IF NO INPUT YET
6307 024340 013701 005522                                MOV     INTCHR,R1     ;GET INPUT CHAR INTO R1
6308 024344 022701 000022                                CMP     #022,R1      ;SEE IF (↑R) TYPED
6309 024350 001721                                BEQ     GIVEID        ;BR IF (↑R), TO RESTART
6310 024352 022701 000115                                CMP     #'M,R1        ;IS IT MANUAL MODE ?
6311 024356 001002                                BNE     2$            ;BR IF NOT MANUAL
6312 024360 000137 024404                                JMP     MANUAL        ;JUMP TO MANUAL MODE ROUTINE
6313 024364 022701 000101                                2$:  CMP     #'A,R1    ;IS IT AUTO MODE ?
6314 024370 001002                                BNE     3$            ;BR IF NOT AUTO
6315 024372 000137 025332                                JMP     AUTO          ;JUMP TO AUTO MODE ROUTINE
6316 024376 004737 027322                                3$:  JSR     PC,ECOBAD ;ECHO BAD INPUT
6317 024402 000747                                BR      ASKMOD        ;BR TO ASK FOR MODE AGAIN
6318
6319
6320
6321                                ;MANUAL SELECT MODE ROUTINE
6322 024404                                MANUAL:
6323 024404 104401 011600                                ASKDRV: TYPE    ,TYPMAN ;TYPE "MANUAL SELECT MODE"
6324 024410 104401 011633                                JSR     PC,PREPKB     ;ASK FOR DRIVE NUMBER
6325 024414 004737 027302                                JSR     PC,PREPKB     ;PREPARE FOR KBD INPUT
6326 024420 005737 005522                                1$:  TST     INTCHR    ;SEE IF ANY INPUT
6327 024424 001775                                BEQ     1$            ;BR IF NONE TO CHECK AGAIN
6328 024426 013701 005522                                MOV     INTCHR,R1     ;GET CHARACTER INTO R1
6329 024432 020127 000022                                CMP     R1,#022       ;TEST FOR (↑R) TYPED
6330 024436 001666                                BEQ     GIVEID        ;BR IF (↑R) TO RESTART
    
```

6331	024440	020127	000060		CMP	R1,#'0	:COMPARE TO ASCII 0
6332	024444	002410			BLT	2\$:BR IF <0 (BAD INPUT)
6333	024446	020127	000067		CMP	R1,#'7	:COMPARE TO ASCII 7
6334	024452	003005			BGT	2\$:BR IF >7 (BAD INPUT)
6335	024454	042701	000060		BIC	#'0,R1	:STRIP ASCII BITS
6336	024460	010137	005500		MOV	R1,DRIVE	:STORE DRIVE NUMBER
6337	024464	000403			BR	3\$:PROCEED WITH DESIRED DRIVE
6338	024466	004737	027322	2\$:	JSR	PC,ECOBAD	:ECHO BAD INPUT
6339	024472	000746			BR	ASKDRV	:GO BACK TO ASK AGAIN
6340	024474	004737	030776	3\$:	JSR	PC,INITSS	:INITIALIZE SUBSYSTEM
6341							:CHECK DESIRED DRIVE FOR PROPER STATUS
6342	024500	004737	031530		JSR	PC,CHKDRV	:CHECK DRIVE FOR ON-LINE,READY,WRITE LOCK
6343	024504	024410			ASKDRV		:ADDRESS OF ERROR RETURN FOR CHKDRV
6344	024506	004737	033602		JSR	PC,DRVSR	:TYPE "DRIVE SER. NO. XXX"
6345							:SET UP DESIRED SUB-MODE OF OPERATION
6346	024512			SUBMOD:			
6347	024512	105037	011434		CLRB	VERIFY	:CLEAR VERIFY MODE FLAG
6348	024516	104401	012416		TYPE	ENTMOD	:ASK FOR ALIGN OR EXERCISE SUB-MODE
6349	024522	004737	027302		JSR	PC,PREPKB	:PREPARE FOR KBD INPUT
6350	024526	005737	005522	1\$:	TST	INTCHR	:SEE IF ANY INPUT
6351	024532	001775			BEQ	1\$:BR IF NONE TO CHECK AGAIN
6352	024534	013701	005522		MOV	INTCHR,R1	:GET CHAR INTO R1
6353	024540	020127	000022		CMP	R1,#022	:SEE IF (↑R) TYPED
6354	024544	001002			BNE	2\$:BR IF NOT (↑R)
6355	024546	000137	024214		JMP	GIVEID	:JUMP TO RESTART
6356	024552	020127	000003	2\$:	CMP	R1,#003	:SEE IF (↑C) TYPED
6357	024556	001002			BNE	3\$:BR IF NOT (↑C)
6358	024560	000137	024410		JMP	ASKDRV	:JUMP TO ASK FOR DRIVE AGAIN
6359	024564	020127	000101	3\$:	CMP	R1,#'A	:SEE IF ALIGNMENT REQUESTED
6360	024570	001415			BEQ	MANAL	:BR IF MANUAL ALIGNMENT REQUESTED
6361	024572	020127	000105		CMP	R1,#'E	:SEE IF EXERCISES REQUESTED
6362	024576	001002			BNE	4\$:BR IF NOT EXERCISES
6363	024600	000137	025076		JMP	MANEX	:JUMP TO DO MANUAL EXERCISES
6364	024604	020127	000126	4\$:	CMP	R1,#'V	:SEE IF VERIFY REQUESTED
6365	024610	001002			BNE	6\$:BR IF NOT VERIFY
6366	024612	000137	025060		JMP	MANVR	:JUMP TO VERIFY MODE
6367	024616	004737	027322	6\$:	JSR	PC,ECOBAD	:ECHO BAD INPUT
6368	024622	000733			BR	SUBMOD	:ASK FOR SUB-MODE AGAIN
6369							
6370							
6371							
6372							
6373	024624						
6374	024624	104401	012556				
6375							
6376	024630	104401	012724				
6377	024634	004737	027302				
6378	024640	005737	005522	1\$:	TST	INTCHR	:SEE IF ANY INPUT
6379	024644	001775			BEQ	1\$:BR IF NONE TO CHECK AGAIN
6380	024646	013701	005522		MOV	INTCHR,R1	:GET CHAR INTO R1
6381	024652	020127	000022		CMP	R1,#022	:SEE IF (↑R)
6382	024656	001007			BNE	2\$:BR IF NOT (↑R)
6383	024660	112765	000113	000001	MOVB	#RECAL,P.CMND(R5)	:SET RECALIBRATE COMMAND
6384	024666	004737	040774		JSR	PC,DRVCL	:RECALIBRATE DRIVE
6385	024672	000137	024214		JMP	GIVEID	:JUMP TO RESTART
6386	024676	020127	000003	2\$:	CMP	R1,#003	:SEE IF (↑C)

H10

```

6387 024702 001007          BNE      3$          ;BR IF NOT (1C)
6388 024704 112765 000113 000001  MOVB   #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6389 024712 004737 040774          JSR    PC,DRV CAL      ;RECALIBRATE DRIVE
6390 024716 000137 024410          JMP    ASK DRV        ;JUMP TO ASK FOR DRIVE AGAIN
6391 024722 020127 000032          3$:    CMP    R1,#032    ;SEE IF (1Z)
6392 024726 001007          BNE      4$          ;BR IF NOT (1Z)
6393 024730 112765 000113 000001  MOVB   #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6394 024736 004737 040774          JSR    PC,DRV CAL      ;RECALIBRATE DRIVE
6395 024742 000137 024512          JMP    SUBMOD         ;JUMP TO ASK FOR NEW MANUAL SUB-MODE
6396 024746 020127 000060          4$:    CMP    R1,#'0    ;COMPARE TRACK TO ASCII 0
6397 024752 002410          BLT     5$          ;BR IF <0 (BAD INPUT)
6398 024754 020127 000062          CMP    R1,#'2        ;COMPARE TRACK TO ASCII 2
6399 024760 003005          BGT     5$          ;BR IF >2 (BAD INPUT)
6400 024762 042701 000060          BIC    #'0,R1        ;STRIP ASCII BITS
6401 024766 010137 005520          MOV    R1,TRACK      ;STORE TRACK (HEAD) NUMBER
6402 024772 000404          BR     6$          ;GO SELECT HEAD
6403 024774 004737 027322          5$:    JSR    PC,ECOBAD   ;ECHO BAD INPUT
6404 025000 000137 024630          JMP    ASKHED        ;ASK AGAIN FOR HEAD NUMBER
6405          ;UNLOAD HEADS, AND WHEN READY, START SPINDLE AND SEEK TO ALN CYL
6406 025004 105737 011434          6$:    TSTB   VERIFY    ;SEE IF VERIFY MODE
6407 025010 001002          BNE      8$          ;BR IF VERIFY
6408 025012 004737 032024          JSR    PC,WAIT4R     ;UNLOAD HEADS, WAIT FOR <R> TYPED, LOAD
6409 025016 113765 005520 000005  8$:    MOVB   TRACK,P.TRCK(R5) ;SET DESIRED HEAD NUMBER
6410 025024 004737 031770          JSR    PC,ALNSEK    ;SEEK TO ALIGNMENT CYLINDER
6411 025030 104401 012653          TYPE   HEDPOS       ;TYPE HEADS POSITIONED MSG
6412 025034 013701 005520          MOV    TRACK,R1     ;GET BINARY TRACK NO.
6413 025040 152701 000060          BISB   #'0,R1       ;CONVERT TO ASCII
6414 025044 110137 013012          MOVB   R1,HEADNO    ;GET HEAD NO. INTO MSG BUFF.
6415 025050 104401 013005          TYPE   HEDSEL       ;TYPE HEAD SELECTED MSG
6416 025054 000137 024630          JMP    ASKHED        ;GO BACK TO ASK FOR NEW HEAD SELECT
6417
6418
6419
6420          ;MANUAL SELECT VERIFY MODE
6421 025060 112737 000001 011434  MANVR: MOVB   #1,VERIFY    ;SET VERIFY MODE FLAG
6422 025066 104401 012616          TYPE   MANVR        ;REPORT MANUAL VERIFY MODE
6423 025072 000137 024630          JMP    ASKHED        ;GO ASK FOR DESIRED HEAD
6424
6425
6426
6427          ;MANUAL RANDOM SEEK EXERCISE ROUTINE
6428 025076          MANEX:
6429 025076 004737 032024          JSR    PC,WAIT4R     ;UNLOAD HEADS, WAIT FOR <R> TYPED, LOAD
6430 025102 013701 005500          MOV    DRIVE,R1     ;GET DRIVE NUMBER
6431 025106 052701 000060          BIS    #'0,R1       ;CONVERT TO ASCII
6432 025112 110137 012550          MOVB   R1,DR1EXR    ;PUT DRIVE NUMBER INTO OUTPUT BUFFER
6433 025116 104401 012471          TYPE   MANEXR       ;TYPE RANDOM SEEKS MSG
6434 025122 012700 016514          MOV    #RNDLNG,R0   ;INITIALIZE RANDOM SEEK COUNTER
6435 025126 112765 000117 000001  MOVB   #SEEK,P.CMND(R5) ;SET SEEK COMMAND
6436 025134 004737 027302          1$:    JSR    PC,PREPKB  ;PREPARE FOR KBD INPUT
6437 025140 004737 034122          2$:    JSR    PC,RNDADR  ;SELECT RANDOM CYLINDER ADDRESS
6438 025144 013765 005504 000002  MOV    CYLNR,P.CYLN(R5) ;SET RANDOM CYLINDER ADDRESS
6439 025152 004737 040774          JSR    PC,DRV CAL    ;DO A RANDOM SEEK
6440          ;CHECK FOR TTY INPUT DURING EXERCISES
6441 025156 005737 005522          TST    INTCHR       ;SEE IF ANY INPUT
6442 025162 001016          BNE      3$          ;BR IF A CHAR WAS TYPED
    
```

6443	025164	005300		DEC	R0	: DECREMENT SEEK COUNTER
6444	025166	001364		BNE	2\$: BR IF NOT DONE SEEKING YET
6445	025170	042777	000100 153746	BIC	#BIT6, \$STKS	: DISABLE KBD INTERRUPT
6446	025176	112765	000113 000001	MOVB	#RECAL.P.CMND(R5)	: SET RECALIBRATE COMMAND
6447	025204	004737	040774	JSR	PC, DRVCAL	: RECALIBRATE DRIVE
6448	025210	104401	001310	TYPE	\$BELL	: RING BELL TO SIGNIFY END OF EXERCISES
6449	025214	000137	024512	JMP	SUBMOD	: GO ASK FOR NEW MANUAL SUB-MODE
6450	025220	013701	005522	3\$: MOV	INTCHR, R1	: GET CHARACTER INTO R1
6451	025224	020127	000022	CMP	R1, #022	: SEE IF (↑R)
6452	025230	001007		BNE	4\$: BR IF NOT (↑R)
6453	025232	112765	000113 000001	MOVB	#RECAL.P.CMND(R5)	: SET RECALIBRATE COMMAND
6454	025240	004737	040774	JSR	PC, DRVCAL	: RECALIBRATE DRIVE
6455	025244	000137	024214	JMP	GIVEID	: JUMP TO RESTART
6456	025250	020127	000003	4\$: CMP	R1, #003	: SEE IF (↑C)
6457	025254	001007		BNE	5\$: BR IF NOT (↑C)
6458	025256	112765	000113 000001	MOVB	#RECAL.P.CMND(R5)	: SET RECALIBRATE COMMAND
6459	025264	004737	040774	JSR	PC, DRVCAL	: RECALIBRATE DRIVE
6460	025270	000137	024410	JMP	ASKDRV	: JUMP TO ASK FOR NEW DRIVE NUMBER
6461	025274	020127	000032	5\$: CMP	R1, #032	: SEE IF (↑Z)
6462	025300	001007		BNE	6\$: BR IF NOT (↑Z)
6463	025302	112765	000113 000001	MOVB	#RECAL.P.CMND(R5)	: SET RECALIBRATE COMMAND
6464	025310	004737	040774	JSR	PC, DRVCAL	: RECALIBRATE DRIVE
6465	025314	000137	024512	JMP	SUBMOD	: JUMP TO ASK FOR NEW MANUAL SUB-MODE
6466	025320	004737	027322	6\$: JSR	PC, ECOBAD	: ECHO BAD INPUT
6467	025324	104401	012471	TYPE	MANEXR	: VERIFY STILL IN MANUAL EXERCISES
6468	025330	000701		BR	1\$: BR TO CONTINUE EXERCISES

6469						
6470						
6471						
6472						
6473	025332			: AUTO SELECT MODE ROUTINE		
6474	025332	004737	030776	AUTO: JSR	PC, INITSS	: INITIALIZE SUBSYSTEM
6475	025336	104401	013027	TYPE	TYPAUT	: TYPE "AUTO SELECT MODE"
6476	025342	105037	011434	1\$: CLR	VERIFY	: CLEAR VERIFY MODE FLAG
6477	025346	104401	012416	TYPE	ENTMOD	: ASK FOR ALIGN, VERIFY, OR EXERCISE
6478	025352	004737	027302	JSR	PC, PREPKB	: PREPARE FOR KEYBOARD INPUT
6479	025356	005737	005522	2\$: TST	INTCHR	: SEE IF ANY INPUT
6480	025362	001775		BEQ	2\$: BR IF NONE, TO CHECK AGAIN
6481	025364	013701	005522	MOV	INTCHR, R1	: GET CHAR INTO R1
6482	025370	020127	000022	CMP	R1, #022	: SEE IF (↑R) TYPED
6483	025374	001002		BNE	3\$: BR IF NOT (↑R)
6484	025376	000137	024214	JMP	GIVEID	: JUMP TO RESTART
6485	025402	020127	000101	3\$: CMP	R1, #'A	: SEE IF AUTO ALIGNMENT REQUESTED
6486	025406	001417		BEQ	AUTAL	: BR IF AUTO ALIGNMENT REQUESTED
6487	025410	020127	000105	CMP	R1, #'E	: SEE IF AUTO EXERCISES REQUESTED
6488	025414	001002		BNE	4\$: BR IF EXERCISES NOT REQUESTED
6489	025416	000137	026442	JMP	AUTEX	: JUMP TO DO AUTO EXERCISES
6490	025422	020127	000126	4\$: CMP	R1, #'V	: SEE IF VERIFY REQUESTED
6491	025426	001004		BNE	6\$: BR IF NOT VERIFY
6492	025430	112737	000001 011434	MOVB	#1, VERIFY	: SET VERIFY MODE FLAG
6493	025436	000403		BR	AUTAL	: PROCEED IN VERIFY MODE
6494	025440	004737	027322	6\$: JSR	PC, ECOBAD	: ECHO BAD INPUT
6495	025444	000736		BR	1\$: BR TO ASK FOR A OR E AGAIN
6496						
6497						
6498						

RK06 HEAD ALIGNMENT AID

```

6499          :AUTO ALIGNMENT MODE
6500          AUTAL:
6501 025446   105737 011434   TSTB   VERIFY           ;SEE IF AUTO VERIFY MODE
6502 025446   001403           BEQ     3$              ;BR IF NOT
6503 025454   104401 013116   TYPE   ,AUTVRF         ;REPORT AUTO VERIFY MODE
6504 025460   000402           BR      4$              ;
6505 025462   104401 013060   3$:    TYPE   ,AUTALN       ;TYPE ALIGNMENT MESSAGE
6506 025466   004737 027302   4$:    JSR    PC,PREPKB    ;PREPARE FOR POSSIBLE KBD INPUT
6507 025472   005037 003014   CLR    T,MR2           ;CLEAR MR2 STORAGE WORD
6508 025476   005037 005524   CLR    SELECT          ;INITIALIZE AUTO-SELECTED DRIVE INDICATOR
6509 025502   012737 000377 005526   MOV    #377,ONLINE     ;INIT. OLD ONLINE DRIVE INDICATOR
6510 025510   012737 000377 005530   MOV    #377,NEWON      ;INIT. NEW ONLINE DRIVE INDICATOR
6511 025516   012737 000377 005532   MOV    #377,SCRACH     ;INITIALIZE LAST DRIVE INDICATOR
6512 025524   005737 005522   DRVLUP: TST   INTCHR        ;SEE IF ANY KBD INPUT
6513 025530   001445           BEQ     6$              ;BR IF NO INPUT
6514 025532   113765 005532 000000   MOVB   SCRACH,P.DRVN(R5) ;DRIVE NO. FOR POSSIBLE RECALIBRATE
6515 025540   123727 005522 000032   CMPB   INTCHR,#032     ;SEE IF (↑Z) TYPED
6516 025546   001013           BNE     2$              ;BR IF NOT (↑Z)
6517 025550   022737 000377 005532   CMP    #377,SCRACH     ;SEE IF ANY DRIVE SELECTED YET
6518 025556   001405           BEQ     1$              ;BR IF NONE
6519 025560   112765 000113 000001   MOVB   #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6520 025566   004737 040774           JSR    PC,DRVCAL       ;RECALIBRATE DRIVE
6521 025572   000137 025332           JMP    AUTO             ;RESTART AUTO MODE
6522 025576   123727 005522 000022   1$:    CMPB   INTCHR,#022   ;SEE IF (↑R) TYPED
6523 025604   001013           BNE     4$              ;BR IF NOT (↑R)
6524 025606   022737 000377 005532   CMP    #377,SCRACH     ;SEE IF ANY DRIVE SELECTED YET
6525 025614   001405           BEQ     3$              ;BR IF NONE
6526 025616   112765 000113 000001   MOVB   #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6527 025624   004737 040774           JSR    PC,DRVCAL       ;RECALIBRATE DRIVE
6528 025630   000137 024214           JMP    GIVEID           ;RESTART ALIGNMENT AID
6529 025634   004737 027322           JSR    PC,ECOBAD        ;ECHO BAD INPUT
6530 025640   004737 027302           JSR    PC,PREPKB       ;PREPARE FOR POSSIBLE KBD INPUT
6531 025644   005037 005500           CLR    DRIVE           ;ZERO THE DRIVE NUMBER
6532 025650   012700 000001           MOV    #1,R0           ;INITIALIZE DRIVE BIT MASK
6533 025654   012737 027064 003036   8$:    MOV    #NEDHDL,A.ABNL   ;SET NED ABNORMAL RETURN ADDRESS
6534 025662   050037 005530           BIS    R0,NEWON        ;SET ON-LINE BIT FOR THIS DRIVE
6535          ;SELECT CURRENT DRIVE, CHECK FOR NON-EXISTENT DRIVE (NED) INDICATION
6536 025666   113765 005500 000000   MOVB   DRIVE,P.DRVN(R5) ;SET DRIVE NUMBER
6537 025674   112765 000101 000001   MOVB   #SELDRV,P.CMND(R5) ;SET SELECT DRIVE COMMAND
6538 025702   004737 040774           JSR    PC,DRVCAL       ;SELECT THIS DRIVE
6539 025706   022737 000007 005500   CMP    #7,DRIVE        ;SEE IF WE JUST CHECKED DRIVE 7
6540 025714   001405           BEQ     10$            ;BR IF IT WAS DRIVE 7
6541 025716   005237 005500           INC    DRIVE           ;ADD 1 TO DRIVE NUMBER
6542 025722   000241           CLC                    ;CLEAR CARRY BEFORE ROTATE
6543 025724   006100           ROL    R0              ;SHIFT BIT POINTER
6544 025726   000752           BR     8$              ;BR TO SELECT NEXT DRIVE
6545 025730   013701 005530           10$:   MOV    NEWON,R1
6546          ;***** TO BE REMOVED *****
6547 025734   023737 005530 005526   CMP    NEWON,ONLINE
6548 025742   001670           BEQ    DRVLUP
6549 025744   012703 000001           MOV    #1,R3
6550 025750   012700 177777           LOOP1: MOV    #177777,R0   ;NUMBER OF 200 MILLI-SEC STALLS
6551 025754   005300           LOOP2: DEC    R0
6552 025756   005700           TST   R0
6553 025760   001375           BNE   LOOP2
6554 025762   005303           DEC   R3

```


K10

```

6555 025764 001371          BNE      LOOP1
6556                      ;*****
6557 025766 043701 005526  BIC      ONLINE,R1      ;GET CHANGED ONLINE BITS
6558 025772 013737 005530 005526  MOV      NEWON,ONLINE   ;UPDATE ONLINE DRIVE BITS
6559 026000 005701          TST      R1              ;SEE IF ANY DRIVE JUST WENT ON-LINE
6560 026002 001650          BEQ      DRVLUP         ;BR IF NO DRIVE JUST WENT ONLINE
6561                      ;SERVICE THE DRIVE WHICH WAS JUST SELECTED
6562 026004 012737 042466 003036  MOV      #ERRHDL,A.ABNL ;RESTORE USUAL ERROR HANDLER ADDRESS
6563 026012 005037 005500          CLR      DRIVE         ;INITIALIZE DRIVE NO.
6564 026016 010103          MOV      R1,R3         ;COPY NEW SELECTED DRIVE BITS
6565 026020 012700 000001          MOV      #1,R0         ;INITIALIZE BIT POINTER
6566 026024 030003 12$:      BIT      R0,R3         ;SEE IF THIS BIT IS SET
6567 026026 001005          BNE      14$          ;BR IF THIS BIT SET
6568 026030 005237 005500          INC      DRIVE         ;INCREMENT DRIVE NO.
6569 026034 000241          CLC                     ;CLEAR CARRY BEFORE ROTATE
6570 026036 006100          ROL      R0            ;SHIFT BIT POINTER
6571 026040 000771          BR       12$          ;TRY AGAIN
6572 026042 040003 14$:      BIC      R0,R3         ;CLEAR OUT THIS BIT
6573 026044 001415          BEQ      16$          ;BR IF ONLY ONE DRIVE SELECTED
6574                      ;ERROR -- MORE THAN ONE DRIVE SELECTED SIMULTANEOUSLY
6575 026046 104401 012317          TYPE    ,MULSEL        ;REPORT ERROR
6576 026052 042762 000100 000000  BIC      #IE,RKCS1(R2) ;INHIBIT RK06 INTERRUPT
6577 026060 000000          HALT                    ;HALT FOR MANUAL INTERVENTION
6578                      ;PRESS 'CONT' TO PROCEED
6579 026062 112765 000177 000001  MOVB     #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR COMMAND
6580 026070 004737 040774          JSR     PC,DRVCAL      ;CLEAR THE SUBSYSTEM
6581 026074 000137 025446          JMP     AUTAL          ;RESTART AUTO ALIGNMENT
6582                      ;CONTINUE WITH SELECTED DRIVE
6583 026100 020137 005524 16$:      CMP      R1,SELECT     ;SEE IF STILL ALIGNING SAME DRIVE
6584 026104 001506          BEQ     26$          ;BR IF SAME DRIVE
6585                      ;PROCEED WITH NEWLY-SELECTED DRIVE
6586 026106 022737 000377 005532  CMP      #377,SCRACH   ;SEE IF THIS IS FIRST DRIVE SELECTED
6587 026114 001421          BEQ     23$          ;BR IF FIRST DRIVE
6588 026116 113765 005532 000000  MOVB     SCRACH,P.DRVN(R5) ;GET LAST DRIVE NUMBER
6589 026124 112765 000141 000001  MOVB     #RDSTAT,P.CMND(R5) ;SET READ STATUS COMMAND
6590 026132 004737 040774          JSR     PC,DRVCAL      ;READ STATUS OF PREVIOUS DRIVE
6591 026136 032765 000040 000044  BIT      #S.HOHM,P.A01(R5) ;SEE IF HEADS ARE UNLOADED
6592 026144 001405          BEQ     23$          ;BR IF HEADS NOT UNLOADED
6593 026146 112765 000111 000001  MOVB     #SRTSPL,P.CMND(R5) ;SET START SPINDLE COMMAND
6594 026154 004737 040774          JSR     PC,DRVCAL      ;START SPINDLE ON PREVIOUS DRIVE
6595 026160 113737 005500 005532 23$:      MOVB     DRIVE,SCRACH  ;STORE DRIVE NO. FOR LATER CLEANUP STRT SPL
6596 026166 010137 005524          MOV      R1,SELECT     ;UPDATE SELECTED DRIVE NUMBER
6597 026172 013701 005500          MOV      DRIVE,R1      ;GET DRIVE NUMBER
6598 026176 052701 000060          BIS      #'0,R1        ;CONVERT TO ASCII
6599 026202 110137 013160          MOVB     R1,DRVSEL     ;GET DRIVE NUMBER INTO OUTPUT BUFFER
6600 026206 104401 013151          TYPE    ,DRSEL        ;TYPE "DRIVE X SELECTED"
6601 026212 004737 031530          JSR     PC,CHKDRV     ;CHECK DRIVE FOR RDY, WRITE PROT.
6602 026216 025446          AUTAL                    ;ERROR RETURN ADDRESS
6603 026220 112737 000377 011433  MOVB     #377,UNLOD    ;SET DRIVE UNLOADED INDICATOR
6604 026226 012737 000002 005520  MOV      #2,TRACK      ;SET TRACK = 2
6605 026234 113765 005500 000000  MOVB     DRIVE,P.DRVN(R5) ;SET DRIVE NUMBER
6606 026242 105737 011434 24$:      TSTB     VERIFY       ;SEE IF AUTO VERIFY
6607 026246 001016          BNE     32$          ;SKIP UNLOAD IF VERIFY
6608 026250 112765 000107 000001  MOVB     #UNLOAD,P.CMND(R5) ;SET UNLOAD COMMAND
6609 026256 004737 040774          JSR     PC,DRVCAL      ;UNLOAD THE DRIVE
6610 026262 112765 000141 000001  MOVB     #RDSTAT,P.CMND(R5) ;SET READ STATUS COMMAND
  
```

```

6611 026270 004737 040774 000044 25$: JSR PC,DRVCAL ;READ STATUS OF DRIVE
6612 026274 032765 000040 000044 BIT #S.HDHM,P.A01(R5) ;SEE IF HEADS ARE UNLOADED YET
6613 026302 001772 BEQ 25$ ;BR IF NOT YET
6614 026304 112765 000177 000001 32$: MOVB #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR COMMAND
6615 026312 004737 040774 JSR PC,DRVCAL ;CLEAR THE S.S.
6616 026316 000137 025524 JMP DRVLUP ;GO BACK TO SCAN DRIVES AGAIN
6617 :PROCEED WITH SAME DRIVE
6618 026322 113765 005500 000000 26$: MOVB DRIVE,P.DRVN(R5) ;SET DRIVE NO.
6619 026330 105137 011433 COMB UNLOD ;COMPLEMENT UNLOAD INDICATOR
6620 026334 001342 BNE 24$ ;BR IF DRIVE SHOULD BE UNLOADED
6621 026336 105737 011434 TSTB VERIFY ;SEE IF AUTO VERIFY
6622 026342 001005 BNE 33$ ;SKIP START SPL IF VERIFY
6623 026344 112765 000111 000001 MOVB #SRTSPL,P.CMND(R5) ;SET START SPINDLE COMMAND
6624 026352 004737 040774 JSR PC,DRVCAL ;START SPINDLE ON THIS DRIVE
6625 026356 022737 000002 005520 33$: CMP #2,TRACK ;SEE IF LAST TRACK WAS 2
6626 026364 001003 BNE 28$ ;BR IF IT WAS NOT 2
6627 026366 005037 005520 CLR TRACK ;SET TRACK = 0
6628 026372 000402 BR 30$ ;GO SEEK TO ALIGNMENT CYLINDER
6629 026374 005237 005520 28$: INC TRACK ;INCREMENT TRACK NUMBER
6630 026400 113765 005520 000005 30$: MOVB TRACK,P.TRCK(R5) ;SET TRACK NO.
6631 026406 004737 031770 JSR PC,ALNSEK ;SEEK IN INCREMENTS TO ALIGN. CYL
6632 026412 104401 012653 TYPE HEDPOS ;TYPE HEADS POSITIONED MSG
6633 026416 013701 005520 MOV TRACK,R1 ;GET CURRENT TRACK NO.
6634 026422 152701 000060 BISB #'0,R1 ;CONVERT TO ASCII
6635 026426 110137 013012 MOVB R1,HEADNO ;GET TRACK INTO MSG BUFFER
6636 026432 104401 013005 TYPE HEDSEL ;TYPE "HEAD X SELECTED"
6637 026436 000137 025524 JMP DRVLUP ;GO BACK TO SCAN DRIVES AGAIN
6638
6639
6640 :AUTO RANDOM SEEK EXERCISE ROUTINE
6641 AUTEX:
6642 026442 004737 030776 JSR PC,INITSS ;INITIALIZE SUBSYSTEM
6643 026446 004737 027302 JSR PC,PREPKB ;PREPARE FOR KBD INPUT
6644 026452 104401 013176 TYPE AUTEXR ;ASK FOR SHORT OR LONG SEEK EXERCISES
6645 026456 005737 005522 1$: TST INTCHR ;SEE IF ANY INPUT
6646 026462 001775 BEQ 1$ ;BR IF NO INPUT
6647 026464 023727 005522 000022 CMP INTCHR,#022 ;SEE IF (↑R) TYPED
6648 026472 001002 BNE 2$ ;BR IF NOT (↑R)
6649 026474 000137 024214 JMP GIVEID ;JUMP TO RESTART ALIGNMENT AID
6650 026500 023727 005522 000032 2$: CMP INTCHR,#032 ;SEE IF (↑Z) TYPED
6651 026506 001002 BNE 3$ ;BR IF NOT (↑Z)
6652 026510 000137 025332 JMP AUTO ;JUMP TO RESTART AUTO MODE
6653 026514 023727 005522 000123 3$: CMP INTCHR,#'S ;SEE IF SHORT EXERCISES DESIRED
6654 026522 001004 BNE 4$ ;BR IF NOT SHORT
6655 026524 012737 002734 005532 MOV #RND SHT,SCRACH ;SET 1 MINUTE SEEK COUNT
6656 026532 000413 BR 6$ ;PROCEED WITH SHORT EXERCISES
6657 026534 023727 005522 000114 4$: CMP INTCHR,#'L ;SEE IF LONG EXERCISES DESIRED
6658 026542 001004 BNE 5$ ;BR IF NOT LONG
6659 026544 012737 016514 005532 MOV #RND LNG,SCRACH ;SET 5 MINUTE SEEK COUNT
6660 026552 000403 BR 6$ ;PROCEED WITH LONG EXERCISES
6661 026554 004737 027322 5$: JSR PC,ECOBAD ;ECHO BAD INPUT
6662 026560 000730 BR AUTEX ;BR TO ASK AGAIN
6663 026562 005037 005500 6$: CLR DRIVE ;SET DRIVE = 0
6664 :SELECT CURRENT DRIVE AND CHECK FOR NON-EXISTENT DRIVE INDICATION
6665 026566 012737 027064 003036 8$: MOV #NED HDL,A.ABNL ;SET NED ABNORMAL RETURN ADDRESS
6666 026574 113765 005500 000000 MOVB DRIVE,P.DRVN(R5) ;SET DRIVE NUMBER

```

M10

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 130
 RK06 HEAD ALIGNMENT AID

SEQ 0129

```

6667 026602 112765 000101 000001      MOVB  #SELDV,P.CMND(R5) ;SET SELECT DRIVE COMMAND
6668 026610 012737 000377 005530      MOV   #377,NEWON        ;INITIALIZE ON-LINE INDICATOR
6669 026616 004737 040774      JSR   PC,DRVCL         ;SELECT THIS DRIVE
6670 026622 012737 042466 003036      MOV   #ERRHDL,A.ABNL   ;RESTORE ABNORMAL DRIVER RETURN ADDRESS
6671 026630 022737 000377 005530      CMP   #377,NEWON      ;SEE IF NED INDICATION ON THIS DRIVE
6672 026636 001075      BNE   18$             ;BR IF NED, TO SKIP THIS DRIVE
6673      ;CHECK DRIVE FOR RDY, WRITE PROTECT
6674 026640 004737 031530      JSR   PC,CHKDRV        ;CHECK THIS DRIVE
6675 026644 026442      AUTEX                  ;ERROR RETURN ADDRESS FOR CHKDRV
6676      ;PROCEED WITH SEEK EXERCISES ON THIS DRIVE
6677 026646 013701 005500      MOV   DRIVE,R1        ;GET DRIVE NUMBER
6678 026652 052701 000060      BIS   #'0,R1          ;CONVERT TO ASCII
6679 026656 110137 013310      MOVB  R1,AUTODR       ;GET DRIVE NUMBER INTO OUTPUT BUFFER
6680 026662 104401 013267      TYPE  ,AUTOEX         ;TYPE "EXERCISING DRIVE X"
6681 026666 112765 000117 000001      MOVB  #SEEK,P.CMND(R5) ;SET SEEK COMMAND
6682 026674 013700 005532      MOV   SCRACH,RO       ;SEEK COUNT
6683 026700 004737 027302      JSR   PC,PREPKB       ;PREPARE FOR POSSIBLE KBD INPUT
6684 026704 004737 034122      JSR   PC,RNDADR       ;SELECT RANDOM CYLINDER ADDRESS
6685 026710 013765 005504 000002      MOV   CYLNDR,P.CYLN(R5) ;SET RANDOM CYLINDER ADDRESS
6686 026716 004737 040774      JSR   PC,DRVCL        ;DO A RANDOM SEEK
6687 026722 004737 005522      TST   INTCHR          ;SEE IF ANY KBD INPUT
6688 026726 004432      BEQ   16$             ;BR IF NO INPUT
6689 026730 023727 005522 000022      CMP   INTCHR,#022    ;SEE IF (↑R) TYPED
6690 026736 001007      BNE   12$             ;BR IF NOT (↑R)
6691 026740 112765 000113 000001      MOVB  #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6692 026746 004737 040774      JSR   PC,DRVCL        ;RECALIBRATE DRIVE
6693 026752 000137 024214      JMP   GIVEID          ;JUMP TO RESTART ALIGNMENT AID
6694 026756 023727 005522 000032      CMP   INTCHR,#032    ;SEE IF (↑Z) TYPED
6695 026764 001007      BNE   14$             ;BR IF NOT (↑Z)
6696 026766 112765 000113 000001      MOVB  #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6697 026774 004737 040774      JSR   PC,DRVCL        ;RECALIBRATE DRIVE
6698 027000 000137 025332      JMP   AUTO           ;JUMP TO RESTART AUTO MODE
6699 027004 004737 027322      JSR   PC,ECOBAD       ;ECHO BAD INPUT
6700 027010 004737 027302      JSR   PC,PREPKB       ;ENABLE KBD INPUT AGAIN
6701 027014 005303      DEC   RO              ;DECREMENT SEEK COUNT
6702 027016 001330      BNE   10$             ;BR IF NOT DONE WITH THIS DRIVE
6703 027020 112765 000113 000001      MOVB  #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
6704 027026 004737 040774      JSR   PC,DRVCL        ;RECALIBRATE DRIVE
6705 027032 005237 005500      INC   DRIVE           ;INCREMENT DRIVE NUMBER
6706 027036 022737 000010 005500      CMP   #10,DRIVE      ;SEE IF DONE WITH ALL DRIVES
6707 027044 001250      BNE   8$              ;BR IF MORE DRIVES TO EXERCISE YET
6708 027046 042777 000100 152070      BIC   #BIT6,@$TKS    ;DISABLE KBD INTERRUPT
6709 027054 104401 001310      TYPE  ,SBELL          ;RING BELL AT END OF AUTO-EXERCISES
6710 027060 000137 025332      JMP   AUTO           ;RESTART AUTO MODE
6711
6712
6713      ;THIS IS THE ABNORMAL RETURN FROM THE DRIVER,
6714      ; WHICH IS USED WHEN NED INDICATION IS EXPECTED (AND VALID).
6715      ; (NED = NON-EXISTENT DRIVE)
6716 027064 032765 010000 000020      NEDHDL: BIT #NED,P.CS2(R5) ;SEE IF NED ON DRIVE SELECT
6717 027072 001002      BNE   1$              ;BR IF NED SET
6718 027074 000137 042466      JMP   ERRHDL         ;GO HANDLE OTHER ERROR
6719 027100 010046      1$: MOV  RO,-(SP)      ;SAVE RO,R1
6720 027102 010146      MOV  R1,-(SP)
6721 027104 012700 000001      MOV  #1,RO           ;SET BIT POINTER
6722 027110 005001      CLR  R1              ;CLEAR COUNTER
    
```

```

6723 027112 120137 005500 2$: CMPB R1,DRIVE ;SEE IF R1 = CURRENT DRIVE NUMBER
6724 027116 001403 BEQ 3$ ;BR IF =
6725 027120 005201 INC R1 ;INCREMENT COUNTER
6726 027122 006300 ASL R0 ;SHIFT BIT POINTER
6727 027124 000772 BR 2$ ;TRY AGAIN
6728 027126 040037 005530 3$: BIC R0,NEWON ;CLEAR ONLINE BIT FOR THIS DRIVE
6729 027132 012601 MOV (SP)+,R1 ;RESTORE R0,R1
6730 027134 012600 MOV (SP)+,R0
6731 027136 000137 044702 JMP RETNML ;TAKE NORMAL RETURN

```

```

;*****
;SBTTL KBDHDL - TTY KEYBOARD INTERRUPT HANDLER
;*****

```

```

6738 027142 010146 KBDHDL: MOV R1, -(SP) ;SAVE R1 ON STACK
6739 027144 017701 151776 MOV @STKB,R1 ;READ A CHAR FROM KBD BUFFER
6740 027150 042701 177600 BIC #177600,R1 ;MASK OUT UNUSED BITS
6741 027154 120127 000172 CMPB R1,#172 ;SEE IF LOWER CASE TYPED
6742 027160 003005 BGT 20$ ;BR IF NOT
6743 027162 120127 000141 CMPB R1,#141
6744 027166 002402 BLT 20$ ;BR IF NOT
6745 027170 042701 000040 BIC #BITS,R1 ;MAKE IT UPPER CASE
6746 027174 010137 005522 20$: MOV R1,INTCHR ;SAVE INPUT CHARACTER
6747 027200 122701 000003 CMPB #003,R1 ;SEE IF (↑C) TYPED
6748 027204 001007 BNE 2$ ;BR IF NOT (↑C)
6749 027206 104401 013360 TYPE ,CNTRLC ;ECHO (↑C)
6750 027212 042777 000100 151724 1$: BIC #BIT6,@STKS ;CLEAR KBD INTERRUPT ENABLE BIT
6751 027220 012601 MOV (SP)+,R1 ;RESTORE R1
6752 027222 000002 RTI ;RETURN
6753 027224 122701 000032 2$: CMPB #032,R1 ;SEE IF (↑Z) TYPED
6754 027230 001003 BNE 3$ ;BR IF NOT (↑Z)
6755 027232 104401 013365 TYPE ,CNTRLZ ;ECHO (↑Z)
6756 027236 000765 BR 1$ ;RETURN
6757 027240 122701 000022 3$: CMPB #022,R1 ;SEE IF (↑R) TYPED
6758 027244 001003 BNE 4$ ;BR IF NOT (↑R)
6759 027246 104401 013372 TYPE ,CNTRLR ;ECHO (↑R)
6760 027252 000757 BR 1$ ;RETURN
6761 027254 122701 000007 4$: CMPB #007,R1 ;SEE IF (↑G) TYPED
6762 027260 001003 BNE 5$ ;BR IF NOT (↑G)
6763 027262 104401 013404 TYPE ,CNTRLG ;ECHO (↑G)
6764 027266 000751 BR 1$ ;RETURN
6765 027270 104401 005522 5$: TYPE ,INTCHR ;ECHO INPUT
6766 027274 104401 001315 TYPE ,$CRLF ;DO <CR> AND <LF>
6767 027300 000744 BR 1$ ;RETURN

```

```

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

```

```

6768
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778

```

B11

```

6779
6780
6781 027302 005077 151640
6782 027306 005037 005522
6783 027312 052777 000100 151624
6784 027320 000207
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797 027322 104401 005522
6798 027326 104401 001314
6799 027332 000207
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6810 027334 104407
6811 027336 012737 000200 056246
6812 027344 004737 056210
6813 027350 005737 056246
6814 027354 100406
6815 027356 013737 056512 005572
6816 027364 005037 005574
6817 027370 000427
6818
6819 027372 013700 056514
6820 027376 005001
6821 027400 012702 000006
6822 027404 000241
6823 027406 006100
6824 027410 006101
6825 027412 005302
6826 027414 001373
6827
6828 027416 063700 056512
6829 027422 005501
6830 027424 010037 005572
6831 027430 010137 005574
6832 027434 042701 000003
6833 027440 001403
6834 027442 112737 000001 003134

```

```

*****
PREPKB: CLR      Q$TKB      :CLEAR KBD BUFFER AND DONE BIT
          CLR      INTCHR     :CLEAR TTY INPUT WORD
          BIS      #BIT6,Q$TKS :ENABLE KBD INTERRUPT
          RTS      PC         :RETURN
*****

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

ECOBAD: TYPE      INTCHR     :ECHO BAD CHARACTER
          TYPE      $QUES     :TYPE '?' AND <CR>, <LF>
          RTS      PC         :RETURN
*****

:*****
:*SIZMEM - SIZE MEMORY, SET LIMITS
:*THIS SUBROUTINE SIZES MEMORY BY CALLING SUBROUTINE $SIZE, AND
:*IT USES THE VALUES RETURNED, TO SET THE LIMITS ON PARAMETER MA.
:*IT ALSO TYPES "LAST PHYS MEM ADR = XXXXXXXX" (LEAD ZEROS SUPRS'D).
:*AND SETS THE FLAG UBMPRS = 1 IF 22-BIT ADDRESSES ARE USED (11/70).
:*****
SIZMEM: SAVREG      :SAVE R0-R5
          MOV      #200,$KT11 :SET MEM MGT KEY FOR $SIZE
          JSR      PC,$SIZE   :SIZE MEMORY
          TST      $KT11      :SEE IF MEM MGT PRESENT
          BMI      B$         :BR IF MEM MGT PRESENT
          MOV      $LSTAD,MAHILM :SET MEM LIMIT LO BITS
          CLR      MAHILM+2    :CLEAR MEM LIMIT HI BITS
          BR      16$         :GO TYPE LAST ADDRESS
B$:      SHIFT SAF LEFT 6, AND PUT IN R1 -R0
          MOV      $LSTBK,R0   :LO BITS
          CLR      R1         :HI BITS
          MOV      #6,R2      :SET LOOP COUNT = 6
12$:    CLC                 :ROTATE LOOP
          ROL      R0
          ROL      R1
          DEC     R2
          BNE     12$
:ADD VIRTUAL ADDRESS TO SHIFTED SAF TO GET PHYSICAL ADDRESS
          ADD     $LSTAD,R0    :ADD LO BITS
          ADC     R1          :HI BITS
          MOV     R0,MAHILM   :SET LO BITS OF MEM LIMIT
          MOV     R1,MAHILM+2 :SET HI BITS OF MEM LIMIT
          BIC     #3,R1      :CLEAR ADRS BITS 16,17
          BEQ     16$        :BR IF NOT 22-BIT ADDRESSES
          MOVB    #1,UBMPRS  :SET "UNIBUS MAP PRESENT" FLAG

```

```

6835
6836 027450 104401 007131
6837 027454 012746 005572
6838 027460 004737 053752
6839 027464 004737 054266
6840 027470 104401 001315
6841 027474 104401 001315
6842 027500 104410
6843 027502 000207
6844
6845
6846
6847
6848
6849
6850
6851
6852
6853 027504 016646 000002
6854 027510 104403
6855 027512 006
6856 027513 000
6857 027514 104401 007337
6858 027520 004737 032272
6859 027524 027556
6860 027526 027556
6861 027530 027556
6862 027532 005700
6863 027534 001001
6864 027536 000207
6865 027540 020027 000006
6866 027544 003407
6867 027546 104401 005262
6868 027552 104401 001314
6869 027556 162716 000010
6870 027562 000207
6871 027564 012746 005262
6872 027570 004737 053014
6873 027574 027546
6874 027576 012600
6875 027600 005737 053146
6876 027604 001360
6877 027606 010066 000002
6878 027612 000207
6879
6880
6881
6882
6883
6884
6885
6886
6887
6888
6889 027614 022737 000176 001140
6890 027622 001010

```

```

:TYPE "LAST PHYS MEM ADR = XXXXXXXX"
16$: TYPE ,LSTMEM ;TYPE "LAST PHYS MEM ADR ="
MOV ,#MAHILM,-(SP) ;PUT POINTER ON STACK
JSR PC,3,#$DB20 ;CONVERT BINARY TO OCTAL ASCII
JSR PC,3,#$SUPRS ;TYPE "XXXXXXXX"
TYPE ,$SCLF ;TYPE <CR>,<LF>
TYPE ,$SCLF
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

```

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

```

```

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

```

```

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

```

```

GTSWRG: CMP #SWREG,SWR ;SEE IF SOFTWARE SWR SELECTED
BNE 6$ ;BR IF NOT

```

69091	027624	013746	000176	MOV	SWREG, -(SP)	:PUT OLD VALUE ON STACK
69092	027630	104401	007330	TYPE	SWMSG	:TYPE "SWR = "
69093	027634	004737	027504	JSR	PC, GETPRM	:TYPE OLD, GET NEW SWREG VALUE
69094	027640	012637	000176	MOV	(SP)+, SWREG	:STORE NEW VALUE
69095	027644	000207		6S: RTS	PC	:RETURN

```

*****
:INITMM - INITIALIZE MEMORY MANAGEMENT REGISTERS
:THIS SUBROUTINE INITIALIZES KIPAR'S AND KIPDR'S, FOR
:CONTIGUOUS 4K PAGES, STARTING AT PHYSICAL ADDRESS 0.
:KIPAR7 IS LOADED WITH 177600, TO PRESERVE THE I/O PAGE.
:22-BIT MODE AND THE UNIBUS MAP ARE ENABLED, IF PRESENT.
: BUT MEM MGT IS NOT TURNED ON IN THIS SUBROUTINE. HOWEVER,
: MEM MGT TRAPS ARE ENABLED.

```

69098				INITMM: SAVREG		:SAVE R0-R5	
69099	027646	104407		CLR	R1	:INIT FOR PAR LOADING	
69100	027650	005001		MOV	#KIPAR0, R2	:ADDR OF FIRST PAR	
69101	027652	012702	172340	MOV	#8, R3	:LOAD 8 PAR'S AND 8 PDR'S	
69102	027656	012703	000010	MOV	#77406, -40(R2)	:PDR = 4K UP, READ/WRITE	
69103	027662	012762	077406	4S: MOV	R1, (R2)+	:LOAD A PAR	
69104	027670	010122		ADD	#200, R1	:UPDATE FOR NEXT PAR	
69105	027672	062701	000200	DEC	R3	:DECREMENT LOOP COUNTER	
69106	027676	005303		BNE	4S	:LOOP UNTIL ALL 8 ARE LOADED	
69107	027700	001370		MOV	#177600, -(R2)	:SET UP KIPAR7 FOR I/O PAGE	
69108	027702	012742	177600	TSTB	UBMPRS	:SEE IF 22-BIT ADDRESSES	
69109	027706	105737	003134	BEQ	10S	:BR IF NOT	
69110	027712	001403		MOV	#60, @#SR3	:ENABLE 22-BIT MODE, AND UNIBUS MAP	
69111	027714	012737	000060	172516	MOV	#BIT9, @#SR0	:ENABLE KT11 TRAPS
69112	027722	012737	001000	177572	RESREG	:RESTORE R0-R5	
69113	027730	104410		RTS	PC	:RETURN	
69114	027732	000207					

```

*****
:SERVICE ROUTINE FOR MEM MGT TRAPS

```

69300				KTERHD: JSR	PC, REPSUP	:GATHER STATUS FOR PRINTOUT	
69301	027734	004737	042206	MOV	@#SR0, \$REG5	:GET SRO	
69302	027740	013737	177572	001174	MOV	@#SR1, \$REG6	:GET SR1
69303	027746	013737	177574	001176	MOV	@#SR2, \$REG7	:GET SR2
69304	027754	013737	177576	001200	MOV	#8, @#ERRVEC	:SET TIME-OUT VECTOR
69305	027762	012737	030006	000004	MOV	#PR7, @#ERRVEC+2	
69306	027770	012737	000340	000006	MOV	@#SR3, \$REG10	:GET SR3
69307	027776	013737	172516	001202	BR	10S	
69308	030004	000406		8S: CMP	(SP)+, (SP)+	:CLEAN UP STACK	
69309	030006	022626		MOV	#3, DF30+18.	:FIX PRINTOUT FOR NO SR3	
69310	030010	112737	000003	064612	CLRB	DH704+11.	
69311	030016	105037	063253	10S: ERROR	121	:KT11 FAILURE	
69312	030022	104121		JMP	HLPARG	:ABORT !!!	
69313	030024	000137	044476				

6944
6945
6946

E11

```

6947
6948
6949
6950 030030 010146
6951 030032 013746 000004
6952 030036 013746 000006
6953 030042 012737 030064 000004
6954 030050 012701 172100
6955 030054 005011
6956 030056 012711 000001
6957 030062 000401
6958 030064 022626
6959 030066 062701 000002
6960 030072 020127 172140
6961 030076 001366
6962 030100 012637 000006
6963 030104 012637 000004
6964 030110 012601
6965 030112 000207
6966
6967
6968
6969
6970
6971
6972 030114 010146
6973 030116 013746 000004
6974 030122 013746 000006
6975 030126 004737 042206
6976 030132 016637 000006 001174
6977 030140 012737 030264 000004
6978 030146 012737 000340 000006
6979
6980 030154 013737 177740 001176
6981 030162 013737 177742 001200
6982 030170 013737 177744 001202
6983 030176 012737 063307 064634
6984 030204 112737 000004 064636
6985 030212 104122
6986 030214 005737 001200
6987 030220 001011
6988 030222 023727 001176 072640
6989 030230 103005
6990 030232 105737 003135
6991 030236 001002
6992 030240 000137 044476
6993 030244 012637 000006
6994 030250 012637 000004
6995 030254 012601
6996 030256 004737 030030
6997 030262 000002
6998
6999 030264 022626
7000 030266 012737 030316 000004
7001 030274 012701 172100
7002 030300 011137 001200

```

```

*****
*ENBCSR - ENABLE MEMORY CSR'S FOR PARITY ERRORS
*****
ENBCSR: MOV R1, -(SP) ;SAVE R1
MOV @#ERRVEC, -(SP) ;SAVE OLD VECTORS
MOV @#ERRVEC+2, -(SP)
MOV @#ERRVEC, @#ERRVEC ;SET TIME-OUT VECTOR
MOV @#MEMCSR, R1 ;ADRS OF MEMORY CSR'S
4$: CLR (R1) ;ZERO THIS CSR
MOV @#BIT0, (R1) ;SET ENABLE IN THIS CSR
BR 10$
8$: CMP (SP)+, (SP)+ ;CLEAN UP THE STACK
10$: ADD #2, R1 ;INCREMENT TO NEXT CSR ADRS
CMP R1, @#MEMCSR+40 ;SEE IF DONE CHECKING YET
BNE 4$ ;BR IF NOT
MOV (SP)+, @#ERRVEC+2 ;RESTORE OLD VECTORS
MOV (SP)+, @#ERRVEC
MOV (SP)+, R1 ;RESTORE R1
RTS PC ;RETURN

```

```

*****
*SERVICE ROUTINE FOR MEM PARITY ERRORS
*****
MPEHRD: MOV R1, -(SP) ;SAVE R1
MOV @#ERRVEC, -(SP) ;SAVE OLD VECTORS
MOV @#ERRVEC+2, -(SP)
JSR PC, REPSUP ;GATHER STATUS FOR PRINTOUT
MOV @#PC, $REG5 ;GET PC OF ERROR
MOV @#10$, @#ERRVEC ;SET T.O. VECTOR
MOV @#PR7, @#ERRVEC+2
;HANDLE 11/70 MEMORY PARITY ERROR
MOV @#LOERAD, $REG6 ;LOW ERROR ADRS REG
MOV @#HIERAD, $REG7 ;HI ERROR ADRS REG
MOV @#MEMSYS, $REG10 ;MEMORY SYSTEM REG
MOV @#DH707, DF31+16. ;FIX ERROR MSG FOR 11/70
MOV @#4, DF31+18.
ERROR 122 ;11/70 MEM PARITY ERROR
TST $REG7 ;SEE IF BAD MEMORY IS IN PROGRAM AREA
BNE 6$ ;BR IF NOT
CMP $REG6, #RWBUFF+E000 ;SEE IF BAD MEM IS IN PROG. AREA
BHS 6$ ;BR IF NOT
4$: TSTB MEMABT ;SEE IF ABORT DESIRED
BNE 6$ ;BR IF NOT
6$: JMP HLTPRG ;ABORT !!!
MOV (SP)+, @#ERRVEC+2 ;RESTORE T.O. VECTOR
MOV (SP)+, @#ERRVEC
MOV (SP)+, R1 ;RESTORE R1
JSR PC, ENBCSR ;GO CLEAR AND ENABLE CSR'S
RTI ;RETURN
;HANDLE ALL OTHER MEMORY PARITY ERRORS (NON-11/70)
10$: CMP (SP)+, (SP)+ ;CLEAN UP THE STACK
MOV @#14$, @#ERRVEC ;SET T.O. VECTOR
MOV @#MEMCSR, R1 ;GET FIRST CSR ADDRESS
12$: MOV (R1), $REG7 ;CHECK FOR A MEMORY CSR

```


F11

MD-11-DZR&N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR&NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 136
GTSWRG - OPEN SOFTWARE SWITCH REGISTER FOR MODIFICATION

SEQ 0135

7003 030304 100005
7004 030306 010137 001176
7005 030312 104122
7006 030314 000746
7007 030316 022626
7008 030320 062701 000002
7009 030324 020127 172140
7010 030330 103763
7011 030332 000744

BPL 16\$;BR IF NO ERROR SET HERE
MOV R1,\$REG6 ;GET CSR ADDRESS
ERROR 122 ;MEMORY PARITY ERROR (NON-11/70)
BR 4\$;GO SEE IF SHOULD ABORT
14\$: CMP (SP)+,(SP)+ ;CLEAN UP STACK
16\$: ADD #2,R1 ;INCR R1 TO POINT TO NEXT CSR
CMP R1,#MEMCSR+40 ;SEE IF DONE CHECKING
BLO 12\$;BR IF NOT
BR 6\$;RETURN

7012
7013
7014
7015
7016
7017
7018
7019
7020

*PREPAR - PREPARE MEM MGT FOR RELOCATION
*THIS SUBROUTINE CALLS INITMM, AND THEN SETS UP CONSTANT
*FOR KIPAR6 IN SAVPAR, AND LOADS THE UNIBUS MAP REGISTERS,
*IF PRESENT.

7021 030334 104407
7022 030336 004737 027646
7023 030342 013700 005602
7024 030346 042700 017777
7025 030352 013701 005604
7026 030356 010003
7027 030360 010104
7028 030362 006100
7029 030364 006101
7030 030366 006100
7031 030370 006101
7032 030372 000301
7033 030374 006100
7034 030376 106001
7035 030400 010137 003176
7036 030404 105737 003134
7037 030410 001420
7038
7039 030412 012700 170200
7040 030416 012701 000037
7041 030422 010320
7042 030424 010420
7043 030426 062703 020000
7044 030432 005504
7045 030434 077106
7046 030436 042765 160000 000010
7047 030444 142765 000003 000007
7048 030452 104410
7049 030454 000207
7050
7051
7052
7053
7054
7055
7056
7057 030456 005737 005574
7058 030462 001404

PREPAR: SAVREG ;SAVE R0-R5
JSR PC,INITMM ;INIT MEM MGT REGISTERS
MOV PMA,R0 ;LO BITS OF MA
BIC #17777,R0 ;MASK FOR BITS 13-15
MOV PMA+2,R1 ;HI BITS OF MA
MOV R0,R3 ;SAVE THESE BITS
MOV R1,R4
ROL R0 ;GET MA BITS 13-21 INTO R1 BITS 7-15
ROL R1
ROL R0
ROL R1
ROL R0
SWAB R1
ROL R0
RORB R1
MOV R1,SAVPAR ;CONSTANT FOR LOADING PAR6 LATER
TST UBMPRS ;SEE IF UNIBUS MAP ENABLED
BEQ 9\$;BR IF NOT (NO UNIBUS MAPPING)
;PREPARE FOR 22-BIT ADDRESSING - LOAD UNIBUS MAP REGISTERS
MOV #MAPLOO,R0 ;STARTING ADDR OF MAP REGISTERS
MOV #31,R1 ;SET REGISTER COUNTER
4\$: MOV R3,(R0)+ ;LOAD A MAP REGISTER
MOV R4,(R0)+
ADD #20000,R3 ;ADD 4K WORDS
ADC R4
SOB R1,4\$;LOOP UNTIL 31(DEC) ARE LOADED
BIC #160000,P.BALO(R5) ;CLEAR BA BITS 13-17
BICB #3,P.BAHI(R5)
9\$: RESREG ;RESTORE R0-R5
RTS PC ;RETURN

*FNDXDP - FIND STARTING ADR OF XXDP LOADER, AND STORE IT
*IN XXDPAD.

FNDXDP: TST MAHILM+2 ;TEST HI BITS OF UPPER MEMORY LIMIT
BEQ 6\$;BR IF HI BITS ARE 0

G11

MD-11-DZRN-C - RK611/RK06 SUBSYS. VERIF. : PART 2 MACY11 27(1006) 05-OCT-76 10:59 PAGE 137
 DZRN.C.P11 05-OCT-76 10:07 GTSWRG - OPEN SOFTWARE SWITCH REGISTER FOR MODIFICATION

SEQ 0136

7059	030464	012737	160000	005536	4\$:	MOV	#160000,XXDPAD	:START OF 29K IS END OF XXDP
7060	030472	000412				BR	8\$	
7061	030474	023727	005572	157776	6\$:	CMP	MAHILM,#157776	:SEE IF MEM LIMIT > OR = 157776
7062	030502	103370				BHIS	4\$:BR IF YES
7063	030504	013737	005572	005536		MOV	MAHILM,XXDPAD	:HI MEM LIMIT IS END OF XXDP
7064	030512	062737	000002	005536		ADD	#2,XXDPAD	:ADD 2 BYTES
7065	030520	162737	006000	005536	8\$:	SUB	#6000,XXDPAD	:COMPUTE START OF XXDP
7066	030526	000207				RTS	PC	:RETURN
7067								
7068								
7069								
7070								
7071								
7072								
7073								
7074								
7075								
7076								
7077								
7078								
7079								
7080								
7081	030530	104407						
7082	030532	005004						
7083	030534	000410						
7084	030536	104407						
7085	030540	105037	003124					
7086	030544	105737	003125					
7087	030550	001425						
7088	030552	012704	000001					
7089	030556	005737	000170					
7090	030562	001020						
7091	030564	012702	003000					
7092	030570	013703	005536					
7093	030574	013700	005540					
7094	030600	013701	005542					
7095	030604	005737	056246					
7096	030610	100417						
7097	030612	005704						
7098	030614	001005						
7099	030616	012320						
7100	030620	005302						
7101	030622	001375						
7102	030624	104410						
7103	030626	000207						
7104	030630	062700	006000					
7105	030634	062703	006000					
7106	030640	014043						
7107	030642	005302						
7108	030644	001375						
7109	030646	000766						
7110								
7111	030650	004737	027646					
7112	030654	006100						
7113	030656	006101						
7114	030660	006100						


```

:*****
:* SAVXDP - SAVE THE XXDP LOADER IN HI MEMORY
:*THIS SUBROUTINE MOVES THE XXDP LOADER, WHICH RESIDES AT THE
:*PHYSICAL ADDRESS STORED IN XXDPAD, TO THE PHYSICAL ADDRESS
:*STORED IN XDPSAV AND XDPSAV+2. A TOTAL OF 1536(DEC) WORDS ARE MOVED.
:*
:* GETXDP - RESTORE THE XXDP LOADER TO ORIGINAL LOCATION
:*THIS SUBROUTINE MOVES THE RELOCATED XXDP LOADER FROM THE ADDRESS
:*STORED IN XDPSAV, XDPSAV+2, BACK TO LOCATION XXDPAD (ITS ORIGINAL
:*ADDRESS).
:*****
SAVXDP: SAVREG          ;SAVE R0-R5
        CLR            R4          ;SET INDICATOR TO SAVE XXDP
        BR            XDP1
GETXDP: SAVREG          ;SAVE R0-R5
        CLRB          XOVLD        ;CLEAR XXDP OVERLAID INDICATOR
        TSTB          XDPSVD        ;SEE IF XXDP WAS SAVED
        BEQ            XDP2        ;BR IF NOT SAVED
        MOV            #1,R4        ;SET INDICATOR TO GET XXDP
XDP1:   TST            KILLDR        ;SEE IF OK TO OVERLAY LOADER
        BNE            XDP2        ;BR IF YES, TO EXIT
        MOV            #1536.,R2    ;GET SET TO MOVE 1536(DEC) WORDS
        MOV            XXDPAD,R3    ;GET ORIG ADR OF START OF XXDP
        MOV            XDPSAV,R0    ;GET SAVE ADR LO BITS
        MOV            XDPSAV+2,R1  ;HI BITS
        TST            $KT11        ;SEE IF MEM MGT PRESENT
        BMI            XDP3        ;BR IF PRESENT
        TST            R4          ;SEE IF WANT TO SAVE OR GET XXDP
        BNE            XDP4        ;BR IF WANT TO GET XXDP
6$:    MOV            (R3)+,(R0)+    ;SAVE A WORD
        DEC            R2          ;SEE IF 1536(DEC) WORDS YET
        BNE            6$          ;BR IF NOT YET
XDP2:  RESREG          ;RESTORE R0-R5
        RTS            PC          ;RETURN
XDP4:  ADD            #6000,R0      ;POINT TO END OF SAVE AREA
        ADD            #6000,R3      ;POINT TO END OF XXDP LOADER AREA
8$:    MOV            -(R0),-(R3)    ;GET A WORD
        DEC            R2          ;SEE IF 1536(DEC) WORDS YET
        BNE            8$          ;BR IF NOT YET
        BR            XDP2        ;GO EXIT
;COME HERE IF MEM MGT
XDP3:  JSR            PC,INITMM     ;INIT MEM MGT REGISTERS
        ROL            R0          ;GET ADR BITS 13-21 INTO R1 BITS 7-15
        ROL            R1
        ROL            R0

```

H11

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 138
GTSWRG - OPEN SOFTWARE SWITCH REGISTER FOR MODIFICATION

SEQ 0137

```
7115 030662 006101 ROL R1
7116 030664 000301 SWAB R1
7117 030666 006100 ROL R0
7118 030670 106001 RORB R1
7119 030672 010137 172354 MOV R1, @#KIPAR6 ;SET UP PAR6
7120 030676 013701 005540 MOV XDF, SAV, R1 ;GET LO ADRS BITS
7121 030702 042701 160000 BIC #160000, R1 ;FIX UP R1 TO REFERENCE PAR6
7122 030706 052701 140000 BIS #140000, R1
7123 030712 005704 16$: TST R4 ;SEE IF WANT TO SAVE OR GET XXDP
7124 030714 001007 BNE 18$ ;BR IF WANT TO GET XXDP
7125 030716 012305 MOV (R3)+, R5 ;SAVE A WORD
7126 030720 005237 177572 INC @#SRO ;TURN ON MEMORY MANAGEMENT
7127 030724 010521 MOV R5, (R1)+
7128 030726 005337 177572 DEC @#SRO ;TURN OFF MEMORY MANAGEMENT
7129 030732 000406 BR 20$
7130 030734 005237 177572 18$: INC @#SRO ;TURN ON MEMORY MANAGEMENT
7131 030740 012105 MOV (R1)+, R5 ;GET A WORD
7132 030742 005337 177572 DEC @#SRO ;TURN OFF MEMORY MANAGEMENT
7133 030746 010523 MOV R5, (R3)+
7134 030750 032701 020000 20$: BIT #BIT13, R1 ;SEE IF OVERFLOW TO PAGE 7
7135 030754 001405 BEQ 22$ ;BR IF NOT
7136 030756 042701 020000 BIC #BIT13, R1 ;SET PAGE = 6 AGAIN
7137 030762 062737 000200 172354 ADD #200, @#KIPAR6 ;UPDATE PAR BY 4K
7138 030770 005302 22$: DEC R2 ;SEE IF 1536 WORDS YET
7139 030772 001347 BNE 16$ ;BR IF NOT YET
7140 030774 000713 BR XDP2 ;BR TO RETURN
7141
7142
7143
7144 ;*****
7145 .SBTTL INITSS - INITIALIZE SUBSYSTEM
7146 ;*
7147 ;*THIS SUBROUTINE INITIALIZES THE DRIVER AND ITS PARAMETERS
7148 ;*AND DOES A SUBSYSTEM CLEAR.
7149 ;* USES - R2, R5
7150 ;* CALL:
7151 ;* JSR PC, INITSS
7152 ;*****
7153
7154 030776 012737 042466 003036 INITSS: MOV #ERRHDL, A, ABNL ;SET UP ABNORMAL ERROR RETURN ADDRESS
7155 031004 012737 041220 003034 MOV #ERRFRE, A, NORM ;SET UP NORMAL RETURN ADDRESS
7156 031012 013702 003026 MOV RKBAS, R2 ;GET ADDRESS OF RK611 REGISTERS
7157 031016 012705 002620 MOV #PARM0, R5 ;GET ADDRESS OF PARAMETER BLOCK
7158 031022 105037 003133 CLRB NORTRY ;CLEAR "NO-RETRY" FLAG
7159 031026 004737 031070 JSR PC, CLRPRM ;CLEAR DRIVER INPUT PARAMETERS
7160 031032 112765 000177 000001 MOVB #SUBCLR, P, CMND(R5) ;SET SUBSYSTEM CLEAR COMMAND
7161 031040 004737 040774 JSR PC, DRVCAL ;DO SUBSYSTEM CLEAR
7162 031044 113765 005500 000000 MOVB DRIVE, P, DRVN(R5) ;SET CURRENT DRIVE NO.
7163 031052 113737 005500 002704 MOVB DRIVE, P, PARM1 ;SET DRIVE NO. IN ALTERNATE P.B.
7164 031060 113765 003115 000007 MOVB FORMAT, P, CS1H(R5) ;SET CURRENT DRIVE FORMAT
7165 031066 000207 RTS ;SUBROUTINE EXIT
7166
7167
7168
7169 ;*****
7170 .SBTTL CLRPRM - CLEAR DRIVER INPUT PARAMETERS
```

7171				
7172				
7173				
7174				
7175				
7176	031070	010046		
7177	031072	010546		
7178	031074	010500		
7179	031076	062705	000016	
7180	031102	005020		
7181	031104	020005		
7182	031106	001375		
7183	031110	012605		
7184	031112	012600		
7185	031114	000207		
7186				
7187				
7188				
7189				
7190				
7191				
7192				
7193				
7194				
7195				
7196				
7197				
7198				
7199				
7200				
7201				
7202				
7203				
7204				
7205				
7206	031116	004737	030776	
7207	031122	113765	005500	000000
7208				
7209	031130	001012		
7210	031132	122737	000013	000041
7211	031140	001006		
7212	031142	105737	003106	
7213	031146	001003		
7214	031150	104401	007673	
7215	031154	000434		
7216	031156	112765	000141	000001
7217	031164	012737	027064	003036
7218	031172	012737	000377	005530
7219	031200	004737	040774	
7220				
7221	031204	012737	042466	003036
7222	031212	022737	000377	005530
7223	031220	001420		
7224	031222	113737	005500	007374
7225	031230	152737	000060	007374
7226	031236	104401	007366	

```

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

```

```

CLRPRM: MOV R0,-(SP) ;SAVE R0
        MOV R5,-(SP) ;SAVE R5
        MOV R5,R0 ;GET PARAMETER BLOCK ADDRESS
1$: ADD #P.CS1,R5 ;COMPUTE LIMIT ADDRESS
    CLR (R0)+ ;CLEAR A WORD IN PARAMETER BLOCK
    CMP R0,R5 ;SEE IF DONE YET
    BNE 1$ ;BR IF NOT DONE YET
    MOV (SP)+,R5 ;RESTORE R5
    MOV (SP)+,R0 ;RESTORE R0
    RTS PC ;RETURN

```

```

;*****

```

```

.SBTTL SCNDRV - SCAN DRIVE FOR STATUS
;*THIS SUBROUTINE SCANS THE SELECTED DRIVE TO INSURE
;*THAT THE DRIVE IS ON-LINE, READY, NOT WRITE-LOCKED,
;*AND NOT LOADED WITH AN ALIGNMENT CARTRIDGE. IF ANY
;*OF THESE CONDITIONS ARE NOT MET, AN APPROPRIATE
;*MESSAGE IS TYPED, RK06 INTERRUPT IS DISABLED, AND
;*A RETURN IS MADE TO THE ADDRESS LISTED IMMEDIATELY
;*AFTER THE CALL TO SCNDRV. THE REQUESTED DRIVE NO.
;*MUST BE PASSED TO THE SUBROUTINE IN WORD "DRIVE".
;*ALSO, IF RUNNING FROM ADRS 200 AND RK06 IS LOAD MEDIUM,
;*DRIVE 0 WILL BE REJECTED FOR USE.
;* CALL - JSR PC,SCNDRV
;* <ERROR RETURN ADDRESS>
;*****

```

```

SCNDRV: JSR PC,INITSS ;INITIALIZE DRIVER AND CLEAR SUBSYSTEM
        MOV# DRIVE,P.DRVN(R5) ;GET DRIVE NUMBER
;SEE IF DRIVE 0 IS XXDP LOAD MEDIUM
        BNE 3$ ;BR IF NOT DRIVE 0
        CMP# #13,0#41 ;SEE IF RK06 IS XXDP MEDIUM
        BNE 3$ ;BR IF NOT
        TSTB MDFLAG ;SEE IF 200 START
        BNE 3$ ;BR IF NOT
        TYPE ,DROXDP ;TYPE "DRIVE 0 IS LOAD MEDIUM"
        BR 2$ ;TAKE ERROR EXIT
3$: MOV# #RDSTAT,P.CMND(R5) ;SET READ DRIVE STATUS COMMAND
    MOV #NEDHDL,A.ABNL ;SET NED ABNORMAL RETURN ADDRESS
    MOV #377,NEWON ;INIT. ON-LINE INDICATOR
    JSR PC,DRVCAL ;READ ALL DRIVE STATUS
;SEE IF THIS DRIVE EXISTS AND IS ON-LINE
    MOV #ERRHDL,A.ABNL ;RESTORE ABNL RETURN ADDRESS
    CMP #377,NEWON ;SEE IF NED INDICATION ON THIS DRIVE
    BEQ 4$ ;BR IF NED NOT SET
    MOV# DRIVE,BADDRV+6 ;GET DRIVE NO. INTO BUF
    BISB #'0,BADDRV+6 ;CONVERT TO ASCII
    TYPE ,BADDRV ;TYPE "DRIVE X"

```

```

7227 031242 104401 007377          TYPE      ,NXDRIV          ;TYPE "NON-EXISTENT"
7228          ;SERVICE ERRORS HERE
7229 031246 042762 000100 000000 2$: BIC      #IE,RKCS1(R2)  ;DISABLE RK06 INTERRUPT
7230 031254 017616 000000          MOV      2(SP),(SP)    ;SET UP ERROR RETURN ADDRESS
7231 031260 000207          RTS      PC            ;ERROR RETURN
7232          ;SEE IF DRIVE IS READY
7233 031262 032765 000200 000040 4$: BIT      #S.DRY,P.A00(R5) ;TEST FOR DRIVE READY
7234 031270 001013          BNE      6$           ;BR IF DRIVE IS READY
7235 031272 113737 005500 007374          MOV      DRIVE,BADDRV+6 ;GET DRIVE NO. INTO BUF
7236 031300 152737 000060 007374          BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
7237 031306 104401 007366          TYPE     ,BADDRV       ;TYPE "DRIVE X"
7238 031312 104401 007416          TYPE     ,NTREDY       ;TYPE "NOT READY"
7239 031316 000753          BR      2$           ;TAKE ERROR EXIT
7240          ;SEE IF DRIVE IS WRITE ENABLED
7241 031320 032765 004000 000040 6$: BIT      #S.WRL,P.A00(R5) ;SEE IF WRITE LOCK SET
7242 031326 001413          BEQ      8$           ;BR IF WRITE LOCK NOT SET
7243 031330 113737 005500 007374          MOV      DRIVE,BADDRV+6 ;GET DRIVE NO.
7244 031336 152737 000060 007374          BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
7245 031344 104401 007366          TYPE     ,BADDRV       ;TYPE "DRIVE X"
7246 031350 104401 007432          TYPE     ,WRTLOK       ;TYPE "WRITE-LOCKED"
7247 031354 000734          BR      2$           ;TAKE ERROR EXIT
7248          ;SEE IF DRIVE NOT LOADED WITH ALIGNMENT CARTRIDGE
7249 031356 112765 000103 000001 8$: MOV      #PACK,P.CMND(R5) ;SET PACK ACKNOWLEDGE COMMAND
7250 031364 004737 040774          JSR      PC,DRVCAL     ;SET VOLUME VALID
7251 031370 112765 000121 000001          MOV      #RDATA,P.CMND(R5) ;SET READ COMMAND
7252 031376 012765 000632 000002          MOV      #LSTCYL,P.CYLN(R5) ;SET CYLINDER = 632(8)
7253 031404 112765 000002 000005          MOV      #LSTTRK,P.TRCK(R5) ;SET TRACK = 2
7254 031412 012765 064640 000010          MOV      #RWBUF,P.BALO(R5) ;BUS ADDRESS
7255 031420 012765 177774 000012          MOV      #-4,P.WC(R5)   ;READ 4 WORDS
7256 031426 105065 000007          CLRB     P.C51H(R5)    ;SET 22-SECTOR FORMAT
7257 031432 004737 040774          JSR      PC,DRVCAL     ;READ 4 WORDS OF BAD SECTOR FILE
7258 031436 032737 100000 005474          BIT      #ANYDER,RECODE ;SEE IF DATA ERROR
7259 031444 001402          BEQ      10$          ;BR IF OK
7260 031446 104401 013314          TYPE     ,BAD632       ;TYPE READ ERROR MESSAGE
7261 031452 022737 177777 064646 10$: CMP      #177777,RWBUF+6 ;SEE IF ALL 1'S IN I.D. WORD 3
7262 031460 001013          BNE      12$          ;BR IF NOT ALL 1'S
7263 031462 113737 005500 007374          MOV      DRIVE,BADDRV+6 ;GET DRIVE NO.
7264 031470 152737 000060 007374          BISB     #'0,BADDRV+6   ;CONVERT TO ASCII
7265 031476 104401 007366          TYPE     ,BADDRV       ;TYPE "DRIVE X"
7266 031502 104401 007451          TYPE     ,ALNPAK       ;TYPE "LOADED WITH ALIGN PACK"
7267 031506 000657          BR      2$           ;TAKE ERROR EXIT
7268          ;ERROR FREE RETURN
7269 031510 112765 000113 000001 12$: MOV      #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
7270 031516 004737 040774          JSR      PC,DRVCAL     ;RECALIBRATE THIS DRIVE
7271 031522 062716 000002          ADD      #2,(SP)       ;FIX UP RETURN PC
7272 031526 000207          RTS      PC            ;RETURN

```

```

7273
7274
7275
7276          ;*****
7277          .SBTTL  CHKDRV - CHECK STATUS OF DRIVE
7278          ;*THIS SUBROUTINE CHECKS THE DESIRED DRIVE TO DETERMINE
7279          ;*IF THE DRIVE IS ON-LINE,READY,WRITE-PROTECTED. IF NOT, THE
7280          ;*APPROPRIATE ERROR MESSAGE IS TYPED,AND THE CPU HALTS
7281          ;*WHILE MANUAL INTERVENTION TAKES PLACE TO CORRECT THE
7282          ;*PROBLEM. THE OPERATOR THEN DEPRESSES 'CONT' TO PROCEED,

```

K11

```

7283 ;*AND A RETURN IS MADE TO THE ADDRESS LISTED IMMEDIATELY
7284 ;*AFTER THE CALL. IF THERE ARE NO PROBLEMS WITH THIS
7285 ;*DRIVE, A NORMAL RETURN IS MADE.
7286 ;*R5 MUST CONTAIN THE ADDRESS OF THE PARAMETER BLOCK,
7287 ;*AND R2 MUST CONTAIN THE RK06 REGISTER BASE ADDRESS.
7288 ;*
7289 ;* CALL:
7290 ;* JSR PC,CHKDRV
7291 ;* ERROR ADDRESS
7292 ;* *****
7293
7294 031530 113765 005500 000000 CHKDRV: MOVB DRIVE,P.DRVN(R5) ;SET DESIRED DRIVE NUMBER
7295 031536 112765 000141 000001 MOVB #RDSTAT,P.CMND(R5) ;SET READ DRIVE STATUS COMMAND
7296 031544 012737 027064 003036 MOV #NEDHDL,A.ABNL ;SET NED ABNORMAL RETURN ADDRESS
7297 031552 012737 000377 005530 MOV #377,NEWON ;INITIALIZE ON-LINE INDICATOR
7298 031560 004737 040774 JSR PC,DRVCAL ;READ ALL DRIVE STATUS
7299 ;SEE IF THIS DRIVE EXISTS AND IS ON-LINE
7300 031564 012737 042466 003036 MOV #ERRHDL,A.ABNL ;RESTORE ABNORMAL DRIVER RETURN ADDRESS
7301 031572 022737 000377 005530 CMP #377,NEWON ;SEE IF NED INDICATION ON THIS DRIVE
7302 031600 001411 BEQ 1$ ;BR IF NED NOT SET
7303 031602 113701 005500 MOVB DRIVE,R1 ;GET DRIVE NUMBER
7304 031606 152701 000060 BISB #'0,R1 ;CONVERT TO ASCII
7305 031612 110137 012043 MOVB R1,DRVND ;PUT DRIVE NO. INTO MSG BUFFER
7306 031616 104401 012031 TYPE ,NONEXD ;TYPE NON-EXISTENT DRIVE MESSAGE
7307 031622 000414 BR 3$ ;SERVICE THE ERROR
7308 ;SEE IF DESIRED DRIVE IS READY
7309 031624 032765 000200 000040 1$: BIT #S.DRY,P.AOD(R5) ;TEST FOR DRIVE READY
7310 031632 001024 BNE 4$ ;BR IF DRIVE READY
7311 031634 113701 005500 MOVB DRIVE,R1 ;GET DRIVE NUMBER
7312 031640 152701 000060 BISB #'0,R1 ;CONVERT TO ASCII
7313 031644 110137 011677 MOVB R1,DRNRDY ;PUT DRIVE NUMBER IN MESSAGE BUFFER
7314 031650 104401 011665 TYPE ,NOTRDY ;TYPE DRIVE NOT READY MESSAGE
7315 ;SERVICE ALL ERRORS HERE
7316 031654 042762 000100 000000 3$: BIC #IE,RKCS1(R2) ;CLEAR RK06 INTERRUPT ENABLE BIT
7317 031662 000000 HALT ;HALT IS NECESSARY DURING MANUAL
7318 ; INTERVENTIONS, TO CHANGE PACK,
7319 ; WRITE LOCK, START DRIVE, ETC.
7320 ; PRESS 'CONT' TO PROCEED !
7321 031664 112765 000177 000001 MOVB #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR COMMAND
7322 031672 004737 040774 JSR PC,DRVCAL ;CLEAR THE SUBSYSTEM
7323 031676 017616 000000 MOV 2(SP),(SP) ;SET UP ERROR RETURN ADDRESS
7324 031702 000207 RTS PC ;ERROR RETURN
7325 ;SEE IF DRIVE IS WRITE-LOCKED
7326 031704 032765 004000 000040 4$: BIT #S.WRL,P.AOD(R5) ;SEE IF WRITE LOCK SET
7327 031712 001011 BNE 5$ ;BR IF WRITE-PROTECTED
7328 031714 113701 005500 MOVB DRIVE,R1 ;GET DRIVE NUMBER
7329 031720 152701 000060 BISB #'0,R1 ;CONVERT TO ASCII
7330 031724 110137 012172 MOVB R1,NODRLK ;PUT DRIVE NUMBER IN MSG BUFFER
7331 031730 104401 012160 TYPE ,NOTLOK ;TYPE "DRIVE NOT WRITE-LOCKED"
7332 031734 000747 BR 3$ ;SERVICE THE ERROR
7333 ;RECALIBRATE DESIRED DRIVE, SET VOLUME VALID
7334 031736 112765 000113 000001 5$: MOVB #RECAL,P.CMND(R5) ;SET RECALIBRATE COMMAND
7335 031744 004737 040774 JSR PC,DRVCAL ;RECALIBRATE DRIVE
7336 031750 112765 000103 000001 MOVB #PACK,P.CMND(R5) ;SET PACK ACKNOWLEDGE COMMAND
7337 031756 004737 040774 JSR PC,DRVCAL ;DO PACK ACK. (SETS VOLUME VALID)
7338 ;RETURN HERE
    
```

```

7339 031762 062716 000002
7340 031766 000207
7341
7342
7343
7344
7345
7346
7347 031770 112765 000117 000001
7348 031776 005065 000002
7349 032002 004737 040774
7350 032006 005265 000002
7351 032012 022765 000366 000002
7352 032020 001370
7353 032022 000207
7354
7355
7356
7357
7358
7359
7360
7361 032024 112765 000107 000001
7362 032032 004737 040774
7363 032036 112765 000141 000001
7364 032044 004737 040774
7365 032050 032765 000040 000044
7366 032056 001772
7367 032060 104401 012755
7368 032064 004737 027302
7369 032070 005737 005522
7370 032074 001775
7371 032076 022737 000122 005522
7372 032104 001403
7373 032106 004737 027322
7374 032112 000762
7375 032114 112765 000111 000001
7376 032122 004737 040774
7377 032126 000207
7378
7379
7380
7381
7382
7383
7384
7385
7386 032130 011637 001076
7387 032134 012706 001076
7388 032140 005037 005474
7389 032144 105037 003116
7390 032150 012705 002620
7391 032154 112765 000177 000001
7392 032162 004737 040774
7393 032166 012737 000000 177776
7394 032174 000207

```

```

7$: ADD #2,(SP) ;FIX UP RETURN ADDRESS ON STACK
RTS PC ;ERROR-FREE RETURN

```

```

*****
* ALNSEK - SEEK IN SINGLE INCREMENTS FROM CYL 0 TO ALIGNMENT CYL (365 OCT).
*****

```

```

ALNSEK: MOVB #SEEK,P.CMND(R5) ;SET SEEK COMMAND
CLR P.CYLN(R5) ;INIT CYL TO 0
2$: JSR PC,DRVCAL ;SEEK TO THIS CYL
INC P.CYLN(R5) ;INCR CYL NO.
CMP #ALNCYL+1,P.CYLN(R5) ;SEE IF 365 REACHED YET
BNE 2$ ;BR IF NOT YET
RTS PC ;RETURN

```

```

*****
* WAIT4R - THIS SUBROUTINE UNLOADS HEADS ON THE CURRENT DRIVE, TYPES
* "TYPE <R> WHEN READY", AND THEN LOADS THE HEADS WHEN <R> IS TYPED.
*****

```

```

WAIT4R: MOVB #UNLOAD,P.CMND(R5) ;SET UNLOAD COMMAND
JSR PC,DRVCAL ;UNLOAD HEADS ON THIS DRIVE
MOVB #RDSTAT,P.CMND(R5) ;SET READ STATUS COMMAND
1$: JSR PC,DRVCAL ;READ STATUS OF DRIVE
BIT #S.HDHM,P.ADI(R5) ;SEE IF HEADS UNLOADED YET
BEQ 1$ ;BR IF NOT YET
2$: TYPE ,RWNRDY ;TYPE "TYPE <R> WHEN READY"
JSR PC,PREPKB ;PREPARE FOR KBD INPUT
4$: TST INTCHR ;SEE IF ANY INPUT YET
BEQ 4$ ;BR IF NOT YET
CMP #'R,INTCHR ;SEE IF <R> TYPED
BEQ 6$ ;BR IF <R> TYPED
JSR PC,ECOBAD ;ECHO BAD INPUT
BR 2$ ;GO ASK AGAIN
6$: MOVB #SRTSPL,P.CMND(R5) ;SET START SPINDLE COMMAND
JSR PC,DRVCAL ;START SPINDLE AND LOAD HEADS
RTS PC ;RETURN

```

```

*****
*SETUP - SET UP FOR LOOP ON ERROR
*THIS SUBROUTINE CANNOT BE CALLED BY ANY OTHER
*SUBROUTINE --- ONLY MAIN-LINE CODE !!!!
*****

```

```

SETUP: MOV (SP),2#STACK-2 ;MOVE RETURN PC ON STACK
MOV #STACK-2,SP ;RE-INIT THE STACK POINTER
CLR RECODE ;CLEAR ERROR RECOVERY FLAGS
CLRB ERRCNT ;CLEAR RETRY ERROR COUNT
MOV #PARMO,R5 ;SET PARAM BLK ADRS
MOVB #SUBCLR,P.CMND(R5) ;SET SUBSYSTEM CLEAR CMND
JSR PC,DRVCAL ;CLEAR THE SUBSYSTEM
MOV #PRO,2#PS ;RE-ESTABLISH PRIORITY 0
RTS PC ;RETURN

```

7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450

```

*****
SBTTL CHKITR - CHECK CURRENT TEST ITERATION NUMBER
*THIS SUBROUTINE LOADS THE ITERATION NUMBER FOR THE CURRENT
*TEST INTO $TIMES, AND CHECKS IT TO DETERMINE IF THIS TEST
*SHOULD BE RUN. IF ITERATION NO. = 0, THE SUBROUTINE RETURNS
*TO THE STORED PC (AND THE TEST WILL BE SKIPPED). IF IT IS
*NOT 0, THE RETURN IS MADE TO PC+2, AND THE TEST WILL
*BE RUN. HOWEVER, ON THE FIRST PASS (QUICK VERIFY) OF A RUN
*WHICH STARTED AT ADRS 200, ALL TESTS WILL BE RUN ONCE.
*****
CHKITR: MOV      R1, -(SP)          ;SAVE R1
        MOVB     #100, $ERMAX      ;SET MAX ERROR CNT TO 100 FOR $SCOPE
        TSTB    DULACS            ;SEE IF DUAL-ACCESS FLAG SET
        BNE     3$,              ;BR IF YES
        TSTB    MDFLAG           ;SEE IF 200 START
        BNE     4$,              ;BR IF NOT
        TST     $PASS            ;SEE IF FIRST PASS
        BNE     4$,              ;BR IF NOT
3$:     MOV      #1, $TIMES        ;SET UP FOR 1 ITERATION
        BR      1$,              ;GO RUN DUAL-ACCESS TEST
4$:     MOVB     $STNM, R1        ;GET CURRENT TEST NUMBER
        DEC     R1                ;DECREMENT BY 1
        ASL     R1                ;DOUBLE IT, TO GET TEST LIST INDEX
        MOV     TSTLST(R1), $TIMES ;LOAD ITERATION NUMBER
        BEQ     2$,              ;BR IF TEST SHOULD BE SKIPPED
1$:     ADD     #4, 2(SP)         ;ADJUST RETURN PC TO RUN THIS TEST
2$:     MOV     (SP)+, R1         ;RESTORE R1
        RTS     PC                ;RETURN

```

```

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

```


N11

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 144
 RDCHRS - READ A STRING OF KBD INPUT CHARS

SEQ 0143

7451										
7452	032272									
7453	032272	010146								
7454	032274	010246								
7455	032276	005000								
7456	032300	005001								
7457										
7458	032302	104406								
7459	032304	112602								
7460										
7461	032306	122702	000003							
7462	032312	001006								
7463	032314	104401	013360							
7464	032320	017666	000004	000004						
7465	032326	000523								
7466										
7467	032330	122702	000032							
7468	032334	001006								
7469	032336	104401	013365							
7470	032342	062766	000002	000004						
7471	032350	000763								
7472										
7473	032352	122702	000025							
7474	032356	001006								
7475	032360	104401	013377							
7476	032364	062766	000004	000004						
7477	032372	000752								
7478										
7479	032374	122702	000007							
7480	032400	001005								
7481	032402	104401	013404							
7482	032406	004737	027614							
7483	032412	000764								
7484										
7485	032414	122702	000177							
7486	032420	001020								
7487	032422	005700								
7488	032424	001726								
7489	032426	005701								
7490	032430	001003								
7491	032432	005201								
7492	032434	104401	013415							
7493	032440	005037	005532							
7494	032444	005300								
7495	032446	116037	005262	005532						
7496	032454	104401	005532							
7497	032460	000710								
7498	032462	005701								
7499	032464	001403								
7500	032466	104401	013415							
7501	032472	005001								
7502										
7503	032474	122702	000015							
7504	032500	001426								
7505										
7506	032502	005037	005532							

```

RDCHRS:
MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
CLR R0 ;INITIALIZE CHARACTER COUNT
CLR R1 ;INITIALIZE RUB-OUT INDICATOR
;READ A CHARACTER
2$: RDCHR ;READ A CHARACTER
MOV (SP)+,R2 ;GET CHARACTER INTO R2
;CHECK FOR (↑C)
CMPB #003,R2 ;SEE IF (↑C) TYPED
BNE 4$ ;BR IF NOT (↑C)
TYPE ,CNTRLC ;ECHO (↑C)
3$: MOV 24(SP),4(SP) ;PUT RETURN ADDRESS ON STACK
BR 24$ ;BR TO TAKE EXIT
;CHECK FOR (↑Z)
4$: CMPB #032,R2 ;SEE IF (↑Z) TYPED
BNE 6$ ;BR IF NOT (↑Z)
TYPE ,CNTRLZ ;ECHO (↑Z)
ADD #2,4(SP) ;MAKE OLD PC POINT TO NEXT RETURN ADR.
BR 3$ ;BR TO TAKE (↑Z) EXIT
;CHECK FOR (↑U)
6$: CMPB #025,R2 ;SEE IF (↑U) TYPED
BNE 8$ ;BR IF NOT (↑U)
TYPE ,CNTRLU ;ECHO (↑U)
7$: ADD #4,4(SP) ;MAKE OLD PC POINT TO NEXT RETURN ADDR.
BR 3$ ;BR TO TAKE (↑U) EXIT
;CHECK FOR (↑G)
8$: CMPB #007,R2 ;SEE IF (↑G) TYPED
BNE 9$ ;BR IF NOT (↑G)
TYPE ,CNTRLG ;ECHO (↑G)
JSR PC,GTSWRG ;OPEN SOFTWARE SWITCH REG. FOR CHANGE
BR 7$ ;TAKE (↑U) RETURN
;CHECK FOR RUB-OUT (DELETE)
9$: CMPB #177,R2 ;SEE IF RUB-OUT (DEL) TYPED
BNE 14$ ;BR IF NOT RUB-OUT
TST R0 ;CHECK THE CHARACTER COUNT
BEQ 2$ ;BR IF COUNT = 0
TST R1 ;CHECK THE RUB-OUT INDICATOR
BNE 11$ ;BR IF WE HAD A PREVIOUS RUB-OUT
INC R1 ;SET RUB-OUT INDICATOR
TYPE ,BKSLSH ;TYPE A BACK-SLASH (\)
11$: CLR SCRACH ;USE SCRATCH WORD FOR TEMP. BUFFER
DEC R0 ;DECREMENT COUNT
MOV (R0),SCRACH ;GET LAST CHAR. INTO BUFFER
TYPE ,SCRACH ;ECHO CHARACTER TO BE DELETED
BR 2$ ;GO READ ANOTHER CHARACTER
14$: TST R1 ;CHECK THE RUB-OUT INDICATOR
BEQ 16$ ;BR IF INDICATOR IS NOT SET
TYPE ,BKSLSH ;TYPE A BACK-SLASH
CLR R1 ;CLEAR THE RUB-OUT INDICATOR
;CHECK FOR CARRIAGE RETURN
16$: CMPB #015,R2 ;SEE IF <CR> TYPED
BEQ 19$ ;BR IF <CR>
;HANDLE POSSIBLE DIGIT
CLR SCRACH ;USE SCRATCH WORD FOR TEMP. BUFFER

```

```

7507 032506 110237 005532      MOVB   R2,SCRACH      :GET THIS CHARACTER INTO BUFFER
7508 032512 104401 005532      TYPE   SCRACH        :ECHO THE CHARACTER TYPED
7509 032516 110260 005262      MOVB   R2,BUFFD(R0)  :PUT CHARACTER INTO BUFFER
7510 032522 005200                INC    RO             :INCREMENT CHARACTER COUNTER
7511 032524 022700 000120      CMP    #80.,RO       :SEE IF TOO MANY CHARACTERS TYPED
7512 032530 001264                BNE    Z$            :BR IF NOT TOO MANY
7513 032532 104401 001315      TYPE   ,SCRLF        :TYPE <CR> AND <LF>
7514 032536 112760 000000 005262      MOVB   #0,BUFFD(R0)  ;PUT TERMINATING NULL INTO BUFFER
7515 032544 104401 005262      TYPE   ,BUFFD        :ECHO INPUT STRING
7516 032550 104401 001314      TYPE   ,SQUES        :TYPE (?), <CR>, <LF>
7517 032554 000703                BR     Z$            :TAKE ERROR EXIT FROM RDCHRS
7518 032556 104401 001315 19$:      TYPE   ,SCRLF        :TYPE <CR>, <LF>
7519 032562 112760 000000 005262      MOVB   #0,BUFFD(R0)  ;PUT TERMINATING NULL INTO BUFFER
7520 032570 062766 000006 000004      ADD    #6,4(SP)      :FIX UP RETURN ADDRESS
7521 032576 012602 24$:      MOV    (SP)+,R2      :RESTORE R2
7522 032580 012601                MOV    (SP)+,R1      :RESTORE R1
7523 032602 000207                RTS    PC            :SUBROUTINE EXIT

```

```

:*****
:SBTTL TYPTST - TYPE CURRENT TEST AND ITERATION NUMBER
:*THIS SUBROUTINE TYPES : "XX YYYYYY" WITH NO <CR>
:*OR <LF>. XX IS THE NUMBER OF THE CURRENT TEST AND
:*YYYYYY IS THE NUMBER OF ITERATIONS FOR THAT TEST.
:*R1 MUST CONTAIN THE INDEX INTO THE TEST LIST FOR
:*THE CURRENT TEST.
:* CALL - JSR PC,TYPTST
:*****

```

```

7527 032604 104401 013426      TYPTST: TYPE   SPACE1      :TYPE A SPACE
7528 032610 010146      MOV    R1,-(SP)      :PUT INDEX ONTO STACK
7529 032612 006216      ASR    (SP)          :DIVIDE BY 2
7530 032614 005216      INC    (SP)          :INCREMENT TO GET TEST NO.
7531 032616 104403      TYPOS                :TYPE TEST NO. IN OCTAL
7532 032620 002                .BYTE   2            :TYPE 2 DIGITS
7533 032621 000                .BYTE   0            :SUPPRESS LEADING ZEROS
7534 032622 104401 013421      TYPE   SPACE6        :TYPE 6 SPACES
7535 032626 016146 005616      MOV    ↑STLST(R1),-(SP) :PUT CURRENT ITERATION NO. ONTO STACK
7536 032632 104403      TYPOS                :TYPE ITERATION NO. IN OCTAL
7537 032634 006                .BYTE   6            :TYPE 6 DIGITS
7538 032636 000                .BYTE   0            :SUPPRESS LEADING ZEROS
7539 032636 000207      RTS    PC            :RETURN

```

```

:*****
:SBTTL TYPPRM - TYPE CURRENT PARAMETER
:*THIS SUBROUTINE TYPES THE LINE : XX=YYYYYYYY
:*WHERE XX IS THE CURRENT PARAMETER MNEMONIC, AND
:*YYYYYYYY IS ITS VALUE IN OCTAL DIGITS, WITH LEADING
:*ZEROS SUPPRESSED. NOTE: <CR> AND <LF> ARE NOT DONE.
:*ON ENTRY, R1 MUST CONTAIN THE INDEX INTO THE
:*PARAMETER TABLES, FOR THE CURRENT PARAMETER.
:* CALL - JSR PC,TYPPRM
:*****

```

```

7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552

```

```

7563 032640 016137 005776 013430 TYPFRM: MOV PRMNEM(R1),PRMBUF ;PUT MNEMONIC INTO OUTPUT BUFFER
7564 032646 104401 013430 TYPE PRMBUF ;TYPE "XX="
7565 032652 016137 005646 005466 MOV PRMLST(R1),LOWOCT ;PUT LOW BITS OF PARAM. INTO LOWOCT
7566 032660 005037 005470 CLR HIGOCT ;CLEAR HIGH BINARY BITS
7567 032664 022761 040515 005776 CMP #MA,PRMNEM(R1) ;SEE IF PARAMETER IS (MA)
7568 032672 001003 BNE IS ;BR IF NOT (MA)
7569 032674 016137 005650 005470 MOV PRMLST+2(R1),HIGOCT ;PUT HI BITS OF PARAM. INTO HIGOCT
7570 032702 012746 005466 1S: MOV #LOWOCT, -(SP) ;PUT ADDRESS OF BINARY VALUE ON STACK
7571 032706 004737 053752 JSR PC, @SDB20 ;CONVERT BINARY TO OCTAL ASCII
7572 032712 004737 054266 JSR PC, @SSUPRS ;TYPE YYYYYYYY, SUPPRESSING LEADING 0'S
7573 032716 000207 RTS PC ;RETURN

```

7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587

```

:*****
:SBTTL TYPFRM - TYPE CURRENT WORD OF DATA PATTERN 15
:*THIS SUBROUTINE TYPES THE LINE : WORD XX = YYYYYY,
:*WHERE XX IS THE NUMBER OF THE CURRENT WORD OF
:*USER-DEFINED PATTERN 15, AND YYYYYY IS ITS 16-BIT
:*VALUE, IN OCTAL DIGITS. <CR> AND <LF> ARE NOT TYPED.
:*ON ENTRY, R1 MUST CONTAIN THE INDEX OF THE CURRENT
:*WORD IN THE PATTERN 15 TABLE.
:* CALL - JSR PC,TYPFRM
:*****

```

```

7588 032720 104401 013434 TYPFRM: TYPE WORDSP ;TYPE "WORD "
7589 032720 010146 MOV R1, -(SP) ;PUT INDEX ONTO STACK
7590 032724 006216 ASR (SP) ;DIVIDE BY TWO FOR WORD NO.
7591 032726 104403 TYPOS ;GO TYPE WORD NUMBER
7592 032730 002 .BYTE 2 ;DIGIT COUNT = 2 FOR TYPOC
7593 032732 001 .BYTE 1 ;TELL TYPOS TO TYPE LEADING ZEROS
7594 032734 104401 013442 TYPE EQUALS ;TYPE " = "
7595 032740 016146 006764 MOV PAT15(R1), -(SP) ;GET BINARY VALUE OF THIS WORD
7596 032744 104403 TYPOS ;TYPE VALUE IN OCTAL
7597 032746 006 .BYTE 6 ;TYPE 6 DIGITS
7598 032747 001 .BYTE 1 ;TYPE LEADING ZEROS
7599 032750 000207 RTS PC ;RETURN

```

7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618

```

:*****
:SBTTL MODP15 - MODIFY USER-DEFINED PATTERN 15
:*THIS SUBROUTINE DETERMINES WHETHER OR NOT USER-DEFINED
:*DATA PATTERN 15 SHOULD BE OPENED FOR MODIFICATION, AND
:*IF SO, IT REQUESTS THE NEW VALUES AND CONTROLS THEIR
:*INPUT ON THE TTY. IT CHECKS VALUES FOR VALIDITY, AND
:*LOADS THE 16 WORDS INTO THE PATTERN 15 TABLE (PAT15).
:*IF THE OPERATOR ANSWERS THE REQUEST FOR A PATTERN WORD
:*WITH EXCLAMATION POINT (!) (CR), THE SUBROUTINE
:*PROPAGATES THE LAST WORD TYPED INTO ALL REMAINING
:*WORDS OF THE PATTERN 15 TABLE.
:* CALL - JSR PC,MODP15
:*****

```

032752

MODP15:

7619	032752	010046				MOV	RO,-(SP)	:SAVE R0
7620	032754	010146				MOV	R1,-(SP)	:SAVE R1
7621	032756	010246				MOV	R2,-(SP)	:SAVE R2
7622								
7623	032760	022761	052120	005776		:SEE IF	PARAMETER 15 PT	
7624	032766	001127				CMP	#PT,PRMNM(R1)	:SEE IF CURRENT PARAMETER IS (PT)
7625						BNE	22\$:BR IF NOT (PT)
7626	032770	032737	100000	005666		:SEE IF	PATTERN 15 IS SPECIFIED	
7627	032776	001523				BIT	#BIT15,PT	:SEE IF PATTERN 15 SPECIFIED
7628						BEQ	22\$:BR IF NOT SPECIFIED
7629	033000	104401	010754			:SEE IF	PATTERN 15 SHOULD BE MODIFIED	
7630	033004	004737	032272			4\$:	TYPE	MDFY15
7631	033010	033256				JSR	PC,RDCHRS	:ASK WHETHER PATTERN 15 SHOULD BE MODIFIED
7632	033012	033266				24\$:READ RESPONSE
7633	033014	033052				26\$:(:C) RETURN ADDRESS
7634	033016	005700				8\$:(:Z) RETURN ADDRESS
7635	033020	001512				TST	RO	:(:U) OR ERROR RETURN ADDRESS
7636	033022	022737	000115	005262		BEQ	22\$:SEE IF NULL INPUT
7637	033030	001405				CMP	#M,BUFF0	:BR IF MODIFICATION NOT REQUESTED
7638	033032	104401	005262			BEQ	6\$:SEE IF (M) TYPED
7639	033036	104401	001314			TYPE	,BUFF0	:BR IF MODIFICATION REQUESTED
7640	033042	000756				TYPE	,SQUES	:ECHO BAD INPUT
7641						BR	4\$:GO ASK AGAIN
7642	033044	104401	010724			:MODIFY	PATTERN 15	
7643	033050	005001				6\$:	TYPE	,SELP15
7644	033052	004737	032720			CLR	R1	:TYPE "MODIFY PATTERN 15"
7645	033056	104401	013426			8\$:	JSR	PC,TYPAT
7646	033062	104401	013446			TYPE	,SPACE1	:INITIALIZE WORD INDEX
7647						TYPE	,PROMPT	:TYPE CURRENT WORD AND VALUE
7648	033066	004737	032272			:READ AND CHECK	INPUT, IF ANY	:TYPE A SPACE
7649	033072	033256				JSR	PC,RDCHRS	:TYPE ASTERISK AND SPACE
7650	033074	033266				24\$:READ NEW DATA PATTERN WORD
7651	033076	033052				26\$:(:C) RETURN ADDRESS FOR RDCHRS
7652	033100	005700				8\$:(:Z) RETURN ADDRESS FOR RDCHRS
7653	033102	001006				TST	RO	:(:U) OR ERROR RETURN ADDR. FOR RDCHRS
7654	033104	062701	000002			BNE	12\$:SEE IF ANY INPUT
7655	033110	022701	000040			10\$:	ADD	#2,R1
7656	033114	001454				CMP	#32.,R1	:BR IF ANY INPUT
7657	033116	000755				BEQ	22\$:INCREMENT WORD INDEX
7658	033120	022737	000041	005262		BR	8\$:SEE IF ALL DONE
7659	033126	001431				CMP	#!,BUFF0	:BR IF DONE, TO RETURN
7660	033130	005002				BEQ	16\$:CONTINUE WITH NEXT WORD
7661	033132	122760	000041	005261		CLR	R2	:SEE IF (!) TYPED
7662	033140	001004				CMPB	#!,BUFF0-1(RO)	:BR TO PROPAGATE CURRENT VALUE
7663	033142	105060	005261			BNE	14\$:INIT. (!) INDICATOR
7664	033146	005300				CLRB	BUFF0-1(RO)	:SEE IF LAST CHAR IN BUF IS (!)
7665	033150	005202				DEC	RO	:BR IF NOT (!)
7666	033152	022700	000006			INC	R2	:INSERT TERMINATOR BYTE
7667	033156	002426				14\$:	CMP	#6,RO
7668	033160	012746	005262			BLT	20\$:DECREMENT CHAR COUNT
7669	033164	004737	053014			MOV	#BUFF0,-(SP)	:SET (!) INDICATOR
7670	033170	033234				JSR	PC,OCTBIN	:SEE HOW MANY CHARS NOW
7671	033172	012600				20\$:BR IF TOO MANY
7672	033174	005737	053146			MOV	(SP)+,RO	:GET BUF. ADDR. ON STACK FOR OCTBIN
7673	033200	001015				TST	\$HIOCT	:CHECK DIGITS AND CONVERT TO BINARY
7674	033202	010061	006764			BNE	20\$:ERROR RETURN ADDRESS FOR OCTBIN
						MOV	RO,PAT15(R1)	:GET BINARY VALUE
								:SEE IF VALUE EXCEEDS 16 BITS
								:BR TO ECHO BAD INPUT
								:PUT NEW WORD VALUE INTO TABLE

E12

```

7675 033206 005702          TST      R2          ;SEE IF (!) WAS TYPED
7676 033210 001735          BEQ      10$         ;BR IF (!) WAS NOT TYPED
7677          ;PROPAGATE CURRENT WORD TO END OF PATTERN 15 TABLE
7678 033212 022701 000036 16$:  CMP      #30.,R1      ;SEE IF ALL DONE YET
7679 033216 001413          BEQ      22$         ;BR IF DONE
7680 033220 016161 006764 006766  MOV     PAT15(R1),PAT15+2(R1) ;PROPAGATE TO NEXT WORD
7681 033226 062701 000002      ADD     #2,R1        ;INCREMENT WORD INDEX
7682 033232 000767          BR       16$         ;LOOP UNTIL DONE
7683          ;ECHO BAD INPUT
7684 033234 104401 005262 20$:  TYPE     .BUFFO      ;ECHO BAD INPUT
7685 033240 104401 001314      TYPE     .SQUES      ;TYPE (?) AND <CR>, <LF>
7686 033244 000702          BR       8$          ;BR TO ASK AGAIN
7687          ;NORMAL RETURN
7688 033246 012602 22$:  MOV     (SP)+,R2      ;RESTORE R2
7689 033250 012601          MOV     (SP)+,R1      ;RESTORE R1
7690 033252 012600          MOV     (SP)+,R0      ;RESTORE R0
7691 033254 000207          RTS      PC          ;RETURN
7692          ;(+C) RETURN
7693 033256 012766 014532 000006 24$:  MOV     #DRVTST,6(SP) ;PC FOR (+C) RETURN
7694 033264 000770          BR       22$        ;BR TO RETURN
7695          ;(+Z) RETURN
7696 033266 012766 016156 000006 26$:  MOV     #ASKMDE,6(SP) ;PC FOR (+Z) RETURN
7697 033274 000764          BR       22$        ;BR TO RETURN
7698
7699
7700
7701          ;*****
7702          ;SBTTL CHKPRM - CHECK VALUE OF INPUT PARAMETER
7703          ;*THIS SUBROUTINE CHECKS THE CONTENTS OF LOWOCT AND
7704          ;*HIGOCT (IF 32-BIT VALUE) AGAINST LOWER AND UPPER
7705          ;*LIMITS FOR THE CURRENT PARAMETER. THE PARAMETER TABLE
7706          ;*INDEX FOR THE CURRENT PARAMETER MUST BE PASSED
7707          ;*TO CHKPRM IN R1 ON ENTRY. IF THE PARAMETER VALUE
7708          ;*IS LEGAL, IT IS ENTERED INTO THE PARAMETER TABLE
7709          ;*(PRMLST). IF NOT, A RETURN IS MADE TO THE ADDR. FOLLOWING
7710          ;*THE CALL TO CHKPRM. IF THE PARAMETER HAS A 2-WORD
7711          ;*VALUE (32 BITS) R1 IS INCREMENTED BY 2 BEFORE RETURN.
7712          ;* CALL -      JSR      PC,CHKPRM
7713          ;*              <ERROR RETURN ADDRESS>
7714          ;*              RETURN
7715          ;*****
7716
7717 033276 104407          CHKPRM: SAVREG          ;SAVE R0-R5
7718 033300 010102          MOV     R1,R2        ;GET COPY OF INDEX
7719 033302 006302          ASL     R2            ;DOUBLE IT
7720 033304 022761 040515 005776  CMP     #*MA,PRMNEM(R1) ;SEE IF PARAMETER IS (MA)
7721 033312 001422          BEQ     20$          ;BR IF (MA)
7722 033314 005737 005470      TST     HIGOCT        ;SEE IF HIGH BITS = 0
7723 033320 001403          BEQ     18$          ;BR IF ZERO
7724 033322 017616 000000 16$:  MOV     2(SP),(SP)    ;GET ERROR RETURN PC
7725 033326 000475          BR       30$        ;GO TO RETURN
7726          ;CHECK VALIDITY OF 16-BIT PARAMETER VALUE
7727 033330 023762 005466 005722 18$:  CMP     LOWOCT,PRMLIM(R2) ;SEE IF INPUT VALUE IS TOO SMALL
7728 033336 103771          BLO     16$          ;BR IF INPUT VALUE TOO SMALL
7729 033340 023762 005466 005724  CMP     LOWOCT,PRMLIM+2(R2) ;SEE IF INPUT VALUE IS TOO LARGE
7730 033346 101365          BHI     16$          ;BR IF INPUT VALUE IS TOO LARGE
  
```

```

7731 ;UPDATE 16-BIT PARAMETER VALUE IN LIST
7732 033350 013761 005466 005646 MOV LOWOCT,PRMLST(R1) ;PUT NEW PARAMETER VALUE INTO LIST
7733 033356 000457 BR 28$ ;BR TO RETURN
7734 ;CHECK VALIDITY OF 32-BIT PARAMETER VALUE
7735 033360 032737 000001 005466 20$: BIT #BIT0,LOWOCT ;SEE IF MA IS ODD
7736 033366 001355 BNE 16$ ;BR IF MA IS ODD
7737 033370 023762 005470 005724 CMP HIGOCT,PRMLIM+2(R2) ;SEE IF HIGH WORD TOO SMALL
7738 033376 103751 BLO 16$ ;BR IF HIGH WORD IS TOO SMALL
7739 033400 001004 BNE 24$ ;BR IF HIGH WORD NOT EQUAL TO LOW LIMIT
7740 033402 023762 005466 005722 CMP LOWOCT,PRMLIM(R2) ;SEE IF LOW WORD IS TOO SMALL
7741 033410 103744 BLO 16$ ;BR IF LOW WORD IS TOO SMALL
7742 033412 023762 005470 005730 24$: CMP HIGOCT,PRMLIM+6(R2) ;SEE IF HIGH WORD IS TOO BIG
7743 033420 101340 BHI 16$ ;BR IF HIGH WORD TOO BIG
7744 033422 001004 BNE 26$ ;BR IF HI WORD NOT EQUAL TO UPPER LIMIT
7745 033424 023762 005466 005726 CMP LOWOCT,PRMLIM+4(R2) ;SEE IF LOW WORD IS TOO LARGE
7746 033432 101333 BHI 16$ ;BR IF LOW WORD IS TOO LARGE
7747 ;UPDATE 32-BIT PARAMETER VALUE IN LIST
7748 033434 013761 005466 005646 26$: MOV LOWOCT,PRMLST(R1) ;PUT LOW WORD INTO LIST
7749 033442 013761 005470 005650 MOV HIGOCT,PRMLST+2(R1) ;PUT HIGH WORD INTO LIST
7750 ;COMPUTE AND TYPE MAX ALLOWABLE WORD COUNT FOR THIS MA
7751 033450 004737 033526 JSR PC,MXWRDC ;GET WORD COUNT IN R1-R0
7752 033454 104401 010455 TYPE MAWRDC ;TYPE "MAX WORD COUNT = "
7753 033460 010037 003166 MOV R0,SUML01
7754 033464 010137 003170 MOV R1,SUMHI1
7755 033470 012746 003166 MOV #SUML01,-(SP) ;PUT POINTER ON STACK
7756 033474 004737 053752 JSR PC,$DB20 ;CONVERT TO OCTAL
7757 033500 004737 054266 JSR PC,$SUPRS ;TYPE IT
7758 033504 104401 001315 TYPE $CRLF ;TYPE <CR>,<LF>
7759 033510 062766 000002 000014 ADD #2,14(SP) ;INCREMENT R1 INDEX
7760 033516 062716 000002 28$: ADD #2,(SP) ;GET NORMAL RETURN PC
7761 033522 104410 30$: RESREG ;RESTORE R0-R5
7762 033524 000207 RTS PC ;RETURN
7763
7764
7765
7766
7767 ;*****
7768 ;*MXWRDC - GET MAX WORD COUNT FOR CURRENT MEM LIMITS AND CURRENT MA,
7769 ;*AND LEAVE THE WORD COUNT IN R1-R0. NO REGISTERS ARE SAVED.
7770 ;*****
7770 033526 013700 005572 MXWRDC: MOV MAHILM,R0 ;GET LO BITS OF MEM LIM
7771 033532 013701 005574 MOV MAHILM+2,R1 ;HI BITS OF MEM LIM
7772 033536 163700 005576 SUB MA,R0 ;SUBTRACT MA FROM MAHILM
7773 033542 005601 SBC R1
7774 033544 163701 005600 SUB MA+2,R1
7775 033550 100413 BMI 27$ ;IF NEGATIVE, RETURN
7776 033552 000241 CLC ;DIVIDE BY 2 TO GET WORDS
7777 033554 006001 ROR R1
7778 033556 006000 ROR R0
7779 033560 062700 000001 ADD #1,R0 ;INCREMENT BY 1 WORD
7780 033564 005501 ADC R1
7781 033566 005701 TST R1
7782 033570 001403 BEQ 27$ ;BR IF WORD COUNT < 65,536 DEC
7783 033572 005000 CLR R0 ;MAKE WORD COUNT = 65,536
7784 033574 012701 000001 MOV #1,R1
7785 033600 000207 27$: RTS PC ;RETURN
7786

```

```

7787
7788
7789
7790
7791
7792
7793
7794 033602 104407
7795 033604 004737 030776
7796 033610 112765 000141 000001
7797 033616 004737 040774
7798 033622 104401 011173
7799 033626 104401 011211
7800 033632 016501 000054
7801 033636 012704 054054
7802 033642 010446
7803 033644 012703 000003
7804 033650 006101
7805 033652 006101
7806 033654 006101
7807 033656 006101
7808 033660 006101
7809 033662 006101
7810 033664 010100
7811 033666 042700 177760
7812 033672 052700 000060
7813 033676 110024
7814 033700 005303
7815 033702 001364
7816 033704 105014
7817 033706 004737 054266
7818 033712 104401 001315
7819 033716 104410
7820 033720 000207
7821
7822
7823
7824
7825
7826
7827
7828
7829 033722 004737 030776
7830 033726 105065 000007
7831 033732 105737 003115
7832 033736 001402
7833 033740 105265 000004
7834 033744 112765 000121 000001
7835 033752 012765 000632 000002
7836 033760 112765 000002 000005
7837 033766 012765 064640 000010
7838 033774 012765 177776 000012
7839 034002 004737 040774
7840 034006 104401 011202
7841 034012 104401 011211
7842 034016 012746 064640

```

```

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

```

```

*****
.SBTTL CRTSER - TYPE CARTRIDGE SERIAL NUMBER
;*THIS SUBROUTINE TYPES "CART. SER. NO. XXXXXXXXXX" (IN OCTAL),
;*WITH LEADING ZEROS SUPPRESSED.
*****
CRTSER: JSR PC,INITSS ;CLEAR S.S. AND PARAMETERS
CLRB P,CS1H(R5) ;SET 22-SECTOR FORMAT
TSTB FORMAT ;CHECK THE ACTUAL FORMAT
BEQ 4$ ;BR IF 22 SECTORS
INCB P,SECT(R5) ;IF 20 SECTORS, READ SECTOR 1
4$: MOVB #RDDATA,P.CMND(R5) ;SET READ COMMAND
MOV #LSTCYL,P.CYLN(R5) ;SET CYL = 632(OCT)
MOVB #LSTTRK,P.TRCK(R5) ;SET TRACK = 2
MOV #RWBUF,P.BALO(R5) ;SET READ-BUFFER ADDRESS
MOV #-2,P.WC(R5) ;SET WORD COUNT TO READ 2 WORDS
JSR PC,DRVCAL ;READ SERIAL NO. IN BSF
TYPE ,CART ;TYPE "CART."
TYPE ,SERNM ;TYPE "SER. NO. "
MOV #RWBUF,-(SP) ;GET POINTER FOR $DB20

```

7843 034022 004737 053752
7844 034026 004737 054266
7845 034032 104401 001315
7846 034036 000207

JSR PC,0#\$DB20 ;CONVERT BINARY TO OCTAL
JSR PC,0#\$SUPRS ;TYPE CART. SERIAL NO. IN OCTAL
TYPE \$CRLF ;TYPE <CR> AND <LF>
RTS PC ;RETURN

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

7858 034040 010046
7859 034042 010146
7860 034044 032777 000400 145066
7861 034052 001407
7862 034054 004737 053150
7863 034060 013700 053250
7864 034064 006200
7865 034066 006200
7866 034070 000403
7867 034072 013700 005502
7868 034076 001406
7869 034100 012701 000016
7870 034104 005301
7871 034106 001376
7872 034110 005300
7873 034112 001372
7874 034114 012601
7875 034116 012600
7876 034120 000207

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

:SBTTL RNDADR - RANDOM CYLINDER ADDRESS GENERATOR
:*THIS SUBROUTINE GENERATES A PSEUDO-RANDOM 9-BIT CYLINDER
:*ADDRESS, AND LEAVES IT IN "CYLNDR". IT REQUIRES THE SYSMAC
:*SUBROUTINE \$RAND.
:* CALL - JSR PC,RNDADR

7887 034122 004737 053150
7888 034126 013737 053250 005504
7889 034134 042737 177000 005504
7890 034142 022737 000632 005504
7891 034150 002003
7892 034152 042737 000400 005504
7893 034160 000207

RNDADR: JSR PC,\$RAND ;GENERATE 2 16-BIT RANDOM NUMBERS
MOV \$LONUM,CYLNR ;GET A RANDOM NUMBER
BIC #177000,CYLNR ;SCALE IT TO 9 BITS
CMP #632,CYLNR ;SEE IF CYL IS TOO BIG
BGE 2\$;BR IF CYL IS OK
BIC #BIT08,CYLNR ;SCALE DOWN TO A VALID CYLINDER
2\$: RTS PC ;RETURN

:* LODSEC - THIS SUBROUTINE LOADS THE CONTENTS OF RO INTO ALL

7894
7895
7896
7897
7898


```

7899          ;*256(DEC) WORDS OF THE DATA BUFFER (RWBUFF).
7900          ;*****
7901 034162   104407          LODSEC: SAVREG          ;SAVE R0-R5
7902 034164   012701   064640      MOV      #RWBUFF,R1      ;GET ADDRESS OF DATA BUF INTO R1
7903 034170   010021          2$:  MOV      R0,(R1)+          ;PUT WORD INTO BUFFER
7904 034172   020127   065640      CMP      R1,#RWBUFF+512. ;SEE IF 256 WORDS WRITTEN YET
7905 034176   001374          BNE     2$              ;BR IF NOT DONE YET
7906 034200   104410          RESREG          ;RESTORE R0-R5
7907 034202   000207          RTS      PC           ;RETURN
7908
7909
7910
7911          ;*****
7912          ;* TRKCHK - THIS SUBROUTINE DOES A WRITE-CHECK OF SECTOR FS ON CYL
7913          ;*FC, FOR EACH OF TRACKS 0, 1, AND 2. THE DATA IS COMPARED TO THE
7914          ;*TRACK NUMBER + 100(OCT) FOR THAT PARTICULAR BLOCK OF DATA WRITTEN.
7915          ;*THE PARAMETER BLOCK MUST BE PRE-LOADED WITH THE PROPER PARAMETERS.
7916          ;*****
7917 034204   104407          TRKCHK: SAVREG          ;SAVE R0-R5
7918 034206   016546   000004      MOV      P.SECT(R5),-(SP) ;SAVE TRACK AND SECTOR PARAMETERS
7919 034212   105065   000005      CLRB   P.TRCK(R5)       ;CLEAR THE TRACK NO.
7920          ;LOAD THE DATA BUFFER WITH THE CURRENT TRACK NO.
7921 034216   116500   000005      2$:  MOVB  P.TRCK(R5),R0    ;GET TRACK NO. INTO R0
7922 034222   062700   000100      ADD     #100,R0         ;ADD 100(OCT) TO TRACK NO.
7923 034226   004737   034162      JSR    PC,LODSEC       ;LOAD DATA BUF WITH TRACK NO. + 100(OCT)
7924 034232   112765   000131   000001      MOVB  #WRTCHK,P.CMND(R5) ;SET WRITE CHECK COMMAND
7925 034240   004737   040774      JSR    PC,DRVCAL       ;PERFORM THE WRITE CHECK
7926 034244   105265   000005      INCB  P.TRCK(R5)       ;INCREMENT THE TRACK NO.
7927 034250   122765   000003   000005      CMPB  #3,P.TRCK(R5)    ;SEE IF DONE WITH ALL TRACKS
7928 034256   001357          BNE     2$              ;BR IF NOT DONE YET
7929 034260   012665   000004      MOV      (SP)+,P.SECT(R5) ;RESTORE TRACK AND SECTOR PARAMETERS
7930 034264   104410          RESREG          ;RESTORE R0-R5
7931 034266   000207          RTS      PC           ;RETURN
7932
7933
7934
7935          ;*****
7936          ;*LODP14 - GENERATE SIXTEEN PSEUDO-RANDOM NUMBERS AND LOAD THEM
7937          ;*INTO THE PATTERN 14 TABLE.
7938          ;*****
7939 034270   104407          LODP14: SAVREG          ;SAVE R0-R5
7940 034272   012701   000010      MOV      #8,R1          ;INIT LOOP COUNTER TO 8.
7941 034276   012700   006724      MOV      #PAT14,R0      ;GET ADDRESS OF PATTERN 14 BUFFER
7942 034302   004737   053150      4$:  JSR    PC,$RAND       ;GENERATE 2 16-BIT RANDOM NUMBERS
7943 034306   013720   053250      MOV      $LONUM,(R0)+   ;PUT ONE NUMBER INTO PATTERN
7944 034312   013720   053246      MOV      $HINUM,(R0)+   ;PUT OTHER NO. INTO PATTERN
7945 034316   005301          DEC     R1              ;SEE IF 16 WORDS LOADED YET
7946 034320   001370          BNE     4$              ;BR IF NOT YET
7947 034322   104410          RESREG          ;RESTORE R0-R5
7948 034324   000207          RTS      PC           ;RETURN
7949
7950
7951
7952          ;*****
7953          ;SBTTL FINADR - COMPUTE FINAL PACK ADDRESS
7954          ;*THIS SUBROUTINE IS USED AFTER A DATA TRANSFER HAS COMPLETED, TO
    
```

FINADR - COMPUTE FINAL PACK ADDRESS

```

7955
7956
7957
7958
7959
7960
7961
7962 034326 104407
7963 034330 013746 005570
7964 034334 005416
7965 034336 005046
7966 034340 116616 000003
7967 034344 005066 000002
7968 034350 116566 000004 000002
7969 034356 066616 000002
7970 034362 005066 000002
7971 034366 012700 000026
7972 034372 105737 003115
7973 034376 001402
7974 034400 012700 000024
7975 034404 020016 19$:
7976 034406 101004
7977 034410 160016
7978 034412 005266 000002
7979 034416 000772
7980 034420 112637 005567 20$:
7981 034424 005046
7982 034426 116516 000005
7983 034432 066616 000002
7984 034436 005066 000002
7985 034442 122716 000003 21$:
7986 034446 101005
7987 034450 162716 000003
7988 034454 005266 000002
7989 034460 000770
7990 034462 112637 005566 22$:
7991 034466 066516 000002
7992 034472 011637 005564
7993 034476 005726
7994 034500 104410
7995 034502 000207

```

```

; *COMPUTE THE FINAL PACK ADDRESS AT TERMINATION. THE 2'S COMP. OF THE
; *WORD NO. AT THE POINT OF INTEREST IS PASSED IN LASTWC. THIS IS USED WITH
; *P.CYLN(R5), P.TRCK(R5), AND P.SECT(R5) TO COMPUTE THE CORRESPONDING
; *PACK ADDRESS, WHICH IS RETURNED IN FINCYL, FINTRK, AND FINSEC.
; *THE SUBROUTINE IS USED TO DETERMINE THE PACK ADDRESS OF A SOFTWARE
; *DATA MISCOMPARE.
; *****

```

```

FINADR: SAVREG ;SAVE R0-R5
MOV LASTWC,-(SP) ;STORE WORD COUNT
NEG (SP) ;MAKE IT POSITIVE
18$: CLR -(SP) ;MAKE ROOM ON STACK
MOVB 3(SP),(SP) ;STORE NO. OF SECTORS TRANSFERRED
CLR 2(SP) ;CLEAR LOCATION ON STACK
MOVB P.SECT(R5),2(SP) ;STORE STARTING SECTOR
ADD 2(SP),(SP) ;DETERMINE FINAL SECTOR ADDRESS
CLR 2(SP) ;CLEAR NO. OF TRACKS TRANSFERRED
MOV #22,R0 ;SET FOR 22 SECTORS
TSTB FORMAT ;DETERMINE THE FORMAT
BEQ 19$ ;BR IF 22 SECTORS
MOV #20,R0 ;SET FOR 20 SECTORS
19$: CMP R0,(SP) ;CHECK FOR SECTOR OVERFLOW
BHI 20$ ;NO, CHECK IF SECTOR CORRECT
SUB R0,(SP) ;DECREMENT SECTOR COUNT BY 20 OR 22
INC 2(SP) ;INCREMENT TRACKS TRANSFERRED
BR 19$ ;CHECK FOR SECTOR OVERFLOW
20$: MOVB (SP)+,FINSEC ;STORE FINAL SECTOR
CLR -(SP) ;MAKE ROOM FOR TRACKS TRANSFERRED
MOVB P.TRCK(R5),(SP) ;STORE STARTING TRACK
ADD 2(SP),(SP) ;DETERMINE FINAL TRACK ADDRESS
CLR 2(SP) ;CLEAR FINAL CYLINDER
21$: CMPB #3,(SP) ;CHECK FOR TRACK OVERFLOW
BHI 22$ ;NO, CHECK FINAL TRACK
SUB #3,(SP) ;DECREMENT TRACK COUNT BY 3
INC 2(SP) ;INCR CYL COUNT
BR 21$ ;CHECK FOR TRACK OVERFLOW
22$: MOVB (SP)+,FINTRK ;STORE FINAL TRACK
ADD P.CYLN(R5),(SP) ;CALCULATE FINAL CYLINDER
MOV (SP),FINCYL ;STORE FINAL CYLINDER
TST (SP)+ ;CLEAN OFF STACK
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

```

7996
7997
7998
7999
8000
8001
8002
8003
8004
8005
8006
8007
8008
8009
8010 034504 104407

```

```

; *****
;SBTTL LODBUF - LOAD THE READ/WRITE DATA BUFFER
; *THIS SUBROUTINE LOADS THE READ/WRITE DATA BUFFER (POINTED TO BY
; * PMA AND PMA+2) WITH THE APPROPRIATE REPEATING DATA PATTERN. IF
; *PARAMETER PATRN = 0 ON ENTRY, THE DATA WHICH IS LOADED IS COMPRISED
; *OF ALL THE PATTERNS 00-15 (QUICK VERIFY DEFAULT DATA TEST).
; *IF PATRN IS NOT 0, THE NO. OF WORDS IN THE WORD WDSXFR
; *OF THE PATTERN IDENTIFIED BY THE NO. OF THE BIT SET IN R1 ON
; *ENTRY, ARE LOADED INTO THE BUFFER.
; *****

```

```

LODBUF: SAVREG ;SAVE R0-R5

```



K12

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 154
 LODBUF - LOAD THE READ/WRITE DATA BUFFER

SEQ 0153

8011	034506	013702	005562			MOV	WDSXFR,R2	:GET NO. OF WORDS
8012	034512	005737	005534			TST	PATRN	:SEE IF QUICK VERIFY DATA TEST DESIRED
8013	034516	001007				BNE	3\$:BR IF NOT QUICK VERIFY
8014	034520	012700	006024			MOV	#PAT00,R0	:SET DATA PATTERN STARTING ADDRESS
8015	034524	012746	000400			MOV	#256,-(SP)	:SET PATTERN WORD COUNT
8016	034530	012701	064640			MOV	#RWBUF,R1	:SET BUFFER ADDRESS
8017	034534	000463				BR	30\$:PROCEED
8018	034536	005000			3\$:	CLR	R0	:INIT PATTERN NUMBER
8019	034540	032701	000001		4\$:	BIT	#BIT0,R1	:SEE IF THIS BIT IS SET
8020	034544	001003				BNE	6\$:BR IF THIS BIT IS SET
8021	034546	005200				INC	R0	:INCREMENT PATTERN NO.
8022	034550	006201				ASR	R1	:SHIFT TO EXAMINE NEXT BIT
8023	034552	000772				BR	4\$:BR TO CHECK NEXT BIT
8024	034554	006300			6\$:	ASL	R0	:MULTIPLY PATTERN NO. BY 32(DEC)
8025	034556	006300				ASL	R0	
8026	034560	006300				ASL	R0	
8027	034562	006300				ASL	R0	
8028	034564	006300				ASL	R0	
8029	034566	062700	006024			ADD	#PAT00,R0	:GET ADDRESS OF DESIRED PATTERN
8030	034572	012746	000020			MOV	#16,-(SP)	:SET PATTERN WORD COUNT
8031	034576	013701	005602			MOV	PMA,R1	:SET BUFFER ADDRESS
8032	034602	005737	056246			TST	\$KT11	:SEE IF MEM MGT PRESENT
8033	034606	100036				BPL	30\$:BR IF NOT PRESENT
8034						;BUFFER LOAD LOOP FOR MEM MGT STARTS HERE		
8035	034610	013737	003176	172354		MOV	SAVPAR,2#KIPAR6	:SET UP WORKING PAR
8036	034616	052737	000001	177572		BIS	#BIT0,2#SRO	:TURN ON MEMORY MANAGEMENT
8037	034624	042701	160000			BIC	#160000,R1	:FORCE RELOCATION THRU KIPAR6
8038	034630	052701	140000			BIS	#140000,R1	
8039	034634	010003			22\$:	MOV	R0,R3	:GET A COPY OF PATTERN ADDRESS
8040	034636	011604				MOV	(SP),R4	:INIT PATTERN WORD COUNT
8041	034640	012321			24\$:	MOV	(R3)+,(R1)+	:LOAD A DATA WORD INTO BUFFER
8042	034642	032701	020000			BIT	#BIT13,R1	:SEE IF OVERFLOW TO NEXT PAGE
8043	034646	001405				BEQ	26\$:BR IF NO OVERFLOW
8044	034650	062737	000200	172354		ADD	#200,2#KIPAR6	:INCREMENT PAR BY 4K FOR NEW PAGE
8045	034656	042701	020000			BIC	#BIT13,R1	:SET PAGE = 6 AGAIN
8046	034662	005302			26\$:	DEC	R2	:DECREMENT WORD COUNTER
8047	034664	001403				BEQ	28\$:BR IF ALL DONE
8048	034666	005304				DEC	R4	:DECREMENT PATTERN WORD COUNT
8049	034670	001363				BNE	24\$:BR IF NOT DONE WITH PATTERN YET
8050	034672	000760				BR	22\$:BR TO REPEAT THE PATTERN
8051	034674	042737	000001	177572	28\$:	BIC	#BIT0,2#SRO	:DISABLE MEMORY MANAGEMENT
8052	034702	000410				BR	44\$:GO TO RETURN
8053						;BUFFER LOAD LOOP FOR NO MEM MGT STARTS HERE		
8054	034704	010003			30\$:	MOV	R0,R3	:GET A COPY OF PATTERN ADDRESS
8055	034706	011604				MOV	(SP),R4	:INIT PATTERN WORD COUNT
8056	034710	012321			34\$:	MOV	(R3)+,(R1)+	:LOAD A DATA WORD INTO BUFFER
8057	034712	005302				DEC	R2	:DECREMENT WORD COUNTER
8058	034714	001403				BEQ	44\$:BR IF ALL DONE
8059	034716	005304				DEC	R4	:DECREMENT PATTERN WORD COUNT
8060	034720	001373				BNE	34\$:BR IF NOT DONE WITH PATTERN YET
8061	034722	000770				BR	30\$:BR TO REPEAT THE PATTERN
8062	034724	005726			44\$:	TST	(SP)+	:POP THE STACK
8063	034726	104410				RESREG		:RESTORE R0-R5
8064	034730	000207				RTS	PC	:RETURN
8065								
8066								

```

8067
8068
8069
8070
8071
8072
8073
8074
8075
8076
8077 034732 104407
8078 034734 112737 000040 063157
8079 034742 012737 000007 064532
8080 034750 013702 005562
8081 034754 005037 005532
8082 034760 004737 042206
8083 034764 005737 005534
8084 034770 001007
8085 034772 012700 006024
8086 034776 012746 000400
8087 035002 012701 064640
8088 035006 000466
8089 035010 005000
8090 035012 032701 000001
8091 035016 001003
8092 035020 005200
8093 035022 006201
8094 035024 000772
8095 035026 006300
8096 035030 006300
8097 035032 006300
8098 035034 006300
8099 035036 006300
8100 035040 062700 006024
8101 035044 012746 000020
8102 035050 013701 005602
8103 035054 005737 056246
8104 035060 100041
8105
8106 035062 013737 003176 172354
8107 035070 052737 000001 177572
8108 035076 042701 160000
8109 035102 052701 140000
8110 035106 010003
8111 035110 011604
8112 035112 022321
8113 035114 001404
8114 035116 013737 172354 001216
8115 035124 000432
8116 035126 032701 020000
8117 035132 001405
8118 035134 062737 000200 172354
8119 035142 042701 020000
8120 035146 005302
8121 035150 001002
8122 035152 000137 035616

;*****
;SBTTL CMPBUF - SOFTWARE COMPARE DATA
;THIS SUBROUTINE PERFORMS A SOFTWARE COMPARE OF THE R/W DATA BUFFER (POINTED
;TO BY PMA AND PMA+2) TO THE APPROPRIATE REPEATING DATA PATTERN. IF
;PARAMETER PATRN = 0 ON ENTRY, THE DATA IS COMPRISED OF ALL THE PATTERNS
;00-15 (QUICK VERIFY DEFAULT DATA TEST).
;IF PATRN IS NOT 0, THE NO. OF THE BIT SET IN R1 ON ENTRY IDENTIFIES THE
;REPEATING DATA PATTERN TO COMPARE AGAINST.
;*****
CMPBUF: SAVREG ;SAVE R0-R5
MOV #40,DH701+38. ;RESTORE ERROR MSG PARAMS
MOV #7,DF25+2
MOV WDSXFR,R2 ;SET NO. OF WORDS
CLR SCRACH ;CLEAR COMPARE ERROR COUNT
JSR PC,REPSUP ;STORE PREV CMND FOR POSS. PRINT
TST PATRN ;SEE IF THIS IS QUICK VERIFY DEFAULT DATA TEST
BNE 3$ ;BR IF NOT
MOV #PAT00,R0 ;GET DATA PATTERN STARTING ADDR
MOV #256,-(SP) ;SET PATTERN WORD COUNT
MOV #RWBUF,R1 ;SET BUFFER ADDRESS
BR 30$ ;PROCEED
3$: CLR R0 ;INIT PATTERN NO.
4$: BIT #BIT0,R1 ;SEE IF THIS BIT IS SET
BNE 6$ ;BR IF THIS BIT IS SET
INC R0 ;INCREMENT PATTERN NUMBER
ASR R1 ;SHIFT TO EXAMINE NEXT BIT
BR 4$ ;BR TO CHECK NEXT BIT
6$: ASL R0 ;MULTIPLY PATTERN NO. BY 32(DEC)
ASL R0
ASL R0
ASL R0
ADD #PAT00,R0 ;GET ADDRESS OF DESIRED PATTERN
MOV #16,-(SP) ;PATTERN WORD COUNT
MOV PMA,R1 ;SET BUFFER ADDRESS
TST $KT11 ;SEE IF MEM MGT PRESENT
BPL 30$ ;BR IF NOT PRESENT
;COMPARE LOOP FOR MEM MGT STARTS HERE
MOV SAVPAR,@#KIPAR6 ;SET UP WORKING PAR
BIS #BIT0,@#SR0 ;TURN ON MEM MGT
BIC #160000,R1 ;FORCE RELOCATION THRU KIPAR6
BIS #140000,R1
22$: MOV R0,R3 ;GET A COPY OF PATTERN ADDRESS
MOV (SP),R4 ;INIT PATTERN WORD COUNT
24$: CMP (R3)+,(R1)+ ;COMPARE DATA WORD TO PATTERN WORD
BEQ 25$ ;BR IF NO ERROR
MOV @#KIPAR6,$REG16 ;SET PAR FOR PRINTOUT
BR 35$ ;GO HANDLE ERROR
25$: BIT #BIT13,R1 ;SEE IF OVERFLOW TO NEXT PAGE
BEQ 26$ ;BR IF NO OVERFLOW
ADD #200,@#KIPAR6 ;INCR PAR BY 4K FOR NEW PAGE
BIC #BIT13,R1 ;SET PAGE = 6 AGAIN
26$: DEC R2 ;DECREMENT WORD COUNTER
BNE 27$ ;BR IF NOT ALL DONE YET
JMP 54$ ;ALL DONE - GET OUT

```

M12

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 156
 CMPBUF - SOFTWARE COMPARE DATA

SEQ 0155

0123	035156	005304			27\$:	DEC R4	;	DECREMENT PATTERN WORD COUNT
0124	035160	001354				BNE 24\$;	BR IF NOT DONE WITH PATTERN YET
0125	035162	000751				BR 22\$;	BR TO REPEAT THE PATTERN
0126							;	COMPARE LOOP FOR NO MEM MGT STARTS HERE
0127	035164	010003			30\$:	MOV RO,R3	;	GET COPY OF PATTERN ADDRESS
0128	035166	011604				MOV (SP),R4	;	INIT PATTERN WORD COUNT
0129	035170	022321			34\$:	CMP (R3)+,(R1)+	;	COMPARE DATA WORD TO PATTERN WORD
0130	035172	001002				BNE 31\$;	BR IF COMPARE ERROR
0131	035174	000137	035576			JMP 40\$;	JUMP IF DATA COMPARES OK
0132	035200	105037	063157		31\$:	CLRB DH701+38.	;	ADJUST DATA HEADER FOR MSG
0133	035204	012737	000005	064532		MOV #5,DF25+2	;	ADJUST ERROR DATA WORD COUNT
0134							;	COMMON COMPARE ERROR HANDLER
0135	035212	010237	001202		35\$:	MOV R2,\$REG10	;	GET WORD NO.
0136	035216	163737	005562	001202		SUB WDSXFR,\$REG10		
0137	035224	013737	001202	005570		MOV \$REG10, LASTWC	;	GET 2'S COMP OF WORD NO.
0138	035232	004737	034326			JSR PC,FINADR	;	COMPUTE ACTUAL PACK ADRS
0139	035236	104407				SAVREG	;	SAVE RO-R5
0140	035240	013700	005564			MOV FINCYL,RO	;	GET CYL
0141	035244	113701	005566			MOVFB FINTRK,R1	;	GET TRACK
0142	035250	113702	005567			MOVFB FINSEC,R2	;	GET SECTOR
0143	035254	004737	040670			JSR PC,BDSRCK	;	SEE IF THIS SECTOR LISTED BAD
0144	035260	104410				RESREG	;	RESTORE RO-R5
0145	035262	032737	001000	005474		BIT #BADSEC,RECODE		
0146	035270	001132				BNE 50\$;	BR IF LISTED- DON'T REPORT ERROR
0147	035272	005737	005532			TST SCRACH	;	CHECK THE ERROR COUNT
0148	035276	001024				BNE 36\$;	BR IF THIS IS NOT FIRST ERROR
0149	035300	105737	003134			TSTB UBMPRS	;	SEE IF UNIBUS MAP PRESENT
0150	035304	001411				BEQ 46\$;	BR IF NOT
0151	035306	013737	005260	001174		MOV CRMPHO,\$REG5	;	GET CURRENT MAP REG 0
0152	035314	013737	005256	001176		MOV CRMPLO,\$REG6		
0153	035322	104116				ERROR 116	;	DATA MISCOMPARE (11/70)
0154	035324	104117				ERROR 117		
0155	035326	000401				BR 48\$		
0156	035330	104034			46\$:	ERROR 34	;	TYPE HEADING FOR ERROR MSG
0157	035332	012737	177777	001174	48\$:	MOV #-1,\$REG5	;	INIT CYL NO.
0158	035340	005037	001176			CLR \$REG6		
0159	035344	005037	001200			CLR \$REG7		
0160	035350	005237	005532		36\$:	INC SCRACH	;	INCREMENT THE ERROR COUNT
0161	035354	032777	000001	143556		BIT #BIT0,DSWR	;	SEE IF ALL ERRORS SHOULD BE REPORTED
0162	035362	001004				BNE 38\$;	BR TO REPORT ALL ERRORS
0163	035364	022737	000012	005532		CMP #10.,SCRACH	;	SEE IF 10(DEC) ERRORS YET
0164	035372	002511				BLT 54\$;	BR IF ERROR LIMIT EXCEEDED
0165	035374	023737	001174	005564	38\$:	CMP \$REG5,FINCYL	;	SEE IF DIFFERENT CYL
0166	035402	001010				BNE 42\$;	BR IF YES
0167	035404	123737	001176	005566		CMPB \$REG6,FINTRK	;	SEE IF DIFFERENT TRACK
0168	035412	001004				BNE 42\$;	BR IF YES
0169	035414	123737	001200	005567		CMPB \$REG7,FINSEC	;	SEE IF DIFFERENT SECTOR
0170	035422	001412				BEQ 44\$;	BR IF SAME PACK ADDRESS
0171	035424	013737	005564	001174	42\$:	MOV FINCYL,\$REG5	;	SET NEW PACK ADRS FOR PRINTOUT
0172	035432	113737	005566	001176		MOVFB FINTRK,\$REG6		
0173	035440	113737	005567	001200		MOVFB FINSEC,\$REG7		
0174	035446	104115				ERROR 115	;	TYPE NEW PACK ADDRESS
0175	035450	005437	001202		44\$:	NEG \$REG10	;	GET WORD NO.
0176	035454	016337	177776	001204		MOV -2(R3),\$REG11	;	GET GOOD DATA
0177	035462	016137	177776	001206		MOV -2(R1),\$REG12	;	GET BAD DATA
0178	035470	013737	001202	001212		MOV \$REG10,\$REG14	;	COMPUTE PHYSICAL ADDRESS

8179	035476	005037	001210		CLR	\$REG13	
8180	035502	006137	001212		ROL	\$REG14	
8181	035506	006137	001210		ROL	\$REG13	
8182	035512	063737	005602	001212	ADD	PMA,\$REG14	
8183	035520	005537	001210		ADC	\$REG13	
8184	035524	063737	005604	001210	ADD	PMA+2,\$REG13	
8185	035532	010137	001214		MOV	R1,\$REG15	;GET VIRT. ADRS FOR PRINTOUT
8186	035536	162737	000002	001214	SUB	#2,\$REG15	
8187	035544	104063			ERROR	63	;TYPE GOOD AND BAD DATA, MEM. ADRS.
8188	035546	032777	000100	143364	BIT	#BIT6,\$SWR	;SEE IF JUST 1 ERROR SHOULD BE REPORTED
8189	035554	001020			BNE	54\$;BR IF JUST 1 ERROR SHOULD BE REPORTED
8190	035556	005737	005534	50\$:	TST	PATRN	;SEE IF DEFAULT DATA TEST
8191	035562	001405			BEQ	40\$;BR IF YES
8192	035564	005737	056246		TST	\$KT11	;SEE IF MEM MGT PRESENT
8193	035570	100002			BPL	40\$;BR IF NOT PRESENT
8194	035572	000137	035126		JMP	25\$;GO HANDLE MEM MGT
8195	035576	005302		40\$:	DEC	R2	;DECREMENT WORD COUNTER
8196	035600	001406			BEQ	54\$;BR IF ALL DONE
8197	035602	005304			DEC	R4	;DECREMENT PATTERN WORD COUNT
8198	035604	001002			BNE	53\$;BR IF NOT DONE WITH PATTERN YET
8199	035606	000137	035164		JMP	30\$;JUMP TO REPEAT THE PATTERN
8200	035612	000137	035170	53\$:	JMP	34\$	
8201	035616	005737	056246	54\$:	TST	\$KT11	;SEE IF MEM MGT PRESENT
8202	035622	100003			BPL	56\$;BR IF NOT PRESENT
8203	035624	042737	000001	177572	BIC	#BIT0,\$#SR0	;DISABLE MEM MGT
8204	035632	005726		56\$:	TST	(SP)+	;POP THE STACK
8205	035634	104410			RESREG		;RESTORE R0-R5
8206	035636	000207			RTS	PC	;RETURN

8207								
8208								
8209								
8210								
8211								
8212								
8213								
8214								
8215								
8216								
8217								
8218								
8219								
8220	035640	122737	000001	003110	CTLOUT:	CMPB	#1,TSTING	;SEE IF CURRENTLY RUNNING TESTS
8221	035646	001077				BNE	10\$;BR IF NOT RUNNING TESTS
8222	035650	005737	005522			TST	INTCHR	;SEE IF ANY TTY INPUT
8223	035654	001474				BEQ	10\$;BR IF NO INPUT
8224	035656	105737	003106			TSTB	MDFLAG	;SEE IF DEFAULT MODE RUN
8225	035662	001446				BEQ	12\$;BR IF YES
8226	035664	122737	000003	005522		CMPB	#003,INTCHR	;SEE IF (↑C) TYPED
8227	035672	001033				BNE	4\$;BR IF NOT (↑C)
8228	035674	012700	014532			MOV	#DRVTST,R0	;SET RETURN ADDR = DRVTST
8229	035700	105737	003124	2\$:		TSTB	XOVLAD	;SEE IF XXDP CURRENTLY OVERLAID
8230	035704	001402				BEQ	6\$;BR IF NOT
8231	035706	004737	030536			JSR	PC,GETXDP	;RESTORE SAVED XXDP, IF NECESSARY
8232	035712	000005		6\$:		RESET		;RESET ALL DEVICES
8233	035714	005037	005522			CLR	INTCHR	;CLEAR TTY CHAR BUFFER WORD
8234	035720	005037	001304			CLR	\$TIMES	;CLEAR THE ITER. COUNT

```

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

```



```

036144 104407
036146 005004
036150 012703 000001
036154 000403
036156 104407
036160 012704 177777
036164 013702 005562
036170 013701 005602
036174 005737 056246
036200 100012
036202 042701 160000
036206 052701 140000
036212 013737 003176 172354
036220 052737 000001 177572
036226 010321
036230 005704
036232 001001
036234 005203
036236 005737 056246
036242 100010
036244 032701 020000
036250 001405
036252 062737 000200 172354
036260 042701 020000
036264 005302
036266 001357
036270 005737 056246
036274 100003
036276 042737 000001 177572
036304 104410
036306 000207
036310 104407
036312 005004
036314 012703 000001
036320 000403
036322 104407
036324 012704 177777
036330 013702 005562
036334 013701 005602
036340 005037 005532
036344 112737 000040 063157

```

```

:LDMEM2 - LOAD MEMORY WITH REPEATED WORD, IN R3 ON ENTRY.
: *BOTH SUBROUTINES REQUIRE MEMORY ADDRESS IN PMA, PMA+2, AND WORD COUNT
: *MUST BE IN WDSXFR.
: *****

```

```

LDMEM1: SAVREG ;SAVE R0-R5
        CLR R4 ;R4=0 INDICATES LDMEM1
        MOV #1,R3 ;INIT DATA WORD TO 1
        BR LDMO ;PROCEED
LDMEM2: SAVREG ;SAVE R0-R5
        MOV #-1,R4 ;R4=177777 INDICATES LDMEM2
LDMO: MOV WDSXFR,R2 ;GET NO. OF WORDS
        MOV PMA,R1 ;GET MEM ADRS
        TST $KT11 ;SEE IF MEM MGT
        BPL 6$ ;BR IF NOT
        BIC #160000,R1 ;FORCE RELOCATION THRU PAR6
        BIS #140000,R1
        MOV SAVPAR,3#KIPAR6 ;INIT PAR6
        BIS #BIT0,3#SR0 ;TURN ON MEM MGT
6$: MOV R3,(R1)+ ;LOAD A WORD INTO BUFFER
        TST R4 ;SEE WHICH DATA DESIRED
        BNE 8$ ;BR IF NOT INCREMENTING DATA
        INC R3 ;INCREMENT THE DATA
8$: TST $KT11 ;SEE IF MEM MGT
        BPL 10$ ;BR IF NOT
        BIT #BIT13,R1 ;SEE IF OVERFLOW TO NEXT PAGE
        BEQ 10$ ;BR IF NO OVERFLOW
        ADD #200,3#KIPAR6 ;INCR PAR BY 4K FOR NEW PAGE
        BIC #BIT13,R1 ;SET PAGE = 6 AGAIN
10$: DEC R2 ;DECREMENT COUNTER
        BNE 6$ ;BR IF NOT DONE LOADING YET
        TST $KT11 ;SEE IF MEM MGT
        BPL 12$ ;BR IF NOT
        BIC #BIT0,3#SR0 ;TURN OFF MEM MGT
12$: RESREG ;RESTORE R0-R5
        RTS PC ;RETURN

```

```

: *****
: *CKMEM1 - PERFORM SOFTWARE COMPARE OF MEMORY, WITH INCREASING
: *NUMBERS, STARTING WITH 1.
: *CKMEM2 - PERFORM SOFTWARE COMPARE OF MEMORY, WITH REPEATED DATA WORD
: *IN R3 ON ENTRY.
: *BOTH SUBROUTINES REQUIRE MEM ADRS IN PMA, PMA+2, AND WORD COUNT MUST BE
: *IN WDSXFR.
: *****

```

```

CKMEM1: SAVREG ;SAVE R0-R5
        CLR R4 ;R4=0 INDICATES CKMEM1
        MOV #1,R3 ;INIT DATA TO 1
        BR CKMO ;PROCEED
CKMEM2: SAVREG ;SAVE R0-R5
        MOV #-1,R4 ;R4=177777 INDICATES CKMEM2
CKMO: MOV WDSXFR,R2 ;GET NO. OF WORDS
        MOV PMA,R1 ;GET MEM ADRS
        CLR SCRACH
        MOVB #40,DH701+38. ;RESTORE ERROR MSG PARAMS

```


8347	036352	012737	000007	064532	MOV	#7,DF25+2	
8348	036360	004737	042206		JSR	PC,REPSUP	:STORE PREV CMND FOR POSS. PRINT
8349	036364	005737	056246		TST	\$KT11	:SEE IF MEM MGT
8350	036370	100012			BPL	6\$:BR IF NOT
8351	036372	042701	160000		BIC	#160000,R1	:FORCE RELOCATION THRU PAR6
8352	036376	052701	140000		BIS	#140000,R1	
8353	036402	013737	003176	172354	MOV	SAVPAR,2#KIPAR6	:INIT PAR6
8354	036410	052737	000001	177572	BIS	#BIT0,2#SR0	:TURN ON MEM MGT
8355	036416	020321			CMP	R3,(R1)+	:COMPARE A DATA WORD
8356	036420	001575			BEQ	16\$:BR IF DATA COMPARES OK
8357							
8358	036422	010237	001202		MOV	R2,\$REG10	:GET WORD NO.
8359	036426	163737	005562	001202	SUB	WD\$XFR,\$REG10	
8360	036434	013737	001202	005570	MOV	\$REG10,LASTWC	:GET 2'S COMP OF WORD NO.
8361	036442	004737	034326		JSR	PC,FINADR	:COMPUTE ACTUAL PACK ADRS
8362	036446	104407			SAVREG		:SAVE R0-R5
8363	036450	013700	005564		MOV	FINCYL,R0	:GET CYL
8364	036454	113701	005566		MOVB	FINTRK,R1	:GET TRACK
8365	036460	113702	005567		MOVB	FINSEC,R2	:GET SECTOR
8366	036464	004737	040670		JSR	PC,BDSACK	:SEE IF THIS SECTOR LISTED BAD
8367	036470	104410			RESREG		:RESTORE R0-R5
8368	036472	032737	001000	005474	BIT	#BADSEC,RECODE	
8369	036500	001145			BNE	16\$:BR IF LISTED- DON'T REPORT ERROR
8370	036502	005737	056246		TST	\$KT11	:SEE IF MEM MGT
8371	036506	100406			BMI	8\$:BR IF YES
8372	036510	105037	063157		CLAB	DH701+38.	:ADJUST DATA HEADER FOR MSG
8373	036514	012737	000005	064532	MOV	#5,DF25+2	:ADJ. ERROR DATA WORD COUNT
8374	036522	000403			BR	9\$	
8375	036524	013737	172354	001216	MOV	2#KIPAR6,\$REG16	:SET PAR FOR PRINTOUT
8376	036532	005737	005532		TST	SCRACH	:SEE IF FIRST ERROR IN THIS BLOCK
8377	036536	001024			BNE	10\$:BR IF NOT FIRST ERROR
8378	036540	105737	003134		TSTB	UBMPRS	:SEE IF UNIBUS MAP PRESENT
8379	036544	001411			BEQ	17\$:BR IF NOT
8380	036546	013737	005260	001174	MOV	CRMPHO,\$REG5	:CURRENT UB MAP REG 0
8381	036554	013737	005256	001176	MOV	CRMPLO,\$REG6	
8382	036562	104116			ERROR	116	:DATA MISCOMPARE (11/70)
8383	036564	104117			ERROR	117	
8384	036566	000401			BR	11\$	
8385	036570	104034			ERROR	34	:TYPE HEADING FOR ERROR MSG
8386	036572	012737	177777	001174	MOV	#-1,\$REG5	:INIT CYL NO.
8387	036600	005037	001176		CLR	\$REG6	
8388	036604	005037	001200		CLR	\$REG7	
8389	036610	005237	005532		INC	SCRACH	:INCREMENT THE ERROR COUNT
8390	036614	032777	000001	142316	BIT	#BIT0,2SWR	:SEE IF ALL ERRORS SHOULD BE REPORTED
8391	036622	001004			BNE	12\$:BR TO REPORT ALL ERRORS
8392	036624	022737	000012	005532	CMP	#10.,SCRACH	:SEE IF 10(DEC) ERRORS YET
8393	036632	002512			BLT	21\$:BR IF ERROR LIMIT EXCEEDED
8394	036634	023737	001174	005564	CMP	\$REG5,FINCYL	:SEE IF DIFFERENT CYL
8395	036642	001010			BNE	14\$:BR IF YES
8396	036644	123737	001176	005566	CMPB	\$REG6,FINTRK	:SEE IF DIFFERENT TRACK
8397	036652	001004			BNE	14\$:BR IF YES
8398	036654	123737	001200	005567	CMPB	\$REG7,FINSEC	:SEE IF DIFFERENT SECTOR
8399	036662	001412			BEQ	15\$:BR IF SAME PACK ADDRESS
8400	036664	013737	005564	001174	MOV	FINCYL,\$REG5	:SET NEW PACK ADR FOR PRINTOUT
8401	036672	113737	005566	001176	MOVB	FINTRK,\$REG6	
8402	036700	113737	005567	001200	MOVB	FINSEC,\$REG7	

```

0403 036706 104115          ERROR 115          ;TYPE NEW PACK ADRS
0404 036710 005437 001202    15$: NEG $REG10      ;GET WORD NO.
0405 036714 010337 001204    MOV R3,$REG11    ;GOOD DATA
0406 036720 016137 177776 001206  MOV -2(R1),$REG12 ;BAD DATA
0407 036726 013737 001202 001212  MOV $REG10,$REG14 ;COMPUTE PHYSICAL ADDRESS
0408 036734 005037 001210    CLR $REG13
0409 036740 006137 001212    ROL $REG14
0410 036744 006137 001210    ROL $REG13
0411 036750 063737 005602 001212  ADD PMA,$REG14
0412 036756 005537 001210    ADC $REG13
0413 036762 063737 005604 001210  ADD PMA+2,$REG13
0414 036770 010137 001214    MOV R1,$REG15    ;GET VIRT ADRS FOR PRINTOUT
0415 036774 162737 000002 001214  SUB #2,$REG15
0416 037002 104063          ERROR 63          ;TYPE GOOD AND BAD DATA
0417 037004 032777 000100 142126  BIT #BIT6,$SWR   ;SEE IF JUST 1 ERROR SHOULD BE REPORTED
0418 037012 001022          BNE 21$          ;BR IF JUST 1
0419          ;PROCEED WITH COMPARISONS
0420 037014 005704    16$: TST R4          ;SEE WHICH DATA DESIRED
0421 037016 001001          BNE 18$          ;BR IF NOT INCREASING DATA
0422 037020 005203          INC R3           ;INCREMENT THE DATA WORD
0423 037022 005737 056246    18$: TST $KT11    ;SEE IF MEM MGT
0424 037026 100010          BPL 20$          ;BR IF NOT
0425 037030 032701 020000    BIT #BIT13,R1   ;SEE IF OVERFLOW TO NEXT PAGE
0426 037034 001405          BEQ 20$          ;BR IF NO OVERFLOW
0427 037036 062737 000200 172354  ADD #200,$#KIPAR6 ;INCR. PAR BY 4K FOR NEW PAGE
0428 037044 042701 020000    BIC #BIT13,R1   ;SET PAGE = 6 AGAIN
0429 037050 005302    20$: DEC R2          ;DECR COUNTER
0430 037052 001402          BEQ 21$          ;BR IF DONE COMPARING
0431 037054 000137 036416    JMP 6$           ;CONTINUE COMPARING WORDS
0432 037060 005737 056246    21$: TST $KT11    ;SEE IF MEM MGT PRESENT
0433 037064 100003          BPL 22$          ;BR IF NOT
0434 037066 042737 000001 177572  BIC #BIT0,$#SR0 ;TURN OFF MEM MGT
0435 037074 104410    22$: RESREG      ;RESTORE RD-RS
0436 037076 000207          RTS PC          ;RETURN
0437
0438
0439
0440
0441
0442          ;*****
0443          ;*SETUPM - SUBROUTINE TO INIT PARAMS FOR NPR/MEMORY TESTS.*
0444          ;*****
0445 037100 013765 005646 000002  SETUPM: MOV FC,P.CYLN(R5) ;SET CYL
0446 037106 113765 005506 000004  MOVB FS,P.SECT(R5) ;SET SECTOR
0447 037114 113765 005652 000005  MOVB FT,P.TRCK(R5) ;SET TRACK
0448 037122 023727 005646 000624  CMP FC,#624       ;SEE IF CYL SHOULD BE SCALED DOWN
0449 037130 003403          BLE 2$           ;BR IF NOT
0450 037132 012765 000624 000002  MOV #624,P.CYLN(R5) ;SCALE DOWN THE CYLINDER
0451 037140 012737 064640 005576  2$: MOV #RWBUF,MA    ;GET MEMORY ADDRESS
0452 037146 005737 000170          TST KILLDR      ;SEE IF LOADER CAN BE OVERLAYED
0453 037152 001003          BNE 4$           ;BR IF YES
0454 037154 062737 006000 005576  ADD #6000,MA     ;LEAVE ROOM TO SAVE LOADER
0455 037162 042737 003777 005576  4$: BIC #3777,MA
0456 037170 062737 004000 005576  ADD #4000,MA
0457 037176 005037          CLR MA+2
0458 037202 000207          RTS PC          ;RETURN

```

0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514

037204
037204 004737 035640
037210 105037 003113
037214 004737 041110
037220 010537 037236
037224 012737 037272 000004
037232 004737 051120
037236 000000
037240 004737 045470
037244 105737 003113
037250 001012
037252 010304
037254 001402
037256 005304 10\$:
037260 001376
037262 005737 160000 12\$:
037266 104111
037270 000401
037272 022626 20\$:
037274 000761 22\$:
037276 012737 000006 000004 26\$:
037304 000207
104407
005001 000012
001002
000401
005400 4\$:
000241 6\$:
006100
006101
060037 005576
005537 005600
060137 005600
104410
000207

* REFNEM - THIS SUBROUTINE PERFORMS A SUBSYSTEM COMMAND, WHILE
* REPEATEDLY REFERENCING A NON-EXISTENT MEMORY LOCATION (760000). THIS
* IS DONE FOR THE PURPOSE OF CAUSING UNIBUS CONTENTION DURING DISK
* NPR'S, SINCE THE BUS IS SIEZED FOR 5-20 USEC AT EACH NEM TIME-OUT.
* ALL DRIVER PARAMETERS (INCLUDING THE COMMAND) MUST BE SET IN THE
* PARAMETER BLOCK ON ENTRY, AND R3 MUST CONTAIN THE TIMING CONSTANT
* WHICH REGULATES THE FREQUENCY OF NEM REFERENCES, WHILE WAITING FOR
* COMMAND COMPLETION.

REFNEM:
JSR PC,CTLOUT ;CHECK FOR (↑C) OR (↑Z) KBD INPUT
CLRB DONE ;CLEAR THE DONE FLAG
JSR PC,STRCMD ;STORE PREV AND CURRENT COMMANDS
MOV R5,4\$;SET PARAM BLK ADDRESS
MOV #20\$,Q#ERRVEC ;SET TIME-OUT VECTOR ADDRESS
JSR PC,C.INIT ;CALL DRIVER
4\$: .WORD
6\$: JSR PC,W.WTCH ;CALL WATCH DOG
TSTB DONE ;DONE SET ?
BNE 26\$;BR IF YES
MOV R3,R4 ;GET COPY OF R3
BEQ 12\$;BR IF TIMER = 0
10\$: DEC R4 ;DECREMENT TIMER
BNE 10\$;BR IF NOT DONE TIMING
12\$: TST Q#160000 ;REFERENCE NON-EXISTENT MEMORY
ERROR 111 ;"NO NEM WHEN EXPECTED"
BR 22\$
20\$: CMP (SP)+,(SP)+ ;RESTORE THE STACK
22\$: BR 6\$;KEEP WAITING FOR COMMAND COMPLETION
26\$: MOV #6,Q#ERRVEC ;RESTORE TIME-OUT VECTOR
RTS PC ;RETURN

* INCRMA - INCREMENT MA BY WORD COUNT IN P.WC(R5).

INCRMA: SAVREG ;SAVE R0-R5
CLR R1
MOV P.WC(R5),R0 ;GET WORD COUNT
BNE 4\$;BR IF NOT 65,536(DEC)
INC R1 ;SET HI BIT
BR 6\$;CONTINUE
4\$: NEG R0 ;GET TRUE WORD COUNT
6\$: CLC ;DOUBLE THE WORD COUNT TO GET BYTES
ROL R0
ROL R1
ADD R0,MA ;ADD IT TO COMPUTE NEW MA
ADC MA+2
ADD R1,MA+2
RESREG ;RESTORE R0-R5
RTS PC ;RETURN

```

0515
0516
0517
0518
0519 037354
0520 037354 004737 030776
0521 037360 105065 000007
0522 037364 112765 000121 000001
0523 037372 012765 000632 000002
0524 037400 112765 000002 000005
0525 037406 012703 000012
0526 037412 105737 003115
0527 037416 001403
0528 037420 105265 000004
0529 037424 005203
0530 037426 012765 177400 000012 6$:
0531 037434 012765 004222 000010
0532 037442 012737 037616 003036 8$:
0533 037450 105037 003131
0534 037454 004737 040774
0535 037460 105737 003131
0536 037464 001413
0537 037466 062765 000002 000004
0538 037474 126503 000004
0539 037500 001360
0540 037502 004737 042206 10$:
0541 037506 104120
0542 037510 000137 044476
0543 037514 012765 003222 000010 12$:
0544 037522 110365 000004
0545 037526 012703 000026
0546 037532 105737 003115
0547 037536 001402
0548 037540 012703 000023
0549 037544 012737 037616 003036 14$:
0550 037552 105037 003131
0551 037556 004737 040774
0552 037562 105737 003131
0553 037566 001407
0554 037570 062765 000002 000004
0555 037576 126503 000004
0556 037602 003760
0557 037604 000736
0558 037606 012737 042466 003036 16$:
0559 037614 000207
0560
0561
0562
0563 037616 032765 130600 000034
0564 037624 001002
0565 037626 000137 042466
0566 037632 105237 003131 4$:
0567 037636 000137 044702
0568
0569
0570

```

```

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

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

```

```

8571
8572
8573
8574
8575
8576
8577 037642 104407
8578 037644 005001
8579 037646 005004
8580 037650 012700 004232
8581 037654 104401 011224
8582 037660 104401 011251
8583 037664 104401 062724
8584 037670 104401 001315
8585 037674 005710
8586 037676 100007
8587 037700 005701
8588 037702 001032
8589 037704 104401 011271
8590 037710 104401 001315
8591 037714 000425
8592 037716 012046
8593 037720 104402
8594 037722 104401 013425
8595 037726 011003
8596 037730 105003
8597 037732 000303
8598 037734 010346
8599 037736 104402
8600 037740 104401 013425
8601 037744 111003
8602 037746 010346
8603 037750 104402
8604 037752 104401 001315
8605 037756 005720
8606 037760 005201
8607 037762 020127 000176
8608 037766 002742
8609 037770 005704
8610 037772 001006
8611 037774 005204
8612 037776 012700 003232
8613 040002 104401 011237
8614 040006 000726
8615 040010 104410
8616 040012 000207
8617
8618
8619
8620
8621
8622
8623
8624 040014 104407
8625 040016 016500 000002
8626 040022 116501 000005

```

```

*****
* TYPBSF - TYPE FACTORY AND SOFTWARE BAD SECTOR FILES (BSF'S)
* WITH THE CYLINDER, TRACK, AND SECTOR NO. OF EACH BAD SECTOR
* LISTED SEPARATELY, IN A TABLE OF OCTAL VALUES, FOR EACH OF THE
* 2 BSF'S.
*****
TYPBSF: SAVREG          ;SAVE R0-R5
          CLR           R1          ;INIT LIMIT COUNTER
          CLR           R4          ;INIT BSF FILE INDICATOR
          MOV          #BSFACT+10,R0 ;ADRS OF DATA IN FACTORY BSF
          TYPE         ,FACTBS      ;TYPE "FACTORY"
          TYPE         ,BDSECT      ;TYPE "BAD SECTORS"
3$:      TYPE         ,DH6041      ;TYPE "CYLNR TRACK SECTOR"
          TYPE         ,$CRLF
4$:      TST           (R0)         ;SEE IF A SECTOR LISTED HERE
          BPL          8$           ;BR IF YES
          TST          R1           ;SEE IF FILE IS EMPTY
          BNE         14$          ;BR IF NOT EMPTY
          TYPE         ,NOFALS      ;TYPE "NONE"
          TYPE         , $CRLF      ;TYPE <CR>,<LF>
8$:      BR           14$
          MOV          (R0)+,-(SP)  ;GET A CYL NO.
          TYPOC        ;TYPE IT
          TYPE         ,SPACE2      ;TYPE SEPARATORS
          MOV          (R0),R3      ;GET TRACK AND SECTOR
          CLRB         R3
          SWAB         R3          ;GET THE TRACK NO.
          MOV          R3,-(SP)
          TYPOC        ;TYPE IT
          TYPE         ,SPACE2      ;TYPE SEPARATORS
          MOVB         (R0),R3      ;GET SECTOR NO.
          MOV          R3,-(SP)
          TYPOC        ;TYPE IT
          TYPE         , $CRLF
          TST         (R0)+
          INC          R1          ;INCR THE POINTER
          INC          R1          ;INCR THE COUNTER
          CMP          R1,#126.     ;SEE IF LIMIT REACHED IN THIS FILE
          BLT         4$           ;BR IF NOT YET
14$:     TST          R4           ;SEE IF BOTH FILES TYPED YET
          BNE         18$          ;BR IF NOT YET
          INC          R4          ;SET INDICATOR FOR SOFT. FILE
          MOV          #BSSOFT+10,R0 ;ADRS OF DATA IN SOFTWARE BSF
          TYPE         ,SOFTBS      ;TYPE "SOFTWARE BAD SECTORS:"
          BR          3$           ;GO TYPE SOFT. BSF
18$:     RESREG
          RTS          PC          ;RESTORE R0-R5
          ;RETURN

```

```

*****
* FINMEM - COMPUTE NO. OF WORDS XFERRED ON PARTIAL XFER
* (TERMINATED BY BSE ERROR), AND STORE IT IN WDSXFR.
*****
FINMEM: SAVREG          ;SAVE R0-R5
          MOV          P.CYLN(R5),R0 ;GET ORIG. CYL NO.
          MOVB         P.TRCK(R5),R1 ;GET TRACK NO.

```

```

8627 040026 116502 000004
8628 040032 005003
8629 040034 026500 000030
8630 040040 001006
8631 040042 126501 000027
8632 040046 001003
8633 040050 126502 000026
8634 040054 001404
8635 040056 005203
8636 040060 004737 040100
8637 040064 000763
8638 040066 000303
8639 040070 010337 005562
8640 040074 104410
8641 040076 000207
    
```

```

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

```

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

```

8650 040100 012704 000025
8651 040104 105737 003115
8652 040110 001402
8653 040112 012704 000023
8654 040116 005202
8655 040120 020204
8656 040122 003407
8657 040124 005002
8658 040126 005201
8659 040130 020127 000002
8660 040134 003402
8661 040136 005001
8662 040140 005200
8663 040142 000207
    
```

```

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

```

*****
*MIDXFR - MID-TRANSFER PARAMETER UPDATE SUBROUTINE
*THIS SUBROUTINE UPDATES PMA,PMA+2, MEM MGT REGS, UNIBUS MAP, P.BALO(R5),
*P.BAHI(R5), AND P.WC(R5), IN PREPARATION TO RESUME A TRANSFER
*TERMINATED BY A BAD SECTOR ERROR (BSE).
*****
    
```

```

8673 040144 104407
8674 040146 004737 040014
8675 040152 016500 000030
8676 040156 116501 000027
8677 040162 116502 000026
8678 040166 004737 040100
8679 040172 010065 000002
8680 040176 110165 000005
8681 040202 110265 000004
8682 040206 013703 005562
    
```

```

MIDXFR: SAVREG ;SAVE R0-R5
JSR PC,FINMEM ;COMPUTE NO. OF WORDS XFERRED
MOV P.DCYL(R5),R0 ;GET CYL NO.
MOV B P.DTS+1(R5),R1 ;GET TRACK NO.
MOV B P.DTS(R5),R2 ;GET SECTOR NO.
JSR PC,INCRSC ;INCREMENT PACK ADRS TO NEXT SECTOR
MOV R0,P.CYLN(R5) ;UPDATE CYLINDER
MOV B R1,P.TRCK(R5) ;UPDATE TRACK
MOV B R2,P.SECT(R5) ;UPDATE SECTOR
MOV WDSXFR,R3 ;GET NO. OF WORDS XFERRED
    
```

```

8683 040212 062703 000400 ADD #400,R3 ;SKIP BAD SECTOR
8684 040216 060365 000012 ADD R3,P.WC(R5) ;UPDATE P.WC(R5)
8685 040222 005004 CLR R4 ;GET BYTES XFERRED
8686 040224 006103 ROL R3
8687 040226 006104 ROL R4
8688 040230 060337 005602 ADD R3,PMA ;UPDATE PMA,PMA+2
8689 040234 005537 005604 ADC PMA+2
8690 040240 060437 005604 ADD R4,PMA+2
8691 040244 013765 005602 000010 MOV PMA,P.BALO(R5) ;UPDATE P.BALO(R5)
8692 040252 013700 005604 MOV PMA+2,R0
8693 040256 042700 177774 BIC #177774,R0
8694 040262 150065 000007 BISB R0,P.BAHI(R5) ;UPDATE P.BAHI(R5)
8695 040266 005737 056246 TST $K11 ;SEE IF MEM MGT
8696 040272 100002 BPL 16$
8697 040274 004737 030334 JSR PC,PREPAR ;UPDATE MEM MGT AND UNIBUS MAP
8698 040300 104410 16$: RESREG ;RESTORE R0-R5
8699 040302 000207 RTS PC ;RETURN

```

```

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

```

```

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

```

```

8736 040416 010446
8737 040420 005004
8738 040422 105037 003131

```

```

*****
*TRANSFR - PERFORM A DATA TRANSFER, AND HANDLE BAD SECTORS, IF
*ENCOUNTERED. THE COMMAND MUST BE SET IN THE PARAM BLK. THIS SUB-
*ROUTINE IS SUBSTITUTED FOR DRVCAL WHEN BAD SECTORS MUST BE HANDLED.
*****
TRANSFR: MOV R4,-(SP) ;SAVE R4 ON STACK
CLR R4 ;CLEAR BSE ERROR INDICATOR
CLRB WCEFLG ;INIT WCE ERROR FLAG TO 0

```

```

8739 040426 004737 040774 4$: JSR PC,DRVCAL ;PERFORM THE S.S. FUNCTION
8740 040432 032737 000100 005474 BIT #WCERR,RECODE ;SEE IF WCE ERROR OCCURRED
8741 040440 001403 BEQ 5$ ;BR IF NOT
8742 040442 112737 000001 003131 MOV#B #1,WCEFLG ;SET WCE ERROR FLAG
8743 040450 032737 000002 005474 5$: BIT #BSERR,RECODE ;SEE IF BSE ERROR OCCURRED
8744 040456 001406 BEQ 6$ ;BR IF NOT
8745 040460 005204 INC R4 ;INCR BSE ERROR INDICATOR
8746 040462 004737 040144 JSR PC,MIDXFR ;UPDATE PARAMS TO RESUME XFER
8747 040466 005765 000012 TST P.WC(R5) ;SEE IF ENTIRE XFER IS COMPLETED
8748 040472 001355 BNE 4$ ;BR IF NOT
8749 040474 005704 6$: TST R4 ;SEE IF ANY BSE ERRORS OCCURRED
8750 040476 001411 BEQ 8$ ;BR IF NOT
8751 040500 004737 040366 JSR PC,GTPRMS ;RESTORE ORIGINAL PARAMS OF XFER
8752 040504 004737 040304 JSR PC,SVPRMS ;RESTORE WDSXFR,PMA,PMA+2
8753 040510 005737 056246 TST $K11 ;SEE IF MEM MGT PRESENT
8754 040514 100002 BPL 8$ ;BR IF NOT
8755 040516 004737 030334 JSR PC,PREPAR ;PREPARE MEM MGT AND U.M.
8756 040522 012604 8$: MOV (SP)+,R4 ;RESTORE R4
8757 040524 000207 RTS PC ;RETURN

```

```

8758
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768
8769
8770
8771

```

```

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

```

```

8772 040526 010237 001266 SRHTBS: MOV R2,$TMP2
8773 040532 010337 001270 MOV R3,$TMP3
8774 040536 012637 001272 MOV (SP)+,$TMP4 ;STORE RETURN CONTENTS OF R4
8775 040542 011402 MOV (R4),R2 ;GET ADDRESS OF BAD SECTOR TABLE TO SEARCH
8776 040544 012437 001264 MOV (R4)+,$TMP1 ;SETUP FOR LENGTH OF BAD SECTOR TABLE
8777 040550 062737 001000 001264 ADD #1000,$TMP1
8778 040556 005003 CLR R3 ;CLEAR R3 FOR COUNT
8779 040560 062702 000010 ADD #10,R2 ;SET R2 FOR POINTER TO CYLINDER ENTRY
8780 040564 005712 1$: TST (R2) ;TEST IF ALL ONES
8781 040566 100430 BMI 5$ ;YES-DONE
8782 040570 020012 CMP R0,(R2) ;TEST IF BAD SECTOR IN PRESENT CYL
8783 040572 001406 BEQ 3$ ;YES-GO CHECK TRACK
8784 040574 062702 000004 ADD #4,R2 ;ELSE BUMP POINTER
8785 040600 020237 001264 CMP R2,$TMP1 ;TEST IF OUT OF TABLE
8786 040604 002021 BGE 5$ ;YES-EXIT
8787 040606 000766 BR 1$ ;LOOP
8788 040610 005722 3$: TST (R2)+ ;TRACK CHECK - PUT POINTER AT TRK/SEC BYTE
8789 040612 011237 001302 MOV (R2),$TMP10 ;GET TRK/SEC WORD
8790 040616 042737 174377 001302 BIC #174377,$TMP10 ;CLEAR ALL BUT TRACK
8791 040624 123701 001303 CMPB $TMP10+1,R1 ;CHECK IF BAD SECTOR IN THIS TRACK
8792 040630 001402 BEQ 4$ ;YES - GO PUT SECTOR NUMBER ON STACK
8793 040632 005722 TST (R2)+ ;ELSE BUMP POINTER TO NEXT CYL WORD
8794 040634 000753 BR 1$ ;GO TEST NEXT CYL

```


SEARCH BAD SECTOR TABLES ROUTINE

8795	040636	005203		
8796	040640	012246		
8797	040642	042716	177700	
8798	040646	000746		
8799	040650	010346		
8800	040652	013702	001266	
8801	040656	013703	001270	
8802	040662	013746	001272	
8803	040666	000204		

```

4$: INC R3 ;BUMP BAD SECTOR COUNT
    MOV (R2)+, -(SP) ;PUT TRK/SEC WORD ON STACK
    BIC #177700, (SP) ;CLEAR ALL BUT SECTOR NUMBER
    BR 1$ ;GO CHECK REST OF FILE
5$: MOV R3, -(SP) ;EXIT - PUT NUMBER OF BAD
    MOV $TMP2, R2 ;SECTORS ON STACK-RESTORE
    MOV $TMP3, R3 ;REGISTERS
    MOV $TMP4, -(SP) ;PUT RETURN ON STACK
    RTS R4 ;RETURN

```

```

.SBTTL BAD SECTOR CHECK ROUTINE
;THIS ROUTINE WILL SEARCH BOTH TABLES TO DETERMINE IF A
;SPECIFIC SECTOR IS LISTED AS BAD. IF THE SECTOR IS LISTED IN THE
;TABLES THE ROUTINE SETS THE "BADSEC" FLAG AND RETURNS. IF THE SECTOR
;IS NOT LISTED THE FLAG IS RESET.

```

8811	040670	104407		
8812	040672	012737	004222	040704
8813	040700	004437	040526	
8814	040704	000000		
8815	040706	012603		
8816	040710	001015		
8817	040712	023727	040704	003222
8818	040720	001404		
8819	040722	012737	003222	040704
8820	040730	000763		
8821	040732	042737	001000	005474
8822	040740	104410		
8823	040742	000207		
8824	040744	022602		
8825	040746	001403		
8826	040750	005303		
8827	040752	001374		
8828	040754	000756		
8829	040756	052737	001000	005474
8830	040764	005303		
8831	040766	001764		
8832	040770	005726		
8833	040772	000774		

```

BDSRCK: SAVREG
    MOV #BSFACT, 1$ ;SET TABLE TO SEARCH
    JSR R4, SRHTBS ;GO SEARCH IT
1$: .WORD ;TABLE ADDRESS GOES HERE
    MOV (SP)+, R3 ;GET NUMBER OF BAD SECTORS, IF ANY
    BNE 6$ ;IF ANY, GO TEST WHICH ONES
7$: CMP 1$, #BSSOFT ;ELSE TEST IF OTHER TABLE ALREADY SEARCHED
    BEQ 3$ ;IF YES-EXIT
    MOV #BSSOFT, 1$ ;SET OTHER TABLE FOR SEARCH
    BR 2$ ;GO SEARCH IT
3$: BIC #BADSEC, RECODE ;CLEAR BAD SECTOR BIT
4$: RESREG
    RTS PC ;RETURN
6$: CMP (SP)+, R2 ;THIS SECTOR IN TABLE?
    BEQ 8$ ;YES-GO SET BIT & EXIT
    DEC R3 ;DECREMENT BAD SECTOR COUNT
    BNE 6$ ;IF NOT ZERO-CHECK NEXT ENTRY
    BR 7$ ;ELSE GO SEARCH OTHER TABLE
8$: BIS #BADSEC, RECODE ;SET BAD SECTOR BIT
9$: DEC R3 ;CLEAR STACK OF OTHER BAD SECTOR
    BEQ 4$ ;NUMBER
    TST (SP)+
    BR 9$ ;EXIT

```

```

.SBTTL CALL DRIVER ROUTINE
;ENTRY JSR PC, DRVCAL
;* WITH R5 POINTING TO PARAMETER BLOCK
;*RETURN RTS PC
;*
;THIS ROUTINE IS USED TO INITIATE A SUBSYSTEM OPERATION BY
;CALLING THE DRIVER. THE PARAMETER BLOCK MUST BE SET UP
;BY THE CALLER AND R5 MUST POINT TO THE PARAMETER
;BLOCK WHEN THE ROUTINE IS CALLED.

```

```

;THIS ROUTINE WAITS FOR THE OPERATION TO BE COMPLETED.
;WHILE WAITING THE WATCHDOG TIMER IS CALLED TO PREVENT
;SILENT DEATH IN CASE THE SUBSYSTEM DOES NOT PROVIDE AN

```

8834
8835
8836
8837
8838
8839
8840
8841
8842
8843
8844
8845
8846
8847
8848
8849
8850

CALL DRIVER ROUTINE

```

8851
8852
8853
8854 040774
8855 040774 004737 034040
8856 041000 004737 035640
8857 041004 105037 003113
8858 041010 105037 003127
8859 041014 004737 041110
8860 041020 010537 041030
8861 041024 004737 051120
8862 041030 000000
8863 041032 004737 045470
8864 041036 105737 003113
8865 041042 001773
8866 041044 105737 003127
8867 041050 001416
8868 041052 105037 003113
8869 041056 105037 003127
8870 041062 010537 041072
8871 041066 004737 051120
8872 041072 000000
8873 041074 004737 045470
8874 041100 105737 003113
8875 041104 001773
8876 041106 000207
8877
8878
8879
8880
8881
8882
8883 041110 104407
8884 041112 012701 005222
8885 041116 012700 005236
8886
8887 041122 012021
8888 041124 012021
8889 041126 012021
8890 041130 012021
8891 041132 012021
8892 041134 012021
8893 041136 105737 003134
8894 041142 001414
8895 041144 013737 005260 005254
8896 041152 013737 005256 005252
8897 041160 013737 170202 005260
8898 041166 013737 170200 005256
8899
8900 041174 012701 005236
8901 041200 012521
8902 041202 012521
8903 041204 012521
8904 041206 012521
8905 041210 012521
8906 041212 012521

```

```

;*INTERRUPT. THE TERMINATION HANDLER ROUTINES WILL SET THE DONE
;*FLAG WHICH KEYS THE ROUTINE TO RETURN TO THE CALLER.
;*****
DRVCAL:
      JSR      PC,STALL          ;PERFORM A STALL IF REQUIRED
      JSR      PC,CTLOUT        ;CHECK FOR (↑C) OR (↑Z) KBD INPUT
      CLRB     DONE             ;CLEAR DONE FLAG
      CLRB     DRNAFG           ;CLEAR DRIVE SIEZED BY OTHER PORT FLAG
      JSR      PC,STRCMD        ;STORE PREV AND CURRENT COMMANDS
      MOV      R5,4$           ;GET PARAM BLOCK ADDRESS
      JSR      PC,C.INIT       ;CALL DRIVER
      4$:      .WORD            ;P.B. ADDRESS GOES HERE
      6$:      JSR      PC,W.WTCH ;CALL WATCH DOG
      TSTB     DONE            ;DONE SET?
      BEQ      6$              ;NO-LOOP
      TSTB     DRNAFG          ;SEE IF DRIVE WAS SIEZED BY OTHER PORT
      BEQ      12$             ;BR IF NOT
      CLRB     DONE            ;CLEAR DONE FLAG
      CLRB     DRNAFG          ;CLEAR DRIVE NOT AVAIL FLAG
      MOV      R5,8$           ;GET PARAMETER BLOCK ADDRESS
      JSR      PC,C.INIT       ;RE-ISSUE THE COMMAND
      8$:      .WORD            ;P.B. ADDRESS GOES HERE
      10$:     JSR      PC,W.WTCH ;CALL WATCH DOG
      TSTB     DONE            ;DONE SET ?
      BEQ      10$             ;NO - LOOP
      12$:     RTS      PC      ;YES-RETURN

;*****
;*STRCMD - STORE PREVIOUS AND CURRENT SUBSYSTEM COMMANDS
;*****
STRCMD: SAVREG          ;SAVE R0-R5
      MOV      #PRVCMD,R1    ;ADDR OF PREV COMMAND STORAGE
      MOV      #COMSTR,R0    ;ADDR OF CURRENT COMMAND STORAGE
;STORE PREVIOUS COMMAND
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      MOV      (R0)+,(R1)+
      TSTB     UBMPRS        ;SEE IF UNIBUS MAP PRESENT
      BEQ      4$            ;BR IF NOT
      MOV      CRMPHO,PRMPHO ;STORE PREV U.B. MAP REG 0
      MOV      CRMPLO,PRMPLO
      MOV      @#MAPH00,CRMPHO ;STORE CURRENT U.B. MAP REG 0
      MOV      @#MAPL00,CRMPLO
;STORE CURRENT COMMAND
      4$:      MOV      #COMSTR,R1
      MOV      (R5)+,(R1)+   ;R5 POINTS TO DRIVER PARAM BLK
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+
      MOV      (R5)+,(R1)+

```

CALL DRIVER ROUTINE

8907 041214 104410
8908 041216 000207
8909
8910
8911
8912
8913
8914
8915
8916
8917
8918
8919 041220 152737 000377 003113
8920 041226 032737 100000 005474
8921 041234 001403
8922 041236 105037 003116
8923 041242 000413
8924 041244 105737 003116
8925 041250 001410
8926 041252 005037 001174
8927 041256 113737 003116 001174
8928 041264 104101
8929 041266 105037 003116
8930 041272 005037 005474
8931 041276 000207
8932
8933
8934
8935
8936
8937
8938
8939
8940
8941
8942 041300 104407
8943 041302 105237 003120
8944 041306 032737 000040 005670
8945
8946 041314 001412
8947 041316 123727 003120 000024
8948 041324 002406
8949 041326 105037 003120
8950 041332 104401 011103
8951 041336 000137 017470
8952 041342 032777 020000 137570 9\$:
8953 041350 001402
8954 041352 000137 041756
8955 041356 005000 6\$:
8956 041360 005005
8957 041362 005105
8958 041364 113700 001114
8959 041370 005300
8960 041372 006300
8961 041374 006300
8962 041376 006300

RESREG ;RESTORE R0-R5
RTS PC ;RETURN

.SBTTL DRIVE ERROR FREE RETURN ROUTINE
*THIS ROUTINE IS CALLED BY THE DRIVER WHEN NO ERROR
*HAS BEEN DETECTED IN THE OPERATION. THE ROUTINE SETS THE
*DONE FLAG THAT IS TESTED IN THE DRIVER CALLING
*ROUTINE.

ERRFRE: BISB #377,DONE ;SET THE DONE FLAG
BIT #ANYDER,RECODE ;TEST IF ANY DATA ERROR
BEQ 2\$;IF NO - DO ERROR RECOVERY PRINT TEST
CLRB ERRCNT ;CLEAR ERROR COUNT
BR 1\$;EXIT
2\$: TSTB ERRCNT ;CHECK IF ANY ERRORS HAVE OCCURRED
BEQ 1\$;NO - SKIP TO EXIT
CLR \$REG5
MOVB ERRCNT,\$REG5 ;GET RETRY COUNT
ERROR 101 ;PRINT RETRY SUCCESSFUL MESSAGE
CLRB ERRCNT ;CLEAR ERROR COUNT
1\$: CLR RECODE ;CLEAR RECOVERY FLAGS
RTS PC ;RETURN

.SBTTL TYPE ERROR ROUTINE
*ENTRY - JSR PC,TYPERR
*RETURN - RTS PC
*
*THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN USES THE "ERROR TABLE" (\$ERRTB)
*ENTRY TO DEFINE WHAT INFORMATION IS TO BE REPORTED CONCERNING
*THE ERROR.

TYPERR: SAVREG
INCB DRIVERS ;INCR ERROR COUNT FOR THIS DRIVE
BIT #BITS,CS ;SEE IF DRIVE SHOULD BE DROPPED
; IF ERROR LIMIT EXCEEDED
BEQ 9\$;BR IF NOT
CMPB DRIVERS,#20 ;SEE IF 20(DEC) ERRORS EXCEEDED
BLT 9\$;BR IF NOT
CLRB DRIVERS ;CLEAR DRIVE ERROR COUNT
TYPE ,DROPDR ;TYPE "DROPPING DRIVE"
JMP NEWDRV ;PROCEED TO TEST NEXT DRIVE
9\$: BIT #SW13,DSWR ;INHIBIT ERROR TYPEOUTS?
BEQ 6\$;BR IF NO
6\$: JMP 20\$
CLR R0 ;CLR R0 FOR ERROR NUMBER
CLR R5 ;INIT INDENT INDICATOR
COM R5
MOVB \$ITEMB,R0 ;ENTER ERROR NUMBER
DEC R0 ;FORM INDEX FOR ERROR TABLE
ASL R0
ASL R0

TYPE ERROR ROUTINE

89.963	041400	062700	001400	1\$:	ADD	;\$ERRTB, R0	:FORM ADDRESS OF ERROR ENTRY
89.964	041404	012037	041424		MOV	(R0)+, 2\$:GET EM POINTER
89.965	041410	001406			BEQ	3\$:BRANCH IF THERE ISN'T ONE
89.966	041412	104401	013411		TYPE	, CR2LF	
89.967	041416	104401	056704		TYPE	, AS2SP2	:TYPE "*** "
89.968	041422	104401			TYPE		:TYPE ERROR MESSAGE (EM)
89.969	041424	000000		2\$:	.WORD	0	:EM POINTER GOES HERE
89.970	041426	012037	041602	3\$:	MOV	(R0)+, 4\$:GET DH POINTER
89.971	041428	001467			BEQ	5\$:BR IF THERE ISN'T ONE
89.972	041434	104401	001315		TYPE	, \$CR2LF	
89.973	041440	104401	056672		TYPE	, \$TSTMSG	:TYPE " TEST "
89.974	041440	013746	001102		MOV	;\$TSTNM, -(SP)	:GET TEST NO. ON STACK
89.975	041450	104403			TYPOS		:TYPE IT
89.976	041450	002			.BYTE	2	:2 DIGITS
89.977	041450	000			.BYTE	0	:SUPPRESS LEADING ZEROS
89.978	041454	104401	013411		TYPE	, CR2LF	
89.979	041460	032777	010000	137452	BIT	;\$BIT12, \$SWR	:REPORT DESCRIPTION ONLY ?
89.980	041466	001133			BNE	20\$:BR IF YES
89.981	041470	104401	062026		TYPE	, DH105	:TYPE "PREVIOUS COMMAND :"
89.982	041474	104401	001315		TYPE	, \$CR2LF	
89.983	041500	104401	062220		TYPE	, DH101+10	:TYPE PREV COMMAND HEADER
89.984	041504	104401	001315		TYPE	, \$CR2LF	
89.985	041510	012701	000006		MOV	;\$R1	:SIX COMMAND VALUES
89.986	041514	012702	001236		MOV	;\$REG26, R2	:STARTING ADDR OF PREV CMND VALUES
89.987	041520	012246		30\$:	MOV	(R2)+, -(SP)	:PUT A WORD ON STACK
89.988	041522	104402			TYPOC		:TYPE IT
89.989	041524	104401	013425		TYPE	, SPACE2	:TYPE SEPARATORS
89.990	041530	005301			DEC	R1	:SEE IF 7 VALUES TYPED YET
89.991	041532	001372			BNE	30\$:BR IF NOT
89.992	041534	104401	001315		TYPE	, \$CR2LF	
89.993	041540	104401	013425		TYPE	, SPACE2	:INDENT
89.994	041544	104401	062277		TYPE	, DH102	:TYPE HDR FOR BA DATA
89.995	041550	104401	001315		TYPE	, \$CR2LF	
89.996	041554	104401	013425		TYPE	, SPACE2	:INDENT
89.997	041560	012246			MOV	(R2)+, -(SP)	:TYPE PREV. HI BA BITS
89.998	041562	104402			TYPOC		
89.999	041564	104401	013425		TYPE	, SPACE2	
9000	041570	011246			MOV	(R2), -(SP)	:TYPE PREV. LO BA BITS
9001	041572	104402			TYPOC		
9002	041574	104401	013411		TYPE	, CR2LF	
9003	041600	104401			TYPE		:TYPE DATA HEADER
9004	041602	000000		4\$:	.WORD	0	:DH POINTER GOES HERE
9005	041604	104401	001315		TYPE	, \$CR2LF	
9006	041610	005005			CLR	R5	:INIT INDENT INDICATOR
9007	041612	032777	010000	137320	5\$:	BIT	;\$BIT12, \$SWR
9008	041620	001056			BNE	20\$:BR IF YES
9009	041622	012001			MOV	(R0)+, R1	:GET DT POINTER
9010	041624	001454			BEQ	20\$:BRANCH IF THERE ARE NONE
9011	041626	012000			MOV	(R0)+, R0	:GET DF POINTER
9012	041630	012002			MOV	(R0)+, R2	:STORE NUMBER OF DH'S
9013	041632	112003		10\$:	MOVB	(R0)+, R3	:GET & STORE NUMBER OF DATA WORDS
9014	041634	105720			TSTB	(R0)+	:BUMP PAST FORMAT WORD
9015	041636	005703			TST	R3	:TEST IF ANY DATA FOR THIS HEADER
9016	041640	001417			BEQ	14\$:NO - SKIP DATA PRINT
9017	041642	005705			TST	R5	:SEE IF SHOULD INDENT
9018	041644	001002			BNE	11\$:BR IF NOT

DZRN.C.P11 05-OCT-76 10:07

TYPE ERROR ROUTINE

```

9019 041646 104401 013425
9020 041652 013146
9021 041654 104402
9022 041656 005303
9023 041660 001403
9024 041662 104401 013425
9025 041666 000771
9026 041670 005702
9027 041672 001402
9028 041674 104401 001315
9029 041700 005302
9030 041702 003425
9031 041704 012037 041736
9032 041710 105710
9033 041712 001004
9034 041714 104401 001315
9035 041720 005005
9036 041722 000404
9037 041724 005105
9038 041726 001002
9039 041730 104401 013425
9040 041734 104401
9041 041736 000000
9042 041740 104401 001315
9043 041744 105710
9044 041746 001331
9045 041750 062700 000002
9046 041754 000751
9047 041756 104410
9048 041760 000207
9049
9050
9051
9052
9053
9054
9055
9056
9057
9058
9059
9060 041762 104407
9061 041764 152737 000377 003113
9062 041772 105237 003116
9063 041776 004737 044712
9064 042002 032737 000001 003042
9065 042010 001402
9066 042012 104064
9067 042014 000470
9068 042016 032737 000002 003042
9069 042024 001402
9070 042026 104065
9071 042030 000462
9072 042032 032737 000004 003042
9073 042040 001407
9074 042042 105737 003127

```

```

11$: TYPE SPACE2 :INDENT
MOV 3(R1)+,-(SP) :PUT FIRST DATA WORD ON STACK
TYPOC :TYPE IT
DEC R3 :MORE DATA WORDS
BEQ 12$ :NO-BRANCH
TYPE ,SPACE2 :TYPE SEPARATORS
BR 11$ :LOOP
TST R2 :SEE IF <CR>,<LF> NEEDED
BEQ 14$ :BR IF NOT
TYPE ,SCLF :TYPE IT
DEC R2 :MORE DH'S?
BLE 20$ :NO-BRANCH
MOV (R0)+,16$ :GET NEXT DH POINTER
TSTB (R0) :SEE IF ANY DATA FOR HDR
BNE 34$ :BR IF YES
TYPE ,SCLF :SKIP EXTRA LINE
CLR R5 :RE-INIT INDENT INDICATOR
BR 36$
34$: COM R5 :COMPLEMENT INDENT INDICATOR
BNE 36$ :BR IF NO INDENT REQUIRED
TYPE ,SPACE2 :INDENT
TYPE DH :TYPE DH
16$: WORD 0 :DH POINTER GOES HERE
TSTB (R0) :TYPE A DT?
BNE 10$ :YES-BRANCH
ADD #2,R0 :INCREMENT DF POINTER
BR 14$ :SEE IF END OF DF BLOCK
20$: RESREG
RTS PC

```

```

*****
:SBTTL CONTROLLER ERROR REPORTER ROUTINE
:ENTRY: JSR PC, CONERR
:RETURN: RTS PC
:
:*THIS ROUTINE DECODES THE CONTROLLER ERROR WORD AND
:*REPORTS THE APPROPRIATE MESSAGE. THE RK611 REGISTERS ARE
:*RETRIEVED FROM THE RK611 AND PLACED IN THE PARAMETER
:*BLOCK. THIS IS DONE BECAUSE PARM 0 MAY NOT BE VALID
:*AT THIS TIME.
*****
CONERR: SAVREG :SAVE R0-R5
BISB #377,DONE :SET DONE FLAG
INCB ERRCNT :INCREMENT ERROR COUNT
JSR PC, TOPROC :LOAD RK REGS INTO $REGS
BIT #BIT0,E.CONT :ERROR 0?
BEQ 1$ :NO-BRANCH
ERROR 64 :CLEAR CONT DID NOT CLEAR ERROR
BR 7$
1$: BIT #BIT1,E.CONT :ERROR 1?
BEQ 2$ :NO-BRANCH
ERROR 65 :NO ATTENTION IN ATTENTION SUM REG
BR 7$
2$: BIT #BIT2,E.CONT :ERROR 2?
BEQ 3$ :NO-BRANCH
TSTB DRNAFG :SEE IF DRIVE WAS SIEZED BY OTHER PORT

```

CONTROLLER ERROR REPORTER ROUTINE

```

9075 042046 001402 BEQ 15$ :BR IF NOT
9076 042050 105237 003130 INCB REISSU :SET FLAG TO RE-ISSUE COMMAND
9077 042054 104066 15$: ERROR 66 :UNSOLICITED ATTENTION
9078 042056 000447 BR 7$
9079 042060 032737 000010 003042 3$: BIT #BIT3,E.CONT :ERROR 3?
9080 042066 001402 BEQ 4$ :NO-BRANCH
9081 042070 104067 ERROR 67 :UNEXPECTED DATA TYPE ERROR
9082 042072 000441 BR 7$
9083 042074 032737 000020 003042 4$: BIT #BIT4,E.CONT :ERROR 4?
9084 042102 001402 BEQ 5$ :NO-BRANCH
9085 042104 104070 ERROR 70 :ATTENTION DID NOT RESET WITH CLEAR
9086 042106 000433 BR 7$
9087 042110 032737 000040 003042 5$: BIT #BIT5,E.CONT :ERROR 5?
9088 042116 001402 BEQ 6$ :NO-BRANCH
9089 042120 104071 ERROR 71 :SUBSYS CLEAR DIDN'T CLEAR DRIVE ATTENTION
9090 042122 000425 BR 7$
9091 042124 032737 000400 003042 6$: BIT #BIT8,E.CONT
9092 042132 001401 BEQ 8$
9093 042134 104072 ERROR 72 :DATA LATE WHEN UNLOADING HEADER
9094 042136 032737 001000 003042 8$: BIT #BIT9,E.CONT
9095 042144 001401 BEQ 9$
9096 042146 104073 ERROR 73 :CONTROLLER ERROR DURING DRIVER SERVICE
9097 042150 032737 002000 003042 9$: BIT #BIT10,E.CONT
9098 042156 001401 BEQ 10$
9099 042160 104074 ERROR 74 :DRIVE DETECTED PARITY ERROR
9100 042162 032737 100000 003042 10$: BIT #BIT15,E.CONT
9101 042170 001401 BEQ 11$
9102 042172 104052 ERROR 52 :MULTIPLE DRIVE SELECT
9103 042174 104075 ERROR 75 :UNDEFINED ERROR
9104 042176 005037 003042 7$: CLR E.CONT :CLEAR CONTROLLER ERROR WORD
9105 042202 000137 044512 JMP BGNRTY :GO DO RETRY
9106 *****
9107 :SBTTL REPORT SUPPORT ROUTINE
9108 :*THIS ROUTINE LOADS ALL THE PARAMETER BLOCK DATA TO BE REPORTED
9109 :*INTO THE PROPER TEMPORARY REGISTERS FOR REPORTING. ALL THESE MAY
9110 :*NOT BE INCLUDED IN THE REPORT, BUT THEY ARE LOADED ANYWAY.
9111 REPSUP:
9112 042206 104407 SAVREG
9113 042210 005037 005476 CLR ERRCOM :CLEAR ERROR COMMAND STORE
9114 042214 116537 000001 005476 MOVB P.CMND(R5),ERRCOM :STORE COMMAND START VALUES
9115 042222 012700 001162 MOV #SREG0,R0 :FOR REPORTING
9116 042226 016520 000002 MOV P.CYLN(R5),(R0)+
9117 042232 116520 000005 MOVB P.TRCK(R5),(R0)+
9118 042236 105020 CLRB (R0)+
9119 042240 116520 000004 MOVB P.SECT(R5),(R0)+
9120 042244 105020 CLRB (R0)+
9121 042246 016520 000012 MOV P.WC(R5),(R0)+
9122 042252 012700 001174 MOV #SREG5,R0
9123 042256 116503 000007 MOVB P.BAHI(R5),R3
9124 042262 042703 177774 BIC #177774,R3
9125 042266 010337 001256 MOV R3,$REG36 :HI BA BITS
9126 042272 016537 000010 001260 MOV P.BALO(R5),$REG37 :LO BA BITS
9127 042300 016520 000016 MOV P.CS1(R5),(R0)+ :GET ALL THE VALUES FROM THE
9128 042304 016520 000020 MOV P.CS2(R5),(R0)+ :PARAMETER BLOCK AND LOAD
9129 042310 016520 000030 MOV P.DCYL(R5),(R0)+ :THE TEMPORARY REGISTERS
9130 042314 016520 000026 MOV P.DTS(R5),(R0)+ :FOR REPORTING. ALL THIS

```

```

9131 042320 016520 000022
9132
9133 042324 016520 000024
9134 042330 016520 000032
9135 042334 016520 000036
9136 042340 016520 000034
9137 042344 016520 000040
9138 042350 016520 000042
9139 042354 016520 000044
9140 042360 016520 000046
9141 042364 016520 000050
9142 042370 016520 000052
9143 042374 016520 000054
9144 042400 016520 000056
9145
9146 042404 012701 001236
9147 042410 012700 005222
9148 042414 112021
9149 042416 105021
9150 042420 112021
9151 042422 105021
9152 042424 012021
9153 042426 116021 000001
9154 042432 105021
9155 042434 111021
9156 042436 105021
9157 042440 016021 000006
9158 042444 116021 000003
9159 042450 042703 177774
9160 042454 010321
9161 042456 016011 000004
9162 042462 104410
9163 042464 000207
9164
9165
9166
9167
9168
9169
9170
9171
9172
9173
9174 042466 104407
9175 042470 152737 000377 003113
9176 042476 105237 003116
9177 042502 005037 005474
9178 042506 032737 000400 005474
9179 042514 001402
9180 042516 012705 002704
9181 042522 012737 044664 003036
9182 042530 012737 044702 003034
9183 042536 004737 042206
9184
9185
9186

```

```

MOV P.WCR(R5),(R0)+ ;DATA MAY NOT BE VALID
;FOR ALL REPORTS (TO BE
;DETERMINED LATER) BUT IT IS
;STORED ANY WAY.
MOV P.BAR(R5),(R0)+
MOV P.ASOF(R5),(R0)+
MOV P.DS(R5),(R0)+
MOV P.ER(R5),(R0)+
MOV P.ADD(R5),(R0)+
MOV P.BOO(R5),(R0)+
MOV P.AO1(R5),(R0)+
MOV P.BO1(R5),(R0)+
MOV P.A10(R5),(R0)+
MOV P.B10(R5),(R0)+
MOV P.A11(R5),(R0)+
MOV P.B11(R5),(R0)+
;STORE PREVIOUS COMMAND FOR PRINTOUT
MOV #SREG26,R1 ;ADRS OF PRINT BUF AREA
MOV #PRVCMO,R0 ;ADRS OF PREV CMND STORAGE
MOVSB (R0)+,(R1)+ ;DRIVE NO.
CLRB (R1)+
MOVSB (R0)+,(R1)+ ;COMMAND
CLRB (R1)+
MOV (R0)+,(R1)+ ;CYL ADDRESS
MOVSB 1(R0),(R1)+ ;TRACK
CLRB (R1)+
MOVSB (R0),(R1)+ ;SECTOR
CLRB (R1)+
MOV 6(R0),(R1)+ ;WORD COUNT
MOVSB 3(R0),R3 ;HI BA BITS
BIC #177774,R3
MOV R3,(R1)+
MOV 4(R0),(R1) ;LO BA BITS
RESREG
RTS PC
;*****
;SBTTL REPORT ERROR ROUTINE
;* ENTRY JSR PC,ERRHDL
;*RETURN RTS PC
;*
;THIS ROUTINE IS CALLED BY THE DRIVER WHEN AN ERROR IS DETECTED
;IN THE OPERATION. THE ROUTINE DETERMINES WHICH COMMAND WAS
;BEING EXECUTED AND GENERATES THE PROPER ERROR MESSAGE.
;*****
ERRHDL: SAVREG
BISB #377,DONE ;SET DONE FLAG
INCB ERRCNT ;INCREMENT ERROR COUNT
CLR RECODE ;CLEAR RECOVERY CODE WORD
ER2ENT: BIT #LEVZER,RECODE ;TEST IF 2ND LEVEL ERROR
BEQ 52$ ;NO - SKIP PARAM BLOCK CHANGE
MOV #PARM1,R5 ;ELSE SET R5 TO PARAMETER BLOCK 1
52$: MOV #RETANL,A.ABNL ;SET NOW ABNORMAL AND NORMAL RETURN FOR
MOV #RETNML,A.NORM ;DRIVER OPERATIONS IN ERROR PROCESSING
JSR PC,REPSUP ;GO SET UP REGISTERS FOR REPORT
;NOW BEGIN TESTING THE ERROR
;BITS. THE SEQUENCE IN
;WHICH THEY ARE TESTED IS

```

F14

```

9187
9188
9189
9190
9191
9192
9193
9194
9195
9196
9197
9198
9199
9200
9201 042542 016504 000034      MOV      P.ER(R5),R4
9202 042546 032765 020000 000020    BIT      #UPE,P.CS2(R5)
9203 042554 001406      BEQ      1$
9204 042556 104001      ERROR   1
9205 042560 052737 000200 005474    BIS      #ABORT,RECODE
9206 042566 000137 044236      JMP      37$
9207 042572 032765 004000 000020 1$:     BIT      #NEM,P.CS2(R5) ;TEST NON-EXISTANT MEMORY
9208 042600 001406      BEQ      2$
9209 042602 104002      ERROR   2
9210 042604 052737 000200 005474    BIS      #ABORT,RECODE
9211 042612 000137 044236      JMP      37$
9212 042616 032765 010000 000020 2$:     BIT      #NED,P.CS2(R5) ;TEST NON-EXISTANT DRIVE
9213 042624 001412      BEQ      3$
9214 042626 032765 000400 000020    BIT      #UFE,P.CS2(R5) ;TEST IF NED & UFE BOTH SET
9215 042634 001403      BEQ      39$
9216 042636 104035      ERROR   35 ;NED & UFE BOTH SET ERROR
9217 042640 000137 044236      JMP      37$
9218 042644 104003      ERROR   3 ;NED ONLY
9219 042646 000137 044236      JMP      37$
9220 042652 032765 000400 000020 3$:     BIT      #UFE,P.CS2(R5) ;TEST UNIT FIELD ERROR
9221 042660 001412      BEQ      4$
9222 042662 032765 010000 000020    BIT      #NED,P.CS2(R5) ;TEST IF UFE & NED BOTH SET
9223 042670 001403      BEQ      39$ ;NO-SKIP
9224 042672 104035      ERROR   35 ;REPORT NED & UFE BOTH SET
9225 042674 000137 044236      JMP      37$
9226 042700 104004      ERROR   4 ;REPORT UFE ONLY
9227 042702 000137 044236      JMP      37$
9228 042706 032765 000100 000014 4$:     BIT      #CMDT0,P.PRST(R5) ;
9229 042714 001423      BEQ      5$
9230 042716 004737 044712      JSR      PC,TOPROC ;GO PROCESS TIMEOUT
9231 042722 032737 002000 005474    BIT      #TWOTOS,RECODE ;2ND TIMEOUT IN TIMEOUT PROC?
9232 042730 001007      BNE     40$ ;YES - SKIP TO ERROR 56
9233 042732 032737 000400 005474    BIT      #LEV2ER,RECODE ;TEST IF LEVEL 2 ERROR
9234 042740 001006      BNE     41$ ;YES - SKIP TO ERROR 57
9235 042742 104005      ERROR   5 ;ELSE MAKE FULL TIMEOUT REPORT
9236 042744 000137 044236      JMP      37$
9237 042750 104056      ERROR   56 ;TWO TIMEOUTS ERROR REPORT
9238 042752 000137 044236      JMP      37$
9239 042756 104057      ERROR   57 ;2ND LEVEL ERROR REPORT
9240 042760 000137 042506      JMP      ER2ENT ;GO BUILD AND MAKE 2ND REPORT
9241 042764 032765 010000 000014 5$:     BIT      #DRVSZD,P.PRST(R5) ;SEE IF DRIVE SIEZED BY OTHER PORT
9242 042772 001431      BEQ      65$ ;BR IF DRIVE IS AVAILABLE

```

:CONSIDERED SIGNIFICANT IN
:THAT ERRORS OF A MORE
:CATASTROPHIC NATURE ARE FIRST
:TESTED.

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

:TEST NON-EXISTANT MEMORY

:TEST NON-EXISTANT DRIVE

:TEST IF NED & UFE BOTH SET

:NED & UFE BOTH SET ERROR

:NED ONLY

:TEST UNIT FIELD ERROR

:TEST IF UFE & NED BOTH SET

:NO-SKIP

:REPORT NED & UFE BOTH SET

:REPORT UFE ONLY

:GO PROCESS TIMEOUT

:2ND TIMEOUT IN TIMEOUT PROC?

:YES - SKIP TO ERROR 56

:TEST IF LEVEL 2 ERROR

:YES - SKIP TO ERROR 57

:ELSE MAKE FULL TIMEOUT REPORT

:TWO TIMEOUTS ERROR REPORT

:2ND LEVEL ERROR REPORT

:GO BUILD AND MAKE 2ND REPORT

:SEE IF DRIVE SIEZED BY OTHER PORT

:BR IF DRIVE IS AVAILABLE

9243	042774	105737	003126		TSTB	DULACS		;SEE IF DUAL-ACCESS FLAG SET
9244	043000	001003			BNE	63\$;BR IF DUAL-ACCESS TEST
9245	043002	104107			ERROR	107		;DRIVE SIEZED BY OTHER PORT
9246	043004	000137	044236		JMP	37\$		
9247	043010	105237	003127	63\$:	INCB	DRNAFG		;SET DRIVE NOT AVAILABLE FLAG
9248	043014	012762	100000	000000	MOV	#100000,RKCS1(R2)		;CLEAR CONTROLLER ERROR
9249	043022	013737	003056	003104	MOV	W.MIN,W.DRV		;SET TIMER FOR ABOUT A MINUTE
9250	043030	113700	005500		MOVB	DRIVE,R0		;DRIVE NUMBER
9251	043034	116037	003072	003071	MOVB	I.DRV(R0),INTMSK		;SET MASK FOR THIS DRIVE
9252	043042	113737	003071	003070	MOVB	INTMSK,W.TIME		;SET DRIVE NUMBER FOR THIS DRIVE
9253	043050	105037	003113		CLRB	DONE		;CLEAR THE DONE FLAG
9254	043054	000207			RTS	PC		;GO WAIT FOR DRIVE ATT'N
9255	043056	032765	020000	000016	65\$:	BIT	#SPAR,P.CS1(R5)	;TEST D TO C PARITY ERROR
9256	043064	001406			BEQ	6\$		
9257	043066	104006			ERROR	6		
9258	043070	052737	004000	005474	BIS	#RCLREQ,RECODE		
9259	043076	000137	044236		JMP	37\$		
9260	043102	032704	000010	6\$:	BIT	#DRPAR,R4		;TEST DRIVE DETECTED PARITY ERROR
9261	043106	001406			BEQ	7\$		
9262	043110	104007			ERROR	7		
9263	043112	052737	004000	005474	BIS	#RCLREQ,RECODE		
9264	043120	000137	044236		JMP	37\$		
9265	043124	032765	000010	000036	7\$:	BIT	#ACLO,P.DS(R5)	;TEST AC LOW
9266	043132	001403			BEQ	8\$		
9267	043134	104010			ERROR	10		
9268	043136	000137	044236		JMP	37\$		
9269	043142	032765	000020	000036	8\$:	BIT	#SPDLSS,P.DS(R5)	;TEST SPEED LOSS
9270	043150	001403			BEQ	24\$		
9271	043152	104011			ERROR	11		
9272	043154	000137	044236		JMP	37\$		
9273	043160	032765	004000	000016	24\$:	BIT	#CT0,P.CS1(R5)	;TEST FOR CONTROLLER TIMEOUT
9274	043166	001401			BEQ	25\$		
9275	043170	104027			ERROR	27		
9276	043172	032704	000001	25\$:	BIT	#ILC,R4		;TEST ILLEGAL FUNCTION CODE
9277	043176	001403			BEQ	10\$		
9278	043200	104012			ERROR	12		
9279	043202	000137	044236		JMP	37\$		
9280	043206	032765	002000	000020	10\$:	BIT	#PGE,P.CS2(R5)	;TEST PROGRAMMING ERROR
9281	043214	001403			BEQ	11\$		
9282	043216	104013			ERROR	13		
9283	043220	000137	044236		JMP	37\$		
9284	043224	032704	000004	11\$:	BIT	#ILF,R4		;TEST ILLEGAL DRIVE FUNCTION
9285	043230	001403			BEQ	12\$		
9286	043232	104014			ERROR	14		
9287	043234	000137	044236		JMP	37\$		
9288	043240	032704	000040	12\$:	BIT	#DTYE,R4		;TEST DRIVE TYPE ERROR
9289	043244	001403			BEQ	13\$		
9290	043246	104015			ERROR	15		
9291	043250	000137	044236		JMP	37\$		
9292	043254	032704	000020	13\$:	BIT	#FMTE,R4		;TEST FORMAT ERROR
9293	043260	001403			BEQ	14\$		
9294	043262	104016			ERROR	16		
9295	043264	000137	044236		JMP	37\$		
9296	043270	032704	004000	14\$:	BIT	#WLE,R4		;TEST WRITE LOCK ERROR
9297	043274	001403			BEQ	15\$		
9298	043276	104017			ERROR	17		

```

9299 043300 000137 044236          JMP      37$
9300 043304 032704 040000          15$:    BIT      #UNS,R4          ;TEST DRIVE UNSAFE
9301 043310 001406          BEQ     16$
9302 043312 104020          ERROR   20
9303 043314 052737 000200 005474          BIS     #ABORT,RECODE
9304 043322 000137 044336          JMP     ALLTRM
9305 043326 032704 000002          16$:    BIT      #SKI,R4          ;TEST SEEK INCOMPLETE
9306 043332 001406          BEQ     17$
9307 043334 104021          ERROR   21
9308 043336 052737 004000 005474          BIS     #RCLREQ,RECODE
9309 043344 000137 044236          JMP     37$
9310 043350 032704 001000          17$:    BIT      #COE,R4          ;TEST CYLINDER OVERFLOW
9311 043354 001406          BEQ     18$
9312 043356 104022          ERROR   22
9313 043360 052737 004000 005474          BIS     #RCLREQ,RECODE
9314 043366 000137 044236          JMP     37$
9315 043372 032704 002000          18$:    BIT      #IDAE,R4         ;TEST ILLEGAL CYLINDER
9316 043376 001406          BEQ     19$
9317 043400 104023          ERROR   23
9318 043402 052737 004000 005474          BIS     #RCLREQ,RECODE
9319 043410 000137 044236          JMP     37$
9320 043414 032765 000040 000036 19$:    BIT      #DROT,P.DS(R5)   ;TEST DRIVE OFF TRACK
9321 043422 001406          BEQ     20$
9322 043424 104024          ERROR   24
9323 043426 052737 004000 005474          BIS     #RCLREQ,RECODE
9324 043434 000137 044236          JMP     37$
9325 043440 032704 010000          20$:    BIT      #DTE,R4          ;TEST DRIVE TIMING ERROR
9326 043444 001406          BEQ     21$
9327 043446 104025          ERROR   25
9328 043450 052737 000200 005474          BIS     #ABORT,RECODE
9329 043456 000137 044236          JMP     37$
9330 043462 032765 100000 000020 21$:    BIT      #DLT,P.CS2(R5)   ;TEST DATA LATE
9331 043470 001403          BEQ     22$
9332 043472 104026          ERROR   26
9333 043474 000137 044236          JMP     37$
9334 043500 032704 020000          22$:    BIT      #OPI,R4          ;TEST IF OPI ERROR
9335 043504 001457          BEQ     29$
9336 043506 052737 000010 005474          BIS     #OPIERR,RECODE
9337 043514 105737 003111          TSTB   DERCNT          ;TEST IF FIRST ERROR
9338 043520 001402          BEQ     50$
9339 043522 000137 044236          JMP     37$          ;NO - SKIP REPORT
9340 043526 032737 000400 005474 50$:    BIT      #LEV2ER,RECODE   ;TEST IF A SECOND LEVEL 2 ERROR
9341 043534 001403          BEQ     26$          ;HAS ALREADY OCCURRED
9342 043536 104054          ERROR   54
9343 043540 000137 044236          JMP     37$          ;GET OUT OF ERROR REPORT
9344 043544 004737 045342          26$:    JSR     PC,BLDEXH      ;GO BUILD EXPECTED HEADER
9345 043550 004737 045062          JSR     PC,RDH0        ;GET HEADER OF SECTOR 0
9346 043554 032737 000400 005474          BIT      #LEV2ER,RECODE   ;TEST IF ERROR GETTING HDR
9347 043562 001025          BNE     28$          ;IF YES-GO MAKE ABBREVIATED REPORT
9348 043564 013702 003026          MOV     RKBAS,R2      ;STORE HEADER 0 INTO REGISTERS
9349 043570 042762 000100 000000          BIC     #IE,RKCS1(R2)   ;RESET INTERRUPT ENABLE
9350 043576 016237 000024 001202          MOV     RKDB(R2),$REG10 ;FOR REPORTING
9351 043604 016237 000024 001204          MOV     RKDB(R2),$REG11
9352 043612 016237 000024 001206          MOV     RKDB(R2),$REG12
9353 043620 032762 100000 000000          BIT      #CERR,RKCS1(R2) ;TEST IF ERROR DURING STORAGE
9354 043626 001343          BNE     27$

```

```

9355 043630 104030          ERROR 30          ;MAKE OPI REPORT
9356 043632 000137 044236    JMP      37$
9357 043636 104054          ERROR 54
9358 043640 000137 042506    28$:  JMP      ER2ENT      ;GO MAKE 2ND LEVEL REPORT
9359 043644 032704 000400    29$:  BIT      #HVRC,R4    ;TEST IF HVRC ERROR
9360 043650 001457          BEQ      23$
9361 043652 052737 000004 005474    BIS      #HVCRCR,RECODE
9362 043660 105737 003111          TSTB    DERCNT      ;TEST IF FIRST ERROR
9363 043664 001402          BEQ      30$          ;YES - REPORT
9364 043666 000137 044236    JMP      37$          ;JUMP TO RETURN
9365 043672 032737 000400 005474 30$:  BIT      #LEV2ER,RECODE ;TEST IF A 2ND LEVEL ERROR HAS ALREADY
9366 043700 001403          BEQ      31$          ;OCCURRED. NO-SKIP EXIT
9367 043702 104055          ERROR 55
9368 043704 000137 044236    JMP      37$          ;GET OUT OF ERROR REPORT
9369 043710 004737 045342    31$:  JSR      PC,BLDEXH   ;GO BUILD EXPECTED HEADER
9370 043714 004737 045062    JSR      PC,RDHDO    ;GO GET HDR 0
9371 043720 032737 000400 005474    BIT      #LEV2ER,RECODE ;TEST IF ERROR IN GETTING HDR
9372 043726 001025          BNE     32$          ;IF YES-GO MAKE ABBREVIATED REPORT
9373 043730 013702 003026          MOV     RKBAS,R2    ;GET RK611 BASE ADDRESS
9374 043734 042762 000100 000000    BIC     #IE,RKCS1(R2) ;CLEAR INTERRUPT ENABLE
9375 043742 016237 000024 001202    MOV     RKDB(R2), $REG10 ;STORE OFF HEADER
9376 043750 016237 000024 001204    MOV     RKDB(R2), $REG11
9377 043756 016237 000024 001206    MOV     RKDB(R2), $REG12
9378 043764 032762 100000 000000    BIT     #CERR,RKCS1(R2) ;TEST IN ANY ERROR IN UNLOAD
9379 043772 001343          BNE     51$          ;IY YES - GO MAKE SHORT REPORT
9380 043774 104031          ERROR 31          ;MAKE FULL REPORT
9381 043776 000137 044236    JMP      37$
9382 044002 104055          ERROR 55          ;ABBREVIATED HVRC ERROR RPORT
9383 044004 000137 042506    JMP      ER2ENT      ;GO REPORT 2ND LEVEL ERROR
9384 044010 032704 000200    23$:  BIT     #BSE,R4    ;TEST FOR BAD SECTOR ERROR
9385 044014 001430          BEQ     33$          ;NO - SKIP
9386 044016 052737 000002 005474    BIS     #BSERR,RECODE ;SET ERROR FLAG
9387 044024 016500 000030          MOV     P.DCYL(R5),R0 ;GET CYL NO.
9388 044030 116501 000027          MOVVB  P.DTS+1(R5),R1 ;GET TRACK NO.
9389 044034 042701 177774          BIC     #177774,R1   ;CLEAR ALL BITS EXCEPT TRACK
9390 044040 016502 000026          MOV     P.DTS(R5),R2 ;GET SECTOR IN ERROR
9391 044044 042702 177740          BIC     #177740,R2   ;CLEAR ALL BITS EXCEPT SECTOR
9392 044050 004737 040670          JSR     PC,BDSRCK   ;GO SEE IF THIS SECTOR LISTED
9393 044054 032737 001000 005474    BIT     #BADSEC,RECODE ;TEST RESULT
9394 044062 001065          BNE     37$          ;YES - EXIT. NO ERROR OR REPORT
9395 044064 104104          ERROR 104          ;ELSE REPORT BAD BSE
9396 044066 042737 000002 005474    BIC     #BSERR,RECODE ;RESET BSE ERROR FLAG
9397 044074 000460          BR     37$          ;GO EXIT
9398 044076 032704 100000    33$:  BIT     #DCK,R4    ;TEST IF DATA CHECK
9399 044102 001435          BEQ     36$
9400 044104 052737 000020 005474    BIS     #DCKERR,RECODE ;SET DATA CHECK ERROR IN RECOVERY CODE
9401 044112 032704 000100          BIT     #ECH,R4    ;TEST IF ECC IS HARD. IF
9402 044116 001406          BEQ     34$          ;YES SET UNCORRECTABLE IN
9403 044120 052737 000040 005474    BIS     #ECCNC,RECODE ;RECOVERY FLAG AND A 0 IN
9404 044126 005037 001206          CLR     $REG12      ;REG12 TO INDICATE UNCORRECTABLE,
9405 044132 000403          BR     35$          ;A 1 IN REG 12 FOR CORRECTABLE
9406 044134 012737 000001 001206 34$:  MOV     #1,$REG12
9407 044142 105737 003111    35$:  TSTB    DERCNT      ;TEST IF FIRST ERROR
9408 044146 001033          BNE     37$          ;NO SKIP REPORT
9409 044150 004737 045216          JSR     PC,GTPKAD   ;GO GET PACK ADDRESS OF ERROR
9410 044154 016537 000060 001202    MOV     P.EPOS(R5), $REG10 ;STORE ECC POSITION &

```

REPORT ERROR ROUTINE

```

9411 044162 016537 000062 001204      MOV      P.EPAT(R5), $REG11 ; PATTERN
9412 044170 104032      ERROR   32                ; REPORT DCK ERROR
9413 044172 000137 044236      JMP     37$
9414 044176 032765 040000 000020 36$:    BIT     #WCE, P.CS2(R5) ; TEST WRITE CHECK ERROR
9415 044204 001414      BEQ    37$
9416 044206 042737 000200 005474      BIC     #ABORT, RECODE ; CLEAR ABORT & SET WRITE
9417 044214 052737 000100 005474      BIS     #WCERR, RECODE ; CHECK ERROR IN RECODE
9418 044222 105737 003111      TSTB   DERCNT           ; TEST IF FIRST ERROR
9419 044226 001003      BNE    37$              ; NO - SKIP
9420 044230 004737 045216      JSR    PC, GTPKAD       ; GO GET ADDRESS OF ERROR
9421 044234 104033      ERROR  33                ; REPORT WCE

9422 044236 032765 000020 000014 37$:    BIT     #DRVHRD, P.PRST(R5) ; TEST HARD ERROR
9423 044244 001404      BEQ    43$
9424 044246 104036      ERROR  36
9425 044250 052737 000200 005474      BIS     #ABORT, RECODE

9426 044256 032765 000040 000014 43$:    BIT     #DRVDSC, P.PRST(R5) ; TEST STATUS CHANGE NOT CLEARED
9427 044264 001404      BEQ    44$
9428 044266 104037      ERROR  37
9429 044270 052737 000200 005474      BIS     #ABORT, RECODE

9430 044276 032765 004000 000014 44$:    BIT     #NODSC, P.PRST(R5) ; IFST ATTENTION BUT NO FAULT OR DSC
9431 044304 001404      BEQ    46$
9432 044306 104040      ERROR  40
9433 044310 052737 000200 005474      BIS     #ABORT, RECODE

9434 044316 032765 000010 000014 46$:    BIT     #UEXATT, P.PRST(R5) ; TEST UNEXPECTED ATTENTION
9435 044324 001404      BEQ    ALLTRM
9436 044326 104042      ERROR  42
9437 044330 052737 000200 005474      BIS     #ABORT, RECODE

9438
9439
9440
9441
9442
9443
9444
9445
9446
9447
9448
9449
9450
9451
9452
9453
9454
9455
9456
9457
9458
9459
9460
9461
9462
9463
9464
9465
9466

```

; ALL ERRORS MUST EXIT THROUGH THIS POINT

```

ALLTRM: MOV     #PARMO, R5 ; RESTORE PARAMETER BLOCK SELECTION
        MOV     #ERRHDL, A.ABNL ; RESTORE INTERRUPT VECTORS FOR RETRY
        MOV     #ERRFRE, A.NORM ; DRIVER OPERATIONS, IF ANY
        BIT     #ABORT, RECODE ; IF ABORT IS NOT SET AND
        BNE    48$ ; THE DRIVE IS READY (HAS NOT
        ; CYCLED DOWN)
        MOV     RKBAS, R2 ; GET BASE ADDRESS
        BIT     #RDY, RKDS(R2) ; TEST IF DRIVE READY SET
        BNE    47$ ; RECALIBRATE REQUIRED BIT IS SET
        BIS     #ABORT, RECODE ; ELSE ABORT WITH ABORT MESSAGE
        BR     48$
47$:    BIT     #RCLREQ, RECODE ; IF RECALIBRATE IS REQUIRED
        BEQ    BGNRTY ; FOR RETRY SET UP PARAM
        MOVB    #RECAL, P.CMND ; BLOCK TO DO IT.
        MOV     #RETANL, A.ABNL
        JSR    PC, DRVCAL
        MOV     #ERRHDL, A.ABNL ; RESTORE ERROR RETURN
        BIT     #LEV2ER, RECODE ; IF AN ERROR OCCURRED IN THE
        BEQ    BGNRTY ; RECAL ATTEMP SET ABORT.
        BIS     #ABORT, RECODE ; PRINT THE RECAL ERROR MESSAGE, AND
        ERROR  60 ; GO REPORT DETAILS

```

REPORT ERROR ROUTINE

```

9467 044470 000137 042506          JMP      ER2ENT
9468 044474 104061          48$:    ERROR    61          ;REPORT ABORT-RETRY FAILED
9469
9470          ;THE PROGRAM WILL HALT HERE IF THE ABORT FLAG HAS BEEN SET OR
9471          ;IF THE DRIVE READY BIT IS RESET.
9472
9473 044476 000000          HLTPRG: HALT
9474 044500 105037 003116          CLR      ERRCNT          ;CLEAR ERROR COUNT
9475 044504 000005          RESET
9476 044506 000137 013522          JMP      CMSTRT          ;RESET ALL DEVICES
9477
9478
9479          ;THE FOLLOWING CODE WILL DETERMINE IF AND DATA TYPE ERROR
9480          ;HAS OCCURRED. IF YES, THE RETRY IS NOT DONE HERE BUT RETURNS TO
9481          ;THE INITIATING ROUTINE FOR RETRY. ANY OTHER ERROR IS TO BE
9482          ;RETRIED HERE. IF RETRY IS UNSUCCESSFUL AFTER 4 ATTEMPTS, THE ABORT
9483          ;FLAG IS SET AND PROGRAM HALTS.
9484
9485 044512 032737 000136 005474 BGNRTY: BIT      #BSERR!HVRCCR!OPIERR!DCKERR!WCERR,RECODE ;TEST IF ANY DATA ERROR
9486 044520 001404          BEQ      3$
9487 044522 052737 100000 005474 9$:    BIS      #ANYDER,RECODE
9488 044530 000453          BR       2$
9489 044532          3$:
9490 044532 105737 003133          TSTB    NORTRY          ;SEE IF "NO-RETRY" FLAG SET
9491 044536 001371          BNE     9$             ;BR IF YES
9492 044540 032737 001000 005474          BIT      #BADSEC,RECODE ;TEST IF BAD SECTOR FLAG SET
9493 044546 001044          BNE     2$             ;IF YES-EXIT TO CALLER
9494 044550 123737 003116 003117          CMPB    ERRCNT,ERRLMT ;BEGIN RETRY IF ERROR COUNT HAS
9495 044556 001012          BNE     1$             ;NOT BEEN EXCEEDED
9496 044560 005037 001174          CLR     $REG5
9497 044564 113737 003116 001174          MOV     ERRCNT,$REG5 ;GET ERROR RETRY COUNT
9498 044572 104102          ERROR  102            ;REPORT RETRY UNSUCCESSFUL
9499 044574 052737 000200 005474          BIS     #ABORT,RECODE ;SET ABORT & QUIT
9500 044602 000735          BR      HLTPRG
9501 044604 013702 003026          1$:    MOV     RKBAS,R2      ;GET RK BASE ADDRESS
9502 044610 112765 000177 000001          MOV     #SUBCLR.P.CMND(R5) ;SET UP TO CLEAR SUBSYSTEM
9503 044616 004737 040774          JSR     PC,DRVCAL    ;GO DO IT
9504 044622 012700 005236          MOV     #COMSTR,R0   ;GO AND REESTABLISH THE COMMAND
9505 044626 012025          MOV     (R0)+,(R5)+ ;AS IT WAS ENTERED INTO THE
9506 044630 012025          MOV     (R0)+,(R5)+ ;PARAMETER BLOCK
9507 044632 012025          MOV     (R0)+,(R5)+
9508 044634 012025          MOV     (R0)+,(R5)+
9509 044636 012025          MOV     (R0)+,(R5)+
9510 044640 012025          MOV     (R0)+,(R5)+
9511 044642 012705 002620          MOV     #PARMO,R5
9512 044646 012737 000000 177776          MOV     #PRO.2#PSW ;LOWER PRIORITY TO ALLOW INTERRUPT
9513 044654 004737 040774          JSR     PC,DRVCAL    ;CALL DRIVER
9514 044660 104410          2$:    RESREG ;IF RETURN GETS HERE, NO ERROR
9515 044662 000207          RTS     PC            ;OCCURRED, RECOVERY WAS SUCCESSFUL
9516
9517
9518 044664 152737 000377 003113 RETANL: BISB    #377,DONE ;SET DONE
9519 044672 052737 000400 005474          BIS     #LEV2ER,RECODE ;SET LEVEL TWO ERROR
9520 044700 000207          RTS     PC
9521 044702 152737 000377 003113 RETNML: BISB    #377,DONE ;SET DONE
9522 044710 000207          RTS     PC

```

9523
9524
9525
9526
9527
9528
9529
9530
9531
9532
9533
9534
9535
9536
9537
9538
9539
9540
9541
9542
9543
9544
9545
9546
9547
9548
9549
9550
9551
9552
9553
9554
9555
9556
9557
9558
9559
9560
9561
9562
9563
9564
9565
9566
9567
9568
9569
9570
9571
9572
9573
9574
9575
9576
9577
9578

044712
044712 104407
044714 013702 003026
044720 012701 001174
044724 016221 000000
044730 016221 000010
044734 016221 000020
044740 016221 000006
044744 016221 000002
044750 016221 000004
044754 016221 000016
044760 016221 000012
044764 016221 000014
044770 005000

044772 012705 002704
044776 112765 000141 000001
045004 004737 040774
045010 032765 000100 000014
045016 001403
045020 052737 002000 005474
045026 062705 000040
045032 012521
045034 012521
045036 012521
045040 012521
045042 012521
045044 012521
045046 012521
045050 012521

045052 012705 002620
045056 104410
045060 000207

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

```
TOPROC:
  SAVREG
  MOV   RKBAS,R2
  MOV   #RREG5,R1      ;SET UP R1 FOR RK REGISTER STORAGE
  MOV   RKCS1(R2),(R1)+ ;STORE ALL VALID RK611
  MOV   RKCS2(R2),(R1)+ ;REGISTERS
  MOV   RKDC(R2),(R1)+
  MOV   RKDA(R2),(R1)+
  MOV   RKWC(R2),(R1)+
  MOV   RKBA(R2),(R1)+
  MOV   RKASOF(R2),(R1)+
  MOV   RKDS(R2),(R1)+
  MOV   RKER(R2),(R1)+
  CLR   R0              ;THIS CODE WILL ATTEMPT TO
                       ;RETRIEVE THE STATUS FROM THE
                       ;DRIVE.
  MOV   #PARM1,R5      ;SET UP TO USE PARM1
  MOVB  #RDSTAT,P.CMND(R5) ;DO READ DRIVE STATUS COMMAND
  JSR   PC,DRVCL      ;CALL DRIVER
  BIT   #CMDTO,P.PRST(R5) ;TEST FOR TIMEOUT
  BEQ   1$             ;NO - SKIP
  BIS   #TWOTOS,RECODE ;SET TWO TIMEOUTS FLAG
  ADD   #P.ADD,R5      ;BUMP R5 TO POINT TO DRIVE STATUS
  MOV   (R5)+,(R1)+    ;MOVE ALL THE DRIVE STATUS INTO THE
                       ;TEMP REGS FOR REPORTING.
  MOV   (R5)+,(R1)+
  MOV   (R5)+,(R1)+
  MOV   (R5)+,(R1)+
  MOV   (R5)+,(R1)+
  MOV   (R5)+,(R1)+
  MOV   (R5)+,(R1)+
  MOV   (R5)+,(R1)+
  MOV   #PARM0,R5      ;RESTORE PARM 0
  RESREG
  RTS   PC
```

:SBTTL READ HEADER 0 ROUTINE

```
R0HDO:
  SAVREG
  MOV   P.DTS(R5),R1   ;STORE TRACK AND SECTOR
  MOV   P.B10(R5),R0  ;GET THE CYLINDER ADRS
  BIC   #160017,R0    ;FROM THE DRIVE STATUS. CLEAR
  ASR   R0             ;OFF UNUSED BITS AND POSITION
  ASR   R0             ;FOR USE AS THE DESIRED
  ASR   R0             ;CYLINDER IN THE READ
```

```

9579 045106 006200          ASR      RO          ;HEADER COMMAND.
9580 045110 012705 002704    MOV      #PARM1,R5      ;SET UP TO USE P.B. 1
9581 045114 010165 000004    MOV      R1,P.SECT(R5) ;INSERT TRK,SECT FOR READ HDR
9582 045120 010065 000002    MOV      RO,P.CYLN(R5) ;SET CYL NO.
9583 045124 132737 000020 003115 BITB     #B.CFMT,FORMAT ;TEST PRESENT FMT AND CHANGE IT
9584 045132 001404          BEQ      1$           ;TO THE OPPOSITE. THIS WILL CAUSE
9585 045134 142765 000020 000007 BICB     #B.CFMT,P.CS1H(R5) ;A READ OF SECT 0 HDR ON
9586 045142 000403          BR       2$           ;THE READ HDR COMMAND
9587 045144 152765 000020 000007 1$:     BISB     #B.CFMT,P.CS1H(R5)
9588 045152 112765 000125 000001 2$:     MOVB     #RDHEAD,P.CMND(R5) ;SET READ HDR COMMAND
9589 045160 012737 000000 177776    MOV      #PRO.2#PSW    ;ALLOW INTERRUPTS
9590 045166 004737 040774    JSR      PC,DRVCAL     ;DO READ HDR
9591 045172 142765 000020 000007 BICB     #B.CFMT,P.CS1H(R5) ;CLEAR THE FORMAT BIT
9592 045200 153765 003115 000007 BISB     FORMAT,P.CS1H(R5) ;RESTORE TYPE AND FMT IN USE
9593 045206 012705 002620    MOV      #PARMO,R5     ;RESTORE P.B. 0 ADDRESS
9594 045212 104410    RESREG  PC           ;RESTORE RO-R5
9595 045214 000207          RTS      PC           ;RETURN
9596
9597
9598
9599
9600
9601
9602 045216 016537 000030 001174 GTPKAD: MOV      P.DCYL(R5),$REG5 ;GET CYLINDER NUMBER
9603 045224 005037 001176    CLR      $REG6         ;CLEAR REGISTERS FOR
9604 045230 005037 001200    CLR      $REG7         ;TRACK & SECTOR STORAGE
9605 045234 116537 000026 001200    MOVB     P.DTS(R5),$REG7 ;STORE THE TRACK AND SECTOR
9606 045242 116537 000027 001176    MOVB     P.DTS+1(R5),$REG6
9607 045250 005737 001200    TST      $REG7         ;ADJUST THE ADDRESS CONTAINED IN
9608                                ;THE RK REGISTERS FOR THE AUTOMATIC
9609                                ;INCREMENT
9609 045254 001403          BEQ      1$
9610 045256 005337 001200    DEC      $REG7
9611 045262 000426          BR       5$
9612 045264 032765 010000 000016 1$:     BIT      #CFMT,P.CS1(R5)
9613 045272 001404          BEQ      2$
9614 045274 012737 000023 001200    MOV      #19.,$REG7
9615 045302 000403          BR       3$
9616 045304 012737 000025 001200 2$:     MOV      #21.,$REG7
9617 045312 005737 001176    3$:     TST      $REG6
9618 045316 001403          BEQ      4$
9619 045320 005337 001176    DEC      $REG6
9620 045324 000405          BR       5$
9621 045326 012737 000002 001176 4$:     MOV      #2,$REG6
9622 045334 005337 001174    DEC      $REG5
9623 045340 000207          5$:     RTS      PC
9624
9625
9626
9627
9628
9629
9630
9631
9632
9633 045342 104407          BLDEXH: SAVREG
9634 045344 016537 000030 001174    MOV      P.DCYL(R5),$REG5 ;CONSTRUCT EXPECTED HDR

```

N14

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 183
BUILD EXPECTED HEADER

SEQ 0192

9635	045352	016501	000026		MOV	P.DTS(R5),R1	; DESIRED CYLINDER & DESIRED TRACK
9636	045356	042701	174377		BIC	#174377,R1	; CLEAR ALL BUT TRACK BITS
9637	045362	006201			ASR	R1	; AND SECTOR. SHIFT THE TRACK
9638	045364	006201			ASR	R1	; OVER TO CONFORM TO HEADER FORMAT
9639	045366	006201			ASR	R1	; CHECK THE FORMAT BIT AND
9640							; IF SET, SET THE HEADER FORMAT
9641	045370	016537	000026	001176	MOV	P.DTS(R5), \$REG6	; BIT.
9642	045376	042737	177740	001176	BIC	#177740, \$REG6	; CLEAR ALL BUT SECTOR
9643	045404	060137	001176		ADD	R1, \$REG6	; ADD TRACK AND SECTOR TOGETHER
9644	045410	052737	140000	001176	BIS	#140000, \$REG6	; INSERT BSE BITS
9645	045416	032765	010000	000016	BIT	#CFMT, P.CS1(R5)	
9646	045424	001403			BEQ	23\$	
9647	045426	052737	001000	001176	BIS	#1000, \$REG6	
9648	045434	013737	001174	001200	MOV	\$REG5, \$REG7	; COMPUTE THE HEADER VRC
9649	045442	013701	001176		MOV	\$REG6, R1	
9650	045446	043737	001176	001200	BIC	\$REG6, \$REG7	
9651	045454	043701	001174		BIC	\$REG5, R1	
9652	045460	050137	001200		BIS	R1, \$REG7	
9653	045464	104410			RESREG		
9654	045466	000207			RTS	PC	
9655							

9696
9697
9698
9699
9700
9701
9702
9703
9704
9705
9706
9707
9708
9709
9710
9711

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

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

.SBTTL *WATCH-DOG TIMER

* THE WATCH-DOG TIMER DOES A PSEUDO-TIMING OF RK06 UNIBUS
* SUBSYSTEM COMMAND. SINCE ONE CAN NOT GUARANTEE THAT A
* REAL-TIME CLOCK (KW11-P OR KW11-L) IS ON THE SYSTEM
* THE RK06 DRIVER WILL USE THE LOCATION W.MTIM FOR
* MILLI-SECOND TIMING. WHEN W.MTIM REACHES ZERO THE
* WATCH-DOG TIMER WILL SCAN THE DRIVES IN USE AS
* DETERMINED BY THE LOCATION W.TIME. THE TIMER COUNTS
* (ONE FOR EACH DRIVE) ARE KEPT IN THE TABLE W.DRV.
* IF ANY COUNT IN THE TABLE W.DRV REACHES ZERO A COMMAND
* TIME-OUT WILL BE DESIGNATED IN THE PROGRAM DEVICE STATUS
* REGISTER OF THAT DRIVE'S PARAMETER BLOCK.

* THE DRIVER WILL USE THE LOCATION W.MIN AS THE NUMBER
* OF MILLISECONDS FOR AN UNLOAD OR START SPINDLE COMMAND.
* THE DRIVER WILL USE THE LOCATION W.SEC AS THE TIME
* LIMIT FOR ALL OTHER COMMANDS.

* FOR QUEUED OPERATIONS THE WATCH-DOG TIMER WILL
* WATCH UP TO 8 OPERATIONS SIMULTEOUSLY. FOR SEQUENTIAL
* OPERATIONS ONLY ONE OPERATION WILL BE WATCHED.

*CALL JSR PC,W.WTCH
* RETURN IF NO DRIVE ORDER EXCEEDED ITS TIME LIMIT

* OTHERWISE AN ABNORMAL RETURN TO THE ROUTINE ADDRESS
* BY LOCATION A.ABNL WILL OCCUR AND THE CMDTO FLAG
* IN THE PROGRAM DEVICE STATUS REGISTER OF THE
* APPROPRIATE PARAMETER BLOCK WILL BE SET.

9698 045470 010546
9699 045472 010446
9700 045474 010346
9701 045476 010246
9702 045500 013746 177776
9703 045504 005337 003046
9704 045510 001034
9705 045512 013737 003050 003046
9706 045520 105737 003070
9707 045524 001426
9708 045526 013737 003032 177776
9709 045534 013702 003026
9710 045540 005337 003104
9711 045544 001016

W.WTCH: MOV R5,-(SP) ;SAVE R5 ON THE STACK
MOV R4,-(SP) ;SAVE R4 ON THE STACK
MOV R3,-(SP) ;SAVE R3 ON THE STACK
MOV R2,-(SP) ;SAVE R2 ON STACK
MOV PS,-(SP) ;SAVE PROGRAM STATUS WORD ON STACK
DEC W.MTIM ;DECREMENT MILLISECOND TIMER
BNE 20\$;IF NOT ZERO RETURN
MOV W.MILI,W.MTIM ;REINITIALIZE MILLISECOND TIMER
TSTB W.TIME ;CHECK IF DRIVE IS BEING TIMED
BEQ 20\$;NO RETURN
MOV RKPRI,PS ;LOCK OUT RK06 INTERRUPTS
MOV RKBAS,R2 ;LOAD BASE OF RK06 REGISTERS
DEC W.DRV ;DECREMENT COMMAND TIMER
BNE 20\$;RETURN IF NO TIME OUT

*WATCH-DOG TIMER

9713	045546	105037	003070		CLAB	W.TIME	:RESET TIMING INDICATOR
9714	045552	013705	003102		MOV	2BLKT,RS	:LOAD ADDRESS OF PARAMETER BLOCK
9715							:TABLE FOR INDEXING
9716	045556	052765	000100	000014	BIS	@CMDTO.P.PRST(RS)	:SET COMMAND TIME OUT
9717	045564	020537	003044		CMF	RS.O.WAIT	:CHECK IF DRIVER IS WAITING FOR
9718							:COMMAND COMPLETION
9719	045570	001002			BNE	SS	:NO, DO NOT ALTER WAITING FOR
9720							:COMMAND COMPLETION
9721	045572	005037	003044		CLR	O.WAIT	:CLEAR WAIT FOR COMMAND COMPLETION
9722	045576	004737	051052		JSR	PC,R.ABNL	:BRANCH TO ERROR ROUTINE
9723	045580	012637	177776	SS:	MOV	(SP)+,PSW	:RESTORE PSW
9724	045584	012602		RS:	MOV	(SP)+,R2	:RESTORE R2
9725	045588	012602			MOV	(SP)+,R3	:RESTORE R3
9726	045592	012604			MOV	(SP)+,R4	:RESTORE R4
9727	045596	012605			MOV	(SP)+,R5	:RESTORE R5
9728	045600	000207			RTS	PC	:RETURN

9728
9729
9730
9731
9732
9733
9734
9735
9736
9737
9738
9739
9740
9741
9742
9743
9744
9745
9746
9747
9748
9749
9750
9751
9752
9753
9754
9755
9756
9757
9758
9759
9760
9761
9762
9763
9764
9765
9766
9767
9768
9769
9770
9771
9772
9773
9774
9775
9776
9777
9778
9779
9780
9781
9782
9783

.SBTTL *RK06 INTERRUPT SERVICE ROUTINE

```

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

```

```

045620 010546
045622 010446
045624 010346
045626 010246
045630 010146
045632 010046
045634 013702 003026
045640 016237 000010 002772
045646 032737 001000 002772
045654 001407
045656 052737 100000 003042
045664 004737 051076

```

```

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

```

```

9784 045670 000137 050050          JMP      I.RTRN          ;RETURN
9785 045674 105737 003064          1$:    TSTB      I.ISRL          ;CHECK IF INTERRUPT OR RELEASE
9786 045700 001410          BEQ      6$              ;NO, CHECK IF DRIVE AVAILABLE
9787 045702 100403          BMI      5$              ;CHECK IF RELEASE COMMAND
9788 045704 105037 003064          CLRB     I.ISRL          ;YES, CLEAR FLAG
9789 045710 000473          BR       I.I00           ;CONTINUE PROCESSING INTERRUPT
9790
9791 045712 105037 003064          5$:    CLRB     I.ISRL          ;CLEAR FLAG
9792 045716 000137 047032          JMP      I.ATTN          ;GO PROCESS DRIVE ATTENTIONS
9793
9794 045722 032737 010400 002772 6$:    BIT       #NED!UFE,T.CS2 ;CHECK FOR NON-EXISTENT DRIVE OR
9795                                     ;UNIT FIELD ERROR
9796 045730 001413          BEQ      7$              ;NO, WAIT FOR DUAL ACCESS INTERRUPT
9797 045732 013704 002772          MOV      T.CS2,R4        ;LOAD R4 FOR DRIVE NUMBER
9798 045736 042704 177770          BIC      #1<DRVMSK>,R4   ;KEEP DRIVE BITS
9799 045742 013705 003102          MOV      PBLKT,R5        ;STORE PARAMETER BLOCK ADDRESS
9800 045746 016237 000000 002770          MOV      RKCS1(R2),T.CS1 ;LOAD TEMPORARY CS1 FOR STATUS REPORT
9801 045754 000137 046242          JMP      I.ERRC          ;REPORT ERROR
9802
9803 045760 016237 000012 003010 7$:    MOV      RKDS(R2),T.DS   ;STORE STATUS REGISTER FOR COMPARISON
9804 045766 032737 000001 003010          BIT      #DRA,T.DS      ;CHECK IF DRIVE SEIZED BY OTHER
9805                                     ;PORT
9806 045774 001041          BNE      I.I00           ;NO, CONTINUE PROCESSING INTERRUPT
9807
9808                                     ;CHECK IF ANY DATA TRANSFER ERROR EXISTS
9809 045776 032737 164000 002772          BIT      #DLT!WCE!UPE!NEM,T.CS2
9810
9811 046004 001007          BNE      10$             ;INDICATE ERROR
9812 046006 016237 000014 003006          MOV      RKER(R2),T.ER   ;STORE ERROR REGISTER
9813
9814                                     ;CHECK FOR DATA TRANSFER ERROR TYPE ERROR
9815 046014 032737 125700 003006          BIT      #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
9816
9817 046022 001407          BEQ      11$             ;NO, WAIT FOR RELEASE OF RK06 DRIVE
9818
9819 046024 052737 000010 003042 10$:   BIS      #E.UDAT,E.CONT  ;SET UNEXPECTED DATA TYPE ERROR
9820 046032 004737 051076          JSR      PC,R.CONT       ;REPORT ERROR
9821 046036 000137 050050          JMP      I.RTRN          ;RESTORE REGISTERS
9822
9823 046042 105037 003070          11$:   CLRB     W.TIME          ;RESET TIMING ON THIS DRIVE
9824 046046 005037 003104          CLR      W.DRV           ;CLEAR TIMING COUNT FOR THIS DRIVE
9825 046052 013705 003102          MOV      PBLKT,R5        ;LOAD R5 WITH PARAMETER BLOCK
9826                                     ;ADDRESS
9827 046056 052765 010000 000014          BIS      #DRVSZD,P.PRST(R5) ;SET DRIVE SEIZED IN THE
9828                                     ;PROGRAM DRIVE STATUS REGISTER
9829 046064 005037 003044          CLR      O.WAIT          ;CLEAR WAIT FOR COMMAND COMPLETION
9830 046070 004737 051052          JSR      PC,R.ABNL       ;INDICATE ABNORMAL TERMINATION
9831 046074 000137 050050          JMP      I.RTRN          ;GO RESTORE REGISTERS
9832
9833 046100 013705 003044          I.I00: MOV      O.WAIT,R5      ;LOAD PARAMETER BLOCK ADDRESS INTO R5
9834 046104 001002          BNE      2$              ;IS COMMAND WAITING PROCESSING
9835                                     ;YES, DO PROCESSING
9836 046106 000137 047032          JMP      I.ATTN          ;NO, PROCESS ATTENTION
9837
9838 046112 013704 002772          2$:    MOV      T.CS2,R4        ;STORE RKCS2 FOR DRIVE NUMBER

```

F15

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 188
 *RK06 INTERRUPT SERVICE ROUTINE

SEQ 0187

9840	046116	042704	177770			BIC	#IC<DRVMSK>,R4	:MASK OUT UNNECESSARY BITS
9841								
9842								
9843	046122	126504	000000			CMPB	P.DRVN(R5),R4	:CHECK IF DRIVE NUMBER IS EXPECTED
9844	046126	001401				BEQ	3\$:YES, CONTINUE
9845	046130	000000				HALT		:NO, DRIVER ERROR
9846	046132	122765	000164	000001	3\$:	CMPB	#RDALHD,P.CMND(R5)	:CHECK IF READ ALL HEADERS
9847	046140	001002				BNE	10\$:NO, EXECUTE NORMAL DATA TRANSFER
9848	046142	000137	046500			JMP	I.HDAL	:GO EXECUTE SPECIAL HEADER SEQUENCE
9849								
9850	046146	005037	003044		10\$:	CLR	O.WAIT	:CLEAR WAIT FOR COMMAND COMPLETION
9851	046152	005037	003104			CLR	W.DRV	:CLEAR WATCH-DOG TIME
9852	046156	105037	003070			CLRB	W.TIME	:RESET TIMING ON THIS DRIVE
9853	046162	016237	000000	002770		MOV	RKCS1(R2),T.CS1	:STORE COMMAND AND STATUS REGISTER 1
9854	046170	032737	100000	002770		BIT	#CERR,T.CS1	:CHECK IF CONTROLLER ERROR
9855	046176	001021				BNE	I.ERRC	:YES, PROCESS ERROR
9856	046200	016237	000016	003004		MOV	RKASOF(R2),T.ASOF	:STORE ATTENTION SUMMARY
9857	046206	133737	003071	003005		BITB	INTMSK,T.ASOF+1	:CHECK IF DRIVE ATTENTION SET
9858	046214	001004				BNE	15\$:YES, REPORT ERROR
9859	046216	004737	051064			JSR	PC.R.NORM	:INDICATE NORMAL RETURN
9860	046222	000137	050050			JMP	I.RTRN	:RESTORE REGISTERS
9861								
9862	046226	052765	000010	000014	15\$:	BIS	#UEXATT,P.PRST(R5)	:SET UNEXPECTED ATTENTION
9863								
9864	046234	004737	050520		I.ERRA:	JSR	PC,I.CSTS	:STORE CONTROLLER STATUS
9865	046240	000405				BR	I.ERR	:STORE PATTERN AND POSITION INFORMATION
9866								
9867	046242	013765	002770	000016	I.ERRC:	MOV	T.CS1,P.CS1(R5)	:GET ERROR RKCS1
9868	046250	004737	050542			JSR	PC,I.CST1	:GET REST OF CONTROLLER STATUS
9869	046254	016265	000032	000062	I.ERR:	MOV	RKECPT(R2),P.EPAT(R5)	:STORE ECC PATTERN
9870	046262	016265	000030	000060		MOV	RKECPS(R2),P.EPOS(R5)	:STORE ECC POSITION
9871	046270	004037	050066			JSR	RD,I.CCLR	:CLEAR CONTROLLER
9872	046274	050050				I.RTRN		:ERROR RETURN
9873	046276	032765	010400	000020		BIT	#NED!UFE,P.CS2(R5)	:CHECK IF IT WAS NON-EXISTENT DRIVE OR
9874								: UNIT FIELD ERROR
9875	046304	001046				BNE	5\$:YES, REPORT ERROR
9876	046306	004037	050624			JSR	RD,I.STAT	:GATHER DRIVE STATUS
9877	046312	050050				I.RTRN		:ERROR RETURN
9878	046314	112737	000005	002770		MOVB	#DR.CLR,T.CS1	:LOAD COMMAND
9879	046322	004037	050150			JSR	RD,I.ISSU	:ISSUE DRIVE CLEAR
9880	046326	050050				I.RTRN		:ERROR RETURN
9881	046330	133737	003071	003005		BITB	INTMSK,T.ASOF+1	:CHECK IF ATTENTION RESET
9882	046336	001407				BEQ	2\$:NO, INDICATE DRIVE ERROR
9883	046340	052737	000020	003042		BIS	#E.CLAT,E.CONT	:SET ATTENTION DID NOT RESET
9884								: WITH CLEAR
9885	046346	004737	051076			JSR	PC.R.CONT	:REPORT CONTROLLER ERROR
9886	046352	000137	050050			JMP	I.RTRN	:GO RESTORE REGISTERS
9887								
9888	046356	032737	040000	003014	2\$:	BIT	#S.DSC,T.MR2	:CHECK IF DRIVE STATUS CHANGE CLEARED
9889	046364	001403				BEQ	3\$:YES, CHECK FAULT
9890	046366	052765	000040	000014		BIS	#DRVDSC,P.PRST(R5)	:SET DSC DID NOT CLEAR
9891	046374	032737	001000	003016	3\$:	BIT	#S.PAR,T.MR3	:CHECK IF DRIVE PARITY ERROR
9892	046402	001407				BEQ	5\$:NO, INDICATE ABNORMAL TERMINATION
9893	046404	052737	002000	003042		BIS	#E.DPAR,E.CONT	:SET DRIVE PARITY ERROR
9894	046412	004737	051076			JSR	PC.R.CONT	:INDICATE CONTROLLER ERROR
9895	046416	000137	050050			JMP	I.RTRN	:RETURN

G15

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 189
*RK06 INTERRUPT SERVICE ROUTINE

SEQ 0188

```
9896  
9897 046422 032765 000020 000014 5$: BIT #DRVHRD,P.PRST(R5) ;CHECK IF HARD DRIVE ERROR  
9898 046430 001017 BNE 10$ ;YES GO REPORT ERROR  
9899 046432 032737 020000 003014 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS CYCLING DOWN  
9900 046440 001413 BEQ 10$ ;NO REPORT ERROR  
9901 046442 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING  
9902 046450 113737 003071 003070 MOVB INTMSK,W.TIME ;SET UP 8 SECONDS FOR DRIVE TO CYCLE UP  
9903 046456 013737 003054 003104 MOV W.8SEC,W.DRV  
9904 046464 000137 050050 JMP I.RTRN ;GO RESTORE REGISTERS  
9905  
9906 046470 004737 051052 10$: JSR PC,R.ABNL ;GO REPORT ERROR  
9907 046474 000137 050050 JMP I.RTRN ;GO RESTORE REGISTERS  
9908  
9909 .SBTTL *READ ALL HEADERS INTERRUPT SEQUENCE  
9910  
9911 046500 016237 000000 002770 I.HDAL: MOV RKCS1(R2),T.CS1 ;STORE CS1 TO CHECK CONTROLLER  
9912 ; ERROR  
9913 046506 032737 100000 002770 BIT #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR  
9914 046514 001422 BEQ 5$ ;NO, CHECK FOR ATTENTION  
9915  
9916 046516 005037 003044 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETE  
9917 046522 105037 003070 CLRB W.TIME ;RESET TIMING ON DRIVE  
9918 046526 005037 003104 CLR W.DRV ;CLEAR TIME OUT COUNT  
9919 046532 013765 002770 000016 MOV T.CS1,P.CS1(R5) ;STORE ERROR RKCS1  
9920 046540 004737 050542 JSR PC,I.CST1 ;STORE CONTROLLER REGISTERS  
9921 046544 004037 050066 JSR RD,I.CCLR ;CLEAR CONTROLLER  
9922 046550 050050 I.RTRN ;ERROR RETURN  
9923 046552 004737 051052 JSR PC,R.ABNL ;INDICATE ERROR RETURN  
9924 046556 000137 050050 JMP I.RTRN ;RESTORE REGISTERS  
9925  
9926 046562 016537 000016 003004 5$: MOV RKASOF(R5),T.ASOF ;STORE ATTENTION SUMMARY  
9927 046570 133737 003071 003005 BITB INTMSK,T.ASOF+1 ;CHECK IF DRIVE ATTENTION IS SET  
9928 046576 001410 BEQ 7$ ;NO, CHECK IF READ ALL HEADERS  
9929 046600 005037 003044 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETION  
9930 046604 105037 003070 CLRB W.TIME ;RESET TIMING ON DRIVE  
9931 046610 005037 003104 CLR W.DRV ;CLEAR TIME OUT COUNT  
9932 046614 000137 046234 JMP I.ERRA ;GO REPORT ERROR  
9933  
9934 046620 013701 003060 7$: MOV HDR.AD,R1 ;GET MAIN MEMORY ADDRESS  
9935 046624 016221 000024 MOV RKDB(R2),(R1)+ ;GET FIRST WORD OF HEADER  
9936 046630 016221 000024 MOV RKDB(R2),(R1)+ ;GET SECOND WORD OF HEADER  
9937 046634 016221 000024 MOV RKDB(R2),(R1)+ ;GET THIRD WORD OF HEADER  
9938 046640 010137 003060 MOV R1,HDR.AD ;STORE ADDRESS FOR NEXT HEADER  
9939 046644 016237 000010 002772 MOV RKCS2(R2),T.CS2 ;STORE CS2 TO CHECK FOR DATA LATE  
9940 046652 032737 100000 002772 BIT #DLT,T.CS2 ;CHECK FOR DATA LATE  
9941 046660 001055 BNE 35$ ;YES, REPORT ERROR  
9942 046662 005337 003062 DEC HDR.CT ;DECREMENT NUMBER OF HEADER YET TO READ  
9943 046666 001026 BNE 25$ ;IF NON-ZERO, GO ISSUE NEXT READ HEADER  
9944 046670 005037 003044 CLR O.WAIT ;CLEAR DRIVER WAITING FOR COMMAND COMPLETION  
9945 046674 005037 003104 CLR W.DRV ;CLEAR TIME OUT COUNT FOR THIS DRIVE  
9946 046700 105037 003070 CLRB W.TIME ;CLEAR WATCH DOG TIME ON THIS DRIVE  
9947 046704 012762 000003 000026 MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER FOR SECTOR COUNT  
9948 046712 112737 000001 002770 MOVB #DR.SEL,T.CS1 ;LOAD SELECT COMMAND  
9949 046720 004037 050150 JSR RD,I.ISSU ;GET SECTOR COUNT  
9950 046724 050050 I.RTRN ;ERROR RETURN  
9951 046726 013765 003016 000056 MOV T.MR3,P.B11(R5) ;LOAD SECTOR COUNT
```



H15

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11.27(1006) 05-OCT-76 10:59 PAGE 190
 *READ ALL HEADERS INTERRUPT SEQUENCE

SEQ 0189

```

9952 046734 004737 051064      JSR    PC.R.NORM      ;INDICATE NORMAL TERMINATION
9953 046740 000137 050050      JMP    I.RTRN        ;RESTORE REGISTERS
9954
9955 046744 016562 000002 000020 25$:  MOV    P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
9956 046752 016562 000004 000006      MOV    P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
9957 046760 116565 000007 000017      MOVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
9958 046766 042765 165777 000016      BIC    #↑C<CDT!CFMT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT AND
9959                                     ;DRIVE TYPE
9960 046774 112765 000125 000016      MOVB   #RDHEAD,P.CS1(R5) ;STORE COMMAND ISSUED
9961 047002 016562 000016 000000      MOV    P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
9962 047010 000137 050050      JMP    I.RTRN        ;RESTORE REGISTERS
9963
9964 047014 052737 000400 003042 35$:  BIS    #E.DLT,E.CONT    ;SET DATA LATE WHILE UNLOADING HEADER
9965 047022 004737 051076      JSR    PC.R.CONT     ;REPORT ERROR
9966 047026 000137 050050      JMP    I.RTRN        ;RESTORE REGISTERS
9967
9968                                     .SBTTL *DRIVE ATTENTION SCANNER
9969
9970 047032 016237 000000 002770 I.ATTN: MOV    RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS
9971                                     ;REGISTER 1 FOR COMPARISON
9972 047040 032737 100000 002770      BIT    #CERR,T.CS1   ;CHECK IF CONTROLLER ERROR OCCURRED
9973 047046 001441                                     BEQ    5$            ;NO, CHECK IF ATTENTION
9974                                     ;CHECK IF ANY DATA TRANSFER TYPE ERROR EXISTS
9975                                     BIT    #DLT!WCE!UPE!NEM,T.CS2
9976 047050 032737 164000 002772
9977
9978 047056 001007                                     BNE    1$            ;INDICATE ERROR
9979 047060 016237 000014 003006      MOV    RKER(R2),T.ER ;STORE ERROR REGISTER
9980
9981                                     ;CHECK FOR DATA TRANSFER ERROR TYPE
9982 047066 032737 125700 003006      BIT    #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
9983
9984 047074 001407                                     BEQ    2$            ;NO DATA TRANSFER ERROR
9985
9986 047076 052737 000010 003042 1$:  BIS    #E.UDAT,E.CONT ;SET UNEXPECTED DATA TYPE ERROR
9987 047104 004737 051076      JSR    PC.R.CONT     ;REPORT ERROR
9988 047110 000137 050050      JMP    I.RTRN        ;RESTORE REGISTERS
9989
9990 047114 013704 002772 2$:  MOV    T.CS2,R4      ;SAVE CS2 FOR REGISTER NUMBER
9991 047120 042704 177770      BIC    #↑C<DRVMSK>,R4 ;STRIP OFF JUNK
9992 047124 105037 003070      CLRB   W.TIME       ;CLEAR WATCH DOG TIMER
9993 047130 005037 003104      CLR    W.DRV        ;RESET TIMER VALUE
9994 047134 013705 003102      MOV    PBLKT,R5     ;STORE PARAMETER BLOCK ADDRESS IN R5
9995
9996                                     ;CLEAR DRIVE POSITIONING AND DRIVE POSITIONED FOR DATA TRANSFER
9997                                     ;IN PROGRAM DEVICE STATUS REGISTER
9998 047140 042765 000006 000014      BIC    #DRVPOS!DRVPDT,P.PRST(R5)
9999
10000 047146 000137 046242      JMP    I.ERRC       ;GO REPORT ERROR
10001
10002 047152 032737 040000 002770 5$:  BIT    #DI,T.CS1    ;CHECK IF ANY DRIVE ATTENTION
10003 047160 001002                                     BNE    6$            ;YES, PROCESS INTERRUPT
10004 047162 000137 050050      JMP    I.RTRN        ;RESTORE REGISTERS
10005
10006 047166 016237 000016 003004 6$:  MOV    RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
10007 047174 105737 003005      TSTB   T.ASOF+1     ;CHECK IF ANY ATTENTIONS SET
  
```

10008	047200	001007				BNE	7\$:YES GO PROCESS INTERRUPT
10009	047202	052737	000002	003042		BIS	#E.NOAT,E.CONT	:SET NO ATTENTION IN ATTENTION SUMMARY
10010	047210	004737	051076			JSR	PC,R.CONT	:GO REPORT ERROR
10011	047214	000137	050050			JMP	I.RTRN	:GO RESTORE REGISTERS
10012								
10013	047220	133737	003071	003005	7\$:	BITB	INTMSK,T.ASOF+1	:CHECK IF DESIRED INTERRUPT
10014	047226	001007				BNE	8\$:YES, GO PROCESS IT
10015	047230	052737	000004	003042		BIS	#E.UATT,E.CONT	:SET UNSOLICATED ATTENTION
10016	047236	004737	051076			JSR	PC,R.CONT	:GO REPORT ERROR
10017	047242	000137	050050			JMP	I.RTRN	:GO RESTORE REGISTERS
10018								
10019	047246	013705	003102		8\$:	MOV	PBLKT,R5	:STORE PARAMETER BLOCK TABLE
10020	047252	116504	000000			MOVB	P.DRVN(R5),R4	:STORE DRIVE NUMBER
10021	047256	032765	020000	000014		BIT	#E.UNLD,P.PRST(R5)	:CHECK IF DRIVE UNLOADING
10022	047264	001402				BEQ	11\$:NO, CONTINUE
10023	047266	000137	047750			JMP	I.UNLD	:SERVICE DRIVE IN POSITION AFTER ERROR
10024								
10025	047272	042765	000002	000014	11\$:	BIC	#DRVPOS,P.PRST(R5)	:RESET DRIVE POSITIONING
10026	047300	005062	000026			CLR	RKMR1(R2)	:CLEAR MAINTENANCE REGISTER 1
10027	047304	112737	000001	002770		MOVB	#DR.SEL,T.CS1	:LOAD COMMAND
10028	047312	004037	050150			JSR	RD,I.ISSU	:SELECT DRIVE WITH ATTENTION HIGH
10029	047316	050050				I.RTRN		:ERROR RETURN
10030	047320	013765	003016	000042		MOV	T.MR3,P.B00(R5)	:STORE STATUS BYTE 00 MESS B
10031	047326	032765	000200	000042		BIT	#S.FLT,P.B00(R5)	:CHECK IF DRIVE FAULT
10032	047334	001401				BEQ	12\$:NO, CHECK FOR DRIVE STATUS CHANGE
10033	047336	000461				BR	I.AERR	:PROCESS ERROR
10034								
10035	047340	013765	003014	000040	12\$:	MOV	T.MR2,P.A00(R5)	:STORE MAINTENANCE REGISTER 2
10036	047346	032765	040000	000040		BIT	#S.DSC,P.A00(R5)	:CHECK FOR DRIVE STATUS CHANGE
10037	047354	001004				BNE	13\$:YES, PROCESS DRIVE STATUS CHANGE
10038	047356	052765	004000	000014		BIS	#NODSC,P.PRST(R5)	:SET NO DRIVE STATUS CHANGE
10039	047364	000446				BR	I.AERR	:PROCESS ERROR
10040								
10041	047366	112737	000005	002770	13\$:	MOVB	#DR.CLR,T.CS1	:LOAD COMMAND
10042	047374	004037	050150			JSR	RD,I.ISSU	:CLEAR DRIVE STATUS CHANGE
10043	047400	050050				I.RTRN		:ERROR RETURN
10044	047402	013765	003004	000032		MOV	T.ASOF,P.ASOF(R5)	:STORE ATTENTION SUMMARY
10045	047410	133765	003071	000033		BITB	INTMSK,P.ASOF+1(R5)	:CHECK IF ATTENTION RESET
10046	047416	001407				BEQ	15\$:YES, CONTINUE INTERRUPT PROCESSING
10047	047420	052737	000020	003042		BIS	#E.CLAT,E.CONT	:SET ATTENTION DID NOT RESET
10048								: WITH DRIVE CLEAR
10049	047426	004737	051076			JSR	PC,R.CONT	:FLAG ERROR
10050	047432	000137	050050			JMP	I.RTRN	:RESTORE REGISTERS
10051								
10052	047436	013765	003014	000040	15\$:	MOV	T.MR2,P.A00(R5)	:STORE MAINTENANCE REGISTER 2
10053	047444	032765	040000	000040		BIT	#S.DSC,P.A00(R5)	:CHECK IF DRIVE STATUS CHANGE
10054								: RESET
10055	047452	001404				BEQ	16\$:YES, CONTINUE INTERRUPT PROCESSING
10056	047454	052765	000040	000014		BIS	#DRVDSC,P.PRST(R5)	:SET DRIVE STATUS CHANGE DID NOT CLEAR
10057	047462	000407				BR	I.AERR	:GO PROCESS ERROR
10058								
10059	047464	105037	003070		16\$:	CLRB	W.TIME	:RESET TIMING ON THIS DRIVE
10060	047470	005037	003104			CLR	W.DRV	:CLEAR DRIVE TIMING COUNT
10061	047474	004737	051064			JSR	PC,R.NORM	:REPORT SUCCESSFUL COMMAND COMPLETION
10062	047500	000563				BR	I.RTRN	:RESTORE REGISTERS
10063								


```

10064          .SBTTL *ATTENTION ERROR HANDLER
10065
10066 047502 042765 000004 000014 I.AERR: BIC      #DRVPTD,P.PRST(R5) :RESET POSITIONING IN PROGRESS BECAUSE
10067          : OF DATA TRANSFER
10068 047510 105037 003070          CLR      W.TIME      :CLEAR TIMING FOR THIS DRIVE
10069 047514 005037 003104          CLR      W.DRV      :RESET WATCH-DOG TIME
10070 047520 042765 177741 000016 BIC      #177741,P.CS1(R5) :KEEP COMMAND ISSUED
10071 047526 042737 000036 002770 BIC      #36,T.CS1      :KEEP CURRENT CONTROLLER STATUS
10072 047534 053765 002770 000016 BIS      T.CS1,P.CS1(R5) :MAKE GOOD MESSAGE
10073 047542 013765 002772 000020 MOV      T.CS2,P.CS2(R5) :STORE CONTROLLER REGISTERS
10074 047550 013765 002774 000022 MOV      T.WCR,P.WCR(R5)
10075 047556 013765 002776 000024 MOV      T.BA,P.BAR(R5)
10076 047564 013765 003000 000026 MOV      T.DA,P.DTS(R5)
10077 047572 013765 003002 000030 MOV      T.DC,P.DCYL(R5)
10078 047600 013765 003004 000032 MOV      T.ASOF,P.ASOF(R5)
10079 047606 013765 003006 000034 MOV      T.ER,P.ER(R5)
10080 047614 013765 003010 000036 MOV      T.DS,P.DS(R5)
10081 047622 004037 050624          JSR      RD,I.STAT    :GATHER DRIVE STATUS
10082 047626 050050          I.RTRN    :ERROR RETURN
10083 047630 112737 000005 002770 MOV      #DR.CLR,T.CS1 :LOAD COMMAND
10084 047636 004037 050150          JSR      RD,I.ISSU    :CLEAR DRIVE ERRORS
10085 047642 050050          I.RTRN    :ERROR RETURN
10086 047644 133737 003071 003005 BIT      INTMSK,T.ASOF+1 :CHECK IF ATTENTION RESET
10087 047652 001407          BEQ      2$          :YES, FLAG DRIVE ERROR
10088 047654 052737 000020 003042 BIS      #E.CLAT,E.CONT :SET ATTENTION DID NOT RESET
10089 047662 004737 051076          JSR      PC,R.CONT    :REPORT ERROR
10090 047666 000137 050050          JMP      I.RTRN      :RESTORE REGISTERS
10091
10092 047672 032765 000020 000014 2$: BIT      #DRVHRD,P.PRST(R5) :CHECK IF AWARD DRIVE ERROR
10093 047700 001017          BNE      10$        :YES, REPORT ERROR
10094 047702 032737 020000 003014 BIT      #S.PIP,T.MR2    :CHECK IF DRIVE IS UNLOADING
10095 047710 001413          BEQ      10$        :NO, REPORT ERROR
10096 047712 052765 020000 000014 BIS      #E.UNLD,P.PRST(R5) :SET DRIVE UNLOADING DUE TO ERROR
10097 047720 113737 003071 003070 MOV      INTMSK,W.TIME  :SET TIMING ON THIS DRIVE
10098 047726 013737 003054 003104 MOV      W.BSEC,W.DRV   :LOAD 8 SECONDS FOR CYCLE UP TIME
10099 047734 000137 050050          JMP      I.RTRN      :RESTORE REGISTERS
10100
10101 047740 004737 051052          10$: JSR      PC,R.ABNL    :REPORT ERROR
10102 047744 000137 050050          JMP      I.RTRN      :RESTORE REGISTERS
10103
10104          .SBTTL *ERROR CAUSING DRIVE TO UNLOAD
10105
10106 047750 052765 020000 000014 I.UNLD: BIS      #E.UNLD,P.PRST(R5) :CLEAR DRIVE UNLOADING BECAUSE OF ERROR
10107 047756 112737 000005 002770 MOV      #DR.CLR,T.CS1 :LOAD IN DRIVE CLEAR
10108 047764 004037 050150          JSR      RD,I.ISSU    :GO ISSUE DRIVE CLEAR
10109 047770 050050          I.RTRN    :ERROR RETURN
10110 047772 136437 003071 003005 BIT      INTMSK(R4),T.ASOF+1 :CHECK IF ATTENTION CLEARED
10111 050000 001406          BEQ      15$        :YES, CONTINUE
10112 050002 012737 000020 003042 MOV      #E.CLAT,E.CONT :SET ATTENTION DID NOT RESET
10113 050010 004737 051076          JSR      PC,R.CONT    :REPORT ERROR
10114 050014 000415          BR       I.RTRN      :RESTORE REGISTERS
10115
10116 050016 032737 040000 003014 15$: BIT      #S.DSC,T.MR2    :CHECK IF DRIVE STAU CHANGE RESET
10117 050024 001403          BEQ      20$        :YES, CONTINUE
10118 050026 052765 000040 000014 BIS      #DRVDSK,P.PRST(R5) :SET DRIVE STAU CHANGE DID NOT CLEAR
10119 050034 105037 003070          20$: CLR      W.TIME      :RESET TIMING ON THIS DRIVE
    
```

K15

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 193
*ERROR CAUSING DRIVE TO UNLOAD

SEQ 0192

10120	050040	005037	003104	CLR	W.DRV	:CLEAR TIME COUNT
10121	050044	004737	051052	JSR	PC,R.ABNL	:REPORT ERROR
10122						
10123	050050	012600		I.RTRN: MOV	(SP)+,R0	:RESTORE R0
10124	050052	012601		MOV	(SP)+,R1	:RESTORE R1
10125	050054	012602		MOV	(SP)+,R2	:RESTORE R2
10126	050056	012603		MOV	(SP)+,R3	:RESTORE R3
10127	050060	012604		MOV	(SP)+,R4	:RESTORE R4
10128	050062	012605		MOV	(SP)+,R5	:RESTORE R5
10129	050064	000002		RTI		:RETURN
10130						

.SBTTL *CONTROLLER CLEAR ROUTINE

10131
10132
10133
10134
10135
10136
10137
10138
10139
10140
10141
10142
10143
10144
10145
10146
10147
10148
10149
10150
10151
10152
10153
10154
10155
10156
10157
10158
10159
10160
10161
10162
10163
10164
10165

```

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

```

```

050066 012762 100000 000000 I.CCLR: MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
050074 016237 000000 002770      MOV      RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS REGISTER 1
050102 032737 100000 002770      BIT      #CERR,T.CS1 ;CHECK IF CONTROLLER CLEAR DID
                                ; CLEAR ERROR
050110 001407                    BEQ      5$ ;YES, RETURN TO DRIVER PROCESSING
050112 052737 000001 003042      BIS      #E.CCLR,E.CONT ;SET CLEAR CONTROLLER DID NOT CLEAR ERROR
050120 004737 051076          JSR      PC,R.CONT ;REPORT CONTROLLER ERROR
050124 011000          MOV      (R0),R0 ;SET UP ERROR RETURN
050126 000200          RTS      R0 ;RETURN

050130 012762 000100 000000 5$:  MOV      #IE,RKCS1(R2) ;SET INTERRUPT ENABLE
050136 112737 177777 003064      MOVB     #-1,I.ISRL ;SET INTERRUPT ENABLE ISSUED
050144 005720          TST      (R0)+ ;ADJUST FOR NORMAL RETURN
050146 000200          RTS      R0 ;RETURN

```

M15

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 195

SEQ 0194

*COMMAND ISSUED BY DRIVER SERVICE ROUTINE

.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
*
*****

```

```

10166
10167
10168
10169
10170
10171
10172
10173
10174
10175
10176
10177
10178
10179
10180
10181
10182
10183
10184
10185
10186
10187
10188
10189
10190
10191
10192
10193
10194 050150 013746 002770
10195 050154 005037 002772
10196 050160 116537 000000 002772
10197 050166 013762 002772 000010
10198 050174 116537 000007 002771
10199 050202 142737 177753 002771
10200
10201 050210 013762 002770 000000
10202 050216 105762 000000
10203 050222 100375
10204 050224 004737 050372
10205 050230 032737 100000 002770
10206 050236 001437
10207 050240 032737 001000 002772
10208 050246 001406
10209 050250 052737 100000 003042
10210 050256 004737 051076
10211 050262 000440
10212
10213
10214 050264 032737 024000 002770 2$:
10215 050272 001027
10216 050274 032737 176400 002772
10217 050302 001023
10218 050304 032737 131761 003006
10219 050312 001017
10220
10221 050314 122716 000005

```

```

I.ISSU: MOV      T.CS1,-(SP)      ;STORE COMMAND ISSUED
        CLR      T.CS2          ;CLEAR TEMPORARY CS2
        MOVB     P.DRVN(R5),T.CS2 ;LOAD IN DRIVE NUMBER
        MOV      T.CS2,RKCS2(R2) ;LOAD DRIVE NUMBER FOR COMMAND
        MOVB     P.CS1H(R5),T.CS1+1 ;STORE BITS 8-15 OF CS1
        BICB     #T<B.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,R.CONT      ;REPORT CONTROLLER ERROR
        BR       10$           ;RETURN

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

CMPB     #DR.CLR,(SP) ;CHECK IF CLEAR DRIVE

```

N15

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2 MACY11 27(1006) 05-OCT-76 10:59 PAGE 196
 DZR6NC.P11 05-OCT-76 10:07 *COMMAND ISSUED BY DRIVER SERVICE ROUTINE

SEQ 0195

10222	050320	001003				BNE	3\$:NO, DO NOT SET DRIVE HARD ERROR
10223	050322	052765	000020	000014		BIS	#DRVHRD,P.PRST(R5)	:SET HARD DRIVE ERROR
10224	050330	004037	050066		3\$:	JSR	RO,I.CCLR	:GO ISSUE A CONTROLLER CLEAR
10225	050334	050364				10\$:ERROR RETURN
10226	050336	012762	000100	000000	5\$:	MOV	#IE,RKCS1(R2)	:SET INTERRUPT ENABLE
10227	050344	005726				TST	(SP)+	:ADJUST STACK
10228	050346	005720				TST	(RO)+	:ADJUST RO FOR NORMAL RETURN
10229	050350	000200				RTS	RO	:RETURN
10230								
10231	050352	052737	001000	003042	7\$:	BIS	#E.CERR,E.CONT	:SET CONTROLLER ERROR DURING
10232								: DRIVER SERVICING
10233	050360	004737	051076			JSR	PC,R.CONT	:REPORT ERROR
10234	050364	005726			10\$:	TST	(SP)+	:ADJUST STACK
10235	050366	011000				MOV	(RO),RO	:ADJUST RO FOR ERROR RETURN
10236	050370	000200				RTS	RO	: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 BASE ADDRESS
R5	ADDRESS OF PARAMETER BLOCK

103000
103001
103002
103003
103004
103005
103006
103007
103008
103009
103010
103011
103012

```

050520 042765 177741 000016 I.CSTS: BIC #177741,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FUNCTION
;OF LAST COMMAND ISSUED
050526 042737 000036 002770 BIC #36,T.CS1 ;CLEAR FUNCTION OF CS1 STATUS
050534 053765 002770 000016 BIS T.CS1,P.CS1(R5) ;GENERATE CS1 STATUS INFORMATION
050542 016265 000010 000020 I.CST1: MOV RKCS2(R2),P.CS2(R5) ;STORE COMMAND AND STATUS REGISTER 2
050550 016265 000002 000022 MOV RKWC(R2),P.WCR(R5) ;STORE WORD COUNT REGISTER
050556 016265 000004 000024 MOV RKBA(R2),P.BAR(R5) ;STORE BUS ADDRESS REGISTER
050564 016265 000006 000026 MOV RKDA(R2),P.DTS(R5) ;STORE DESIRED TRACK AND SECTOR
050572 016265 000012 000036 MOV RKDS(R2),P.DS(R5) ;STORE DRIVE STATUS REGISTER
050600 016265 000014 000034 MOV RKER(R2),P.ER(R5) ;STORE ERROR REGISTER
050606 016265 000016 000032 MOV RKASOF(R2),P.ASOF(R5) ;STORE ATTENTION SUMMARY AND
;OFFSET
050614 016265 000020 000030 MOV RKDCYL(R2),P.DCYL(R5) ;STORE CYLINDER ADDRESS
050622 000207 RTS PC ;RETURN

```

.SBTTL *GATHER DRIVE STATUS

* THIS SUBROUTINE WILL BE USED TO GATHER DRIVE STATUS
* BYTE 01, 10, AND 11. IT IS ASSUMED THAT THE DRIVE
* HAS PREVIOUSLY BEEN SEIZED. IT RUNS AT PRIORITY 7.

*CALL JSR RO,I,STAT
* <ADDRESS OF ERROR RETURN>
* RETURN

* THIS ROUTINE ASSUMES THE FOLLOWING REGISTERS CONTAIN:

REGISTER

CONTENTS

R2
R5

RK06 BASE ADDRESS
ADDRESS OF PARAMETER BLOCK

ROUTINES USED:
I.ISSU

1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069

```

050624 012762 000001 000026 I. STAT: MOV #1,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 01
050632 112737 000001 002770 MOVB #DR_SEL,T.CS1 ;LOAD COMMAND
050640 004037 050150 JSR RO,I.ISSU ;GET STATUS BYTES 01
050644 051034 3$ ;ERROR RETURN
050646 013765 003014 000044 MOV T.MR2,P.A01(R5) ;STORE STATUS BYTE 01 MESS A
050654 013765 003016 000046 MOV T.MR3,P.B01(R5) ;STORE STATUS BYTE 01 MESS B
050662 012762 000002 000026 MOV #2,RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 10
050670 112737 000001 002770 MOVB #DR_SEL,T.CS1 ;LOAD COMMAND
050676 004037 050150 JSR RO,I.ISSU ;GET STATUS BYTES 10
050702 051034 3$ ;ERROR RETURN
050704 013765 003014 000050 MOV T.MR2,P.A10(R5) ;STORE STATUS BYTE 10 MESS A
050712 013765 003016 000052 MOV T.MR3,P.B10(R5) ;STORE STATUS BYTE 10 MESS B
050720 012762 000003 000026 MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER
; FOR STATUS BYTE 11
050726 112737 000001 002770 MOVB #DR_SEL,T.CS1 ;LOAD COMMAND
050734 004037 050150 JSR RO,I.ISSU ;GET STATUS BYTES 11
050740 051034 3$ ;ERROR RETURN
050742 013765 003014 000054 MOV T.MR2,P.A11(R5) ;STORE STATUS BYTE 11 MESS A
050750 013765 003016 000056 MOV T.MR3,P.B11(R5) ;STORE STATUS BYTE 11 MESS B
050756 005062 000026 CLR RKMR1(R2) ;LOAD MAINTENANCE REGISTER 1
; FOR STATUS BYTE 00
050762 112737 000001 002770 MOVB #DR_SEL,T.CS1 ;LOAD COMMAND
050770 004037 050150 JSR RO,I.ISSU ;GET STATUS BYTES 00
050774 051034 3$ ;ERROR RETURN
050776 013765 003014 000040 MOV T.MR2,P.A00(R5) ;STORE STATUS BYTE 00 MESS A
051004 013765 003016 000042 MOV T.MR3,P.B00(R5) ;STORE STATUS BYTE 00 MESS B
051012 032737 001000 003016 BIT #S.PAR,T.MR3 ;CHECK IF BAD PARITY DETECTED BY DRIVE
051020 001407 BEQ 5$ ;NO, RETURN NORMALLY

```



```

10379          .SBTTL  *COMMON DRIVER RETURNS
10380
10381 051052 105037 003071 R.ABNL: CLRB INTMSK :INHIBIT FUTURE DRIVE INTERRUPT REPORTING
10382 051056 004777 131754 JSR PC,QA.ABNL :INDICATE ABNORMAL RETURN
10383 051062 000207 RTS PC :RETURN
10384
10385 051064 105037 003071 R.NORM: CLRB INTMSK :INHIBIT FUTURE DRIVE INTERRUPT REPORTING
10386 051070 004777 131740 JSR PC,QA.NORM :INDICATE NORMAL RETURN
10387 051074 000207 RTS PC :RETURN
10388
10389 051076 105037 003071 R.CONT: CLRB INTMSK :INHIBIT FUTURE DRIVE INTERRUPT REPORTING
10390 051102 105037 003070 CLRB W.TIME :RESET WATCH DOG TIMING ON THIS DRIVE
10391 051106 005037 003104 CLR W.DRV :CLEAR TIMING COUNT FOR THIS DRIVE
10392 051112 004777 131722 JSR PC,QA.CONT :INDICATE CONTROLLER ERROR RETURN
10393 051116 000207 RTS PC :RETURN

```

DZR6NC.P11 05-OCT-76 10:07

*COMMAND INITATOR

.SBTTL *COMMAND INITATOR

THIS SUBROUTINE WILL INITIATE ALL COMMANDS AS SPECIFIED BY THE COMMAND FIELD OF THE PARAMETER BLOCK. THE FOLLOWING SPECIAL COMMAND ARE ALSO EXECUTED:

- RELEASE
- CONROLLER CLEAR
- SUBSYSTEM CLEAR
- READ ALL DRIVE STATUS
- READ SPECIFIED HEADER

THE ABOVE COMMANDS ARE TRANSLATED INTO A SEQUENCE OF COMMANDS

*CALL JSR PC.C.INIT
<ADDRESS OF PARAMETER BLOCK>
RETURN

FOR THE SEQUENTIAL OPERATIONS, THE DRIVER WILL LOAD THE LOCATIONS, PBLKT AND INTMSK.

- ROUTINES USED:
- W.WTCH
 - I.CSTS
 - I.STAT
 - I.CCLR

10394
10395
10396
10397
10398
10399
10400
10401
10402
10403
10404
10405
10406
10407
10408
10409
10410
10411
10412
10413
10414
10415
10416
10417
10418
10419

051120	010546		
051122	010446		
051124	010346		
051126	010246		
051130	010146		
051132	010046		
051134	013746	177776	
051140	013737	003032	177776
051146	017605	000016	
051152	062766	000002	000016
051160	016504	000000	
051164	042704	177770	
051170	010537	003102	
051174	116437	003072	003071
051202	116437	003072	003070
051210	013737	003052	003104
051216	013702	003026	

```

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

        MOV RKBAS, R2 ;LOAD R2 WITH RK06 ADDRESS BASE
  
```

```

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

H16

```

10450 051222 042765 075176 000014      BIC      #↑C<DRVUSE!W.WCK!NOCHK!DRPDRV!DTBAIL>,P.PRST(R5)
10451
10452 051230 010500                      MOV      R5,R0                ;STORE PARAMETER BLOCK ADDRESS
10453 051232 062700 000016      ADD      #P.CS1,R0           ;CALCULATE FIRST LOCATION TO BE CLEARED
10454 051236 010501                      MOV      R5,R1                ;STORE PARAMETER BLOCK ADDRESS
10455 051240 062701 000062      ADD      #P.EPAT,R1          ;CALCULATE LAST LOCATION TO BE CLEARED
10456
10457 051244 005020                      1$:    CLR      (R0)+              ;CLEAR RETURN PARAMETER
10458 051246 020001                      CMP      R0,R1                ;CHECK IF FINISHED
10459 051250 101775                      BLOS    1$                    ;NO, CLEAR NEXT RETURN PARAMETER
10460 051252 105037 003064      CLRB    1,ISRL               ;CLEAR RELEASE OR INTERRUPT ISSUED
10461 051256 010465 000020      MOV      R4,P.CS2(R5)         ;STORE DRIVE NUMBER
10462 051262 005062 000026      CLR      RKMRI(R2)           ;CLEAR RK06 MAINTENANCE REGISTER 1
10463 051266 132765 000040 000001  BITB    #BITS,P.CMND(R5)     ;CHECK IF SPECIAL COMMAND
10464 051274 001402                      BEQ     3$                    ;NO, PROCESS
10465 051276 000137 052012      JMP     C.SPEC                ;JUMP TO SPECIAL COMMAND PROCESSOR
10466
10467 051302 122765 000107 000001  3$:    CMPB    #UNLOAD,P.CMND(R5)   ;CHECK IF POSITIONING COMMAND
10468                                     ;START SPINDLE
10469                                     ;RECALIBRATE
10470                                     ;OFFSET
10471                                     ;SEEK
10472                                     ;UNLOAD
10473
10474 051310 101174                      BHI     25$                   ;NO, DRIVE COMMAND
10475                                     ;SELECT DRIVE
10476                                     ;PACK ACKNOWLEDGE
10477                                     ;CLEAR
10478
10479 051312 122765 000117 000001  CMPB    #SEEK,P.CMND(R5)     ;CHECK IF DATA TRANSFER
10480 051320 103540                      BLO     20$                   ;YES, DATA TRANSFER COMMAND
10481                                     ;READ DATA
10482                                     ;WRITE DATA
10483                                     ;READ HEADER
10484                                     ;WRITE HEADER
10485                                     ;WRITE CHECK
10486
10486 051322 016562 000020 000010  MOV      P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
10487 051330 052765 000002 000014  BIS     #DRVPOS,P.PRST(R5)   ;SET DRIVE POSITIONING
10488 051336 005037 003044      CLR      0,WAIT              ;CLEAR WAIT FOR COMMAND
10489 051342 122765 000117 000001  CMPB    #SEEK,P.CMND(R5)     ;CHECK IF SEEK
10490 051350 001007                      BNE     5$                    ;NO, CHECK FOR OFFSET
10491 051352 016562 000002 000020  MOV      P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
10492 051360 016562 000004 000006  MOV      P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
10493 051366 000431                      BR      8$                    ;GO ISSUE COMMAND
10494
10495 051370 122765 000115 000001  5$:    CMPB    #OFFSET,P.CMND(R5)   ;CHECK IF OFFSET
10496 051376 001007                      BNE     6$                    ;NO, CHECK FOR UNLOAD
10497 051400 116565 000006 000032  MOVB    P.OFST(R5),P.ASOF(R5) ;STORE OFFSET
10498 051406 016562 000032 000016  MOV      P.ASOF(R5),RKASOF(R2) ;LOAD OFFSET REGISTER
10499 051414 000416                      BR      8$                    ;GO ISSUE COMMAND
10500
10501 051416 122765 000111 000001  6$:    CMPB    #SRTSPL,P.CMND(R5)   ;CHECK IF START SPINDLE
10502 051424 001003                      BNE     7$                    ;NO, CHECK IF RECAL
10503 051426 013737 003056 003104  MOV      W.MIN,W.DRV         ;LOAD WATCH DOG TIME FOR 1 MINUTE
10504 051434 122765 000113 000001  7$:    CMPB    #RECAL,P.CMND(R5)   ;CHECK IF RECAL
10505 051442 001003                      BNE     8$                    ;NO, CONTINUE
    
```

```

10506 051444 013737 003054 003104      MOV      W.BSEC,W.DRV      ;LOAD RECAL TIME FOR 8 SECONDS
10507 051452 116565 000007 000017 8$:      MOVVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
10508 051460 042765 165777 000016      BIC     #↑C<CFMT!CDT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT
10509                                     ; AND DRIVE TYPE
10510 051466 116565 000001 000016      MOVVB   P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
10511 051474 042765 000200 000014      BIC     #W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
10512 051502 032765 000400 000014      BIT     #NOCHK,P.PRST(R5) ;CHECK IN NO CHECK MODE
10513 051510 001533 000000 000000      BEQ     30$ ;NO, SKIP CLEAR OF INTERRUPT ENABLE
10514 051512 042765 000100 000016      BIC     #IE,P.CS1(R5) ;CLEAR INTERRUPT ENABLE
10515 051520 016562 000016 000000      MOV     P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
10516 051526 004737 045470 000000 10$:     JSR     PC.W.WTCH ;CALL WATCH DOG TIMER
10517 051532 016237 000000 002770      MOV     RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS REGISTER 1
10518 051540 032737 000200 002770      BIT     #RDY,T.CS1 ;WAIT FOR READY
10519 051546 001767 000000 000000      BEQ     10$
10520 051550 032737 100000 002770      BIT     #CERR,T.CS1 ;CHECK FOR ERROR
10521 051556 001011 000000 000000      BNE     15$ ;YES, GIVE NORMAL RETURN
10522 051560 004737 045470 000000 11$:     JSR     PC.W.WTCH ;CALL WATCH DOG TIMER
10523 051564 016237 000016 003004      MOV     RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
10524 051572 133737 003071 003005      BITB   INTMSK,T.ASOF+1 ;CHECK IF INTERRUPT HAS OCCURRED
10525 051600 001767 000000 000000      BEQ     11$ ;WAIT FOR DRIVE INTERRUPT
10526 051602 105037 003070 000000 15$:     CLRB   W.TIME ;RESET TIMING ON THIS DRIVE
10527 051606 005037 003104 000000      CLR    W.DRV ;CLEAR DRIVE TIMING COUNT
10528 051612 004737 051064 000000      JSR     PC.R.NORM ;INDICATE COMMAND IS FINISHED
10529 051616 000137 052772 000000      JMP     C.RTRN ;RESTORE REGISTERS
10530
10531 051622 016562 000010 000004 20$:     MOV     P.BALO(R5),RKBA(R2) ;LOAD BUS ADDRESS REGISTER
10532 051630 016562 000012 000002      MOV     P.WC(R5),RKWC(R2) ;LOAD WORD COUNT REGISTER
10533 051636 016562 000002 000020      MOV     P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
10534 051644 016562 000004 000006      MOV     P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK NUMBER
10535 051652 122765 000131 000001      CMPB   #WRTCHK,P.CMND(R5) ;CHECK IF WRITE CHECK COMMAND
10536 051660 001010 000000 000000      BNE     25$ ;NO, GO ISSUE THE COMMAND
10537 051662 032765 000200 000014      BIT     #W.WCK,P.PRST(R5) ;CHECK IF WRITE COMMAND SHOULD BE ISSUED
10538 051670 001404 000000 000000      BEQ     25$ ;NO, GO ISSUE THE COMMAND
10539 051672 012765 000123 000016      MOV     #WRDATA,P.CS1(R5) ;ISSUE WRITE COMMAND
10540 051700 000406 000000 000000      BR     26$ ;GO ISSUE COMMAND
10541
10542 051702 116565 000001 000016 25$:     MOVVB   P.CMND(R5),P.CS1(R5) ;MOVE COMMAND INTO CS1
10543 051710 042765 000200 000014      BIC     #W.WCK,P.PRST(R5) ;RESET WRITE FOR WRITE CHECK
10544 051716 116565 000007 000017 26$:     MOVVB   P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
10545 051724 142765 177750 000017      BICB   #↑C<B.CFMT!B.CDT!A.BA16!B.BA17>,P.CS1+1(R5) ;CLEAR ALL BITS EXCEPT
10546                                     ; FORMAT, DRIVE TYPE, AND BUS ADDRESS
10547                                     ; BITS 16-17
10548 051732 010537 003044 000000      MOV     R5,0.WAIT ;LOAD WAITING FOR COMMAND
10549 051736 032765 100000 000014      BIT     #DTBAII,P.PRST(R5) ;CHECK IF INHIBIT BUS ADDRESS INCREMENT
10550 051744 001403 000000 000000      BEQ     27$ ;NO, LOAD CS2
10551 051746 052765 000020 000020      BIS     #BAI,P.CS2(R5) ;SET INHIBIT BUS ADDRESS INCREMENT
10552 051754 016562 000020 000010 27$:     MOV     P.CS2(R5),RKCS2(R2) ;LOAD CS2
10553 051762 032765 000400 000014      BIT     #NOCHK,P.PRST(R5) ;CHECK IN NO CHECK MODE
10554 051770 001403 000000 000000      BEQ     30$ ;NO, SKIP CLEAR OF INTERRUPT ENABLE
10555 051772 042765 000100 000016      BIC     #IE,P.CS1(R5) ;CLEAR INTERRUPT ENABLE
10556 052000 016562 000016 000000 30$:     MOV     P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
10557 052006 000137 052772 000000      JMP     C.RTRN ;RESTORE REGISTERS
10558
10559                                     .SBTTL  *SPECIAL COMMAND PROCESSING
10560
10561 052012 122765 000141 000001  C.SPEC: CMPB   #RDSTAT,P.CMND(R5) ;CHECK IF READ DRIVE STATUS

```

J16

MD-11-DZ6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZ6NC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 205
 *SPECIAL COMMAND PROCESSING

SEQ 0204

10562	052020	001132			BNE	10\$:NO, PROCESS OTHER COMMANDS	
10563	052022	016562	000020	000010	MOV	P.CS2(R5),RKCS2(R2)	:LOAD CS2 FOR COMMAND	
10564	052030	116565	000007	000017	MOV	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1	
10565	052036	042765	165777	000016	BIC	#C<CFMT!COT>,P.CS1(R5)	:CLEAR ALL BITS EXCEPT FORMAT AND DRIVE TYPE	
10567	052044	112765	000001	000016	MOV	#DR.SEL,P.CS1(R5)	:STORE COMMAND	
10568	052052	016562	000016	000000	MOV	P.CS1(R5),RKCS1(R2)	:ISSUE COMMAND	
10569	052060	004737	045470		JSR	PC.W.WTCH	:CALL WATCH-DOG TIMER	
10570	052064	016265	000000	000016	MOV	RKCS1(R2),P.CS1(R5)	:STORE COMMAND AND STATUS REG. 1	
10571	052072	032765	000200	000016	BIT	#RDY,P.CS1(R5)	:WAIT FOR READY	
10572	052100	001767			BEG	2\$		
10573	052102	004737	050542		JSR	PC.I.CST1	:STORE CONTROLLER REGISTERS	
10574	052106	016265	000034	000040	MOV	RKMR2(R2),P.A00(R5)	:STORE STATUS BYTE 00 MESSAGE A	
10575	052114	016265	000036	000042	MOV	RKMR3(R2),P.B00(R5)	:STORE STATUS BYTE 00 MESSAGE B	
10576	052122	032765	100000	000016	BIT	#CERR,P.CS1(R5)	:CHECK IF CONTROLLER ERROR	
10577	052130	001436			BEG	6\$:NO, GATHER DRIVE STATUS	
10578	052132	105037	003070		CLRB	W.TIME	:RESET WATCH DOG TIMING ON THIS DRIVE	
10579	052136	005037	003104		CLR	W.DRV	:CLEAR WATCH DOG COUNT	
10580	052142	032765	000400	000014	BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE	
10581	052150	001043			BNE	8\$:YES, INDICATE NORMAL RETURN	
10582	052152	032765	001000	000020	BIT	#MDS,P.CS2(R5)	:CHECK IF MULTIPLE DRIVE SELECT	
10583	052160	001043			BNE	9\$:YES, INDICATE CONTROLLER ERROR	
10584	052162	004037	050066		JSR	RO,I.CCLR	:CLEAR ERROR	
10585	052166	052772			C.RTRN		:ERROR RETURN	
10586	052170	032765	010400	000020	BIT	#NED!UFE,P.CS2(R5)	:CHECK IF NON-EXISTENT DRIVE OR UNIT FIELD ERROR	
10587	052176	001007			BNE	5\$:REPORT ERROR	
10588	052200	032765	000001	000036	BIT	#DRA,P.DS(R5)	:CHECK IF DRIVE AVAILIABLE	
10589	052206	001003			BNE	5\$:YES, REPORT ERROR	
10590	052210	052765	010000	000014	BIS	#DRVSZD,P.PRST(R5)	:INDICATE DRIVE IS SEIZED BY OTHER PORT	
10591	052216	004737	051052		JSR	PC.R.ABNL	:INDICATE ABNORMAL RETURN	
10592	052222	000137	052772		JMP	C.ATRN	:RESTORE REGISTERS	
10594	052226	004037	050624		JSR	RO,I.STAT	:GATHER DRIVE STATUS	
10595	052232	052772			C.RTRN		:ERROR RETURN	
10596	052234	105037	003070		CLRB	W.TIME	:STOP WATCH-DOG TIMING ON DRIVE	
10597	052240	005037	003104		CLR	W.DRV	:RESET WATCH-DOG TIME	
10598	052244	032765	000400	000014	BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE	
10599	052252	001402			BEG	8\$:NO, REPORT ERROR	
10600	052254	005062	000000		CLR	RKCS1(R2)	:CLEAR INTERRUPT ENABLE	
10601	052260	004737	051064		JSR	PC.R.NORM	:REPORT COMMAND COMPLETE	
10602	052264	000137	052772		JMP	C.ATRN	:RESTORE REGISTERS	
10604	052270	052737	100000	003042	BIS	#E.MDS,E.CONT	:SET MULTIPLE DRIVE SELECT	
10605	052276	004737	051076		JSR	PC.R.CONT	:INDICATE CONTROLLER ERROR	
10606	052302	000137	052772		JMP	C.ATRN		
10607								
10608	052306	122765	000140	000001	10\$:	CMPB	#RELAS,P.CMND(R5)	:CHECK IF RELEASE COMMAND
10609	052314	001040			BNE	13\$:NO, CHECK IF READ ALL HEADERS	
10610	052316	010537	003044		MOV	RS,O.WAIT	:STORE PARAMETER BLOCK ADDRESS IN WAIT FOR COMMAND	
10611								
10612	052322	052765	000010	000020	BIS	#RLS,P.CS2(R5)	:SET RELEASE BIT	
10613	052330	016562	000020	000010	MOV	P.CS2(R5),RKCS2(R2)	:LOAD CS2 FOR DESELECT	
10614	052336	112737	000001	003064	MOV	#I.I.ISRL	:SET FLAG FOR RELEASE COMMAND	
10615	052344	116565	000007	000017	MOV	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1	
10616	052352	042765	165777	000016	BIC	#C<CFMT!COT>,P.CS1(R5)	:CLEAR ALL BITS EXCEPT FORMAT AND DRIVE TYPE	
10617								

K16

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 206
 *SPECIAL COMMAND PROCESSING

SEQ 0205

10618	052360	112765	000101	000016		MOVB	#SELDRV,P.CS1(R5) ;STORE COMMAND
10619	052366	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
10620	052374	001403				BEQ	11\$;NO, DO NOT RESET INTERRUPT ENABLE
10621	052376	042765	000100	000016		BIC	#IE,P.CS1(R5) ;RESET INTERRUPT ENABLE
10622	052404	016562	000016	000000	11\$:	MOV	P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
10623	052412	000137	052772			JMP	C.RTRN ;RESTORE REGISTERS
10624							
10625	052416	122765	000164	000001	13\$:	CMPB	#RDALHD,P.CMND(R5) ;CHECK IF READ ALL HEADERS
10626	052424	001053				BNE	30\$;NO, CHECK IF CONTROLLER CLEAR
10627	052426	010537	003044			MOV	R5,0.WAIT ;SET WAITING FOR COMMAND COMPLETION
10628	052432	016537	000010	003060		MOV	P.BALO(R5),HDR.AD ;LOAD HEADER ADDRESS
10629	052440	132765	000020	000007		BITB	#B.CFMT,P.CS1H(R5) ;CHECK IF 22 SECTOR FORMANT
10630	052446	001404				BEQ	14\$;YES, LOAD 22 IN HEADER COUNT
10631	052450	012737	000024	003062		MOV	#20.,HDR.CT ;LOAD 20 IN SECTOR COUNT
10632	052456	000403				BR	22\$;GO ISSUE READ HEADER COMMAND
10633							
10634	052460	012737	000026	003062	14\$:	MOV	#22.,HDR.CT ;LOAD 22 IN SECTOR COUNT
10635	052466	016562	000002	000020	22\$:	MOV	P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
10636	052474	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2) ;LOAD TRACK NUMBER
10637	052502	016562	000020	000010		MOV	P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
10638	052510	116565	000007	000017		MOVB	P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
10639	052516	042765	165777	000016		BIC	#1C<CFMT!CDT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT DRIVE TYPE
10640							AND FORMAT
10641	052524	112765	000125	000016		MOVB	#RDHEAD,P.CS1(R5) ;STORE READ HEADER COMMAND
10642	052532	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
10643	052540	001027				BNE	34\$;YES, INDICATE ILLEGAL DRIVER COMMAND
10644	052542	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
10645	052550	000137	052772			JMP	C.RTRN ;RESTORE REGISTERS
10646							
10647	052554	122765	000176	000001	30\$:	CMPB	#CONCLR,P.CMND(R5) ;CHECK IF CONTROLLER CLEAR
10648	052562	001012				BNE	32\$;NO, CHECK IF SUBSYSTEM CLEAR
10649	052564	004037	050066			JSR	RD,I.CCLR ;CLEAR CONTROLLER
10650	052570	052772				C.RTRN	ERROR RETURN
10651	052572	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
10652	052600	001472				BEQ	40\$;NO, INDICATE NORMAL RETURN
10653	052602	005062	000000			CLR	RKCS1(R2) ;RESET INTERRUPT ENABLE
10654	052606	000467				BR	40\$;INDICATE NORMAL RETURN
10655							
10656	052610	122765	000177	000001	32\$:	CMPB	#SUBCLR,P.CMND(R5) ;CHECK IF SUBSYSTEM CLEAR
10657	052616	001406				BEQ	36\$;YES, CLEAR SUBSYSTEM
10658	052620	052737	000100	003042	34\$:	BIS	#E.ILLD,E.CONT ;SET ILLEGAL DRIVER COMMAND
10659	052626	004737	051076			JSR	PC,R.CONT ;REPORT ERROR
10660	052632	000457				BR	C.RTRN ;RESTORE REGISTERS
10661							
10662	052634	012762	000040	000010	36\$:	MOV	#SCLR,RKCS2(R2) ;ISSUE SUBSYSTEM CLEAR
10663	052642	016265	000000	000016		MOV	RKCS1(R2),P.CS1(R5) ;STORE COMMAND AND STATUS REGISTER 1
10664	052650	032765	100000	000016		BIT	#CERR,P.CS1(R5) ;CLEAR IF CONTROLLER ERROR RESET
10665	052656	001406				BEQ	37\$;NO, FINISH COMMAND
10666	052660	052737	000001	003042		BIS	#BIT0,E.CONT ;SET CLEAR SUBSYSTEM DID NOT CLEAR
10667							CONTROLLER ERROR
10668	052666	004737	051076			JSR	PC,R.CONT ;REPORT ERROR
10669	052672	000437				BR	C.RTRN ;RESTORE REGISTERS
10670							
10671	052674	013746	003050		37\$:	MOV	W.MILI,-(SP) ;LOAD 16 MILI-SECOND COUNT FOR ATTENTION
10672							TO DISAPPEAR
10673	052700	016265	000000	000016	38\$:	MOV	RKCS1(R2),P.CS1(R5) ;STORE CS1

```

10674 052706 032765 040000 000016 BIT #DI,P.CS1(R5) ;CHECK IF ATTENTIONS CLEARED
10675 052714 001411 BEQ 39$ ;YES, FINISH COMMAND
10676 052716 005316 DEC (SP) ;DECREMENT 16 MILLISECOND COUNT
10677 052720 001367 BNE 38$ ;CHECK DRIVE INTERRUPT AGAIN
10678 052722 005726 TST (SP)+ ;ADJUST STACK
10679 052724 052737 000040 003042 BIS #E.SCLR,E.CONT ;SET SUBSYSTEM CLEAR DID NOT CLEAR
10680 ;DRIVE ATTENTIONS
10681 052732 004737 051076 JSR PC,R.CONT ;REPORT ERROR
10682 052736 000415 BR C.RTRN ;RESTORE REGISTER
10683
10684 052740 005726 39$: TST (SP)+ ;ADJUST STACK
10685 052742 032765 000400 000014 BIT #NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
10686 052750 001010 BNE C.RTRN ;YES, RESTORE REGISTERS
10687 052752 112737 177777 003064 MOVB #-1,I.ISRL ;SET INTERRUPT ENABLE SET
10688 052760 012762 000100 000000 MOV #IE,RKCS1(R2) ;SET INTERRUPT ENABLE
10689 052766 004737 051064 40$: JSR PC,R.NORM ;INDICATE NORMAL TERMINATION
10690
10691 052772 012637 177776 C.RTRN: MOV (SP)+,PS ;RESTORE PSW
10692 052776 012600 MOV (SP)+,R0 ;RESTORE R0
10693 053000 012601 MOV (SP)+,R1 ;RESTORE R1
10694 053002 012602 MOV (SP)+,R2 ;RESTORE R2
10695 053004 012603 MOV (SP)+,R3 ;RESTORE R3
10696 053006 012604 MOV (SP)+,R4 ;RESTORE R4
10697 053010 012605 MOV (SP)+,R5 ;RESTORE R5
10698 053012 000207 RTS PC ;RETURN
10699
10700 .SBTTL OCTAL TO BINARY CONVERSION ROUTINE
10701
10702 ;*****
10703 ;*
10704 ;* THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
10705 ;* WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL
10706 ;* IT WILL GENERATE TWO BINARY WORDS PLACING THE LOW 16 BITS
10707 ;* ON THE STACK AND THE HIGH 16 BITS IN LOCATION $HI0CT.
10708 ;*
10709 ;*CALL
10710 ;* MOV <ADDRESS OF ASCII STRING>,-(SP)
10711 ;* JSR PC,OCTBIN
10712 ;* <ADDRESS OF ERROR RETURN>
10713 ;* RETURN
10714 ;*
10715 ;*****
10716 053014 010046 OCTBIN: MOV R0,-(SP) ;SAVE R0
10717 053016 010146 MOV R1,-(SP) ;SAVE R1
10718 053020 010246 MOV R2,-(SP) ;SAVE R2
10719 053022 016600 000010 MOV 10(SP),R0 ;GET ADDRESS OF ASCII STRING
10720 053026 005001 CLR R1 ;CLEAR DATA WORDS
10721 053030 005002 CLR R2
10722 053032 112046 2$: MOVB (R0)+,-(SP) ;PICK THIS CHARACTER
10723 053034 001423 BEQ 3$ ;IF ZERO GET OUT
10724 053036 121627 000054 CMPB (SP),#', ;CHECK IF COMMA
10725 053042 001420 BEQ 3$ ;IF COMMA GET OUT
10726 053044 122716 000060 CMPB #'0,(SP) ;MAKE SURE THIS CHARACTER IS
10727 053050 003030 BGT 4$ ; AN OCTAL DIGIT
10728 053052 122716 000067 CMPB #'7,(SP)
10729 053056 002425 BLT 4$

```


M16

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE, 208
 OCTAL TO BINARY CONVERSION ROUTINE

SEQ 0207

10730	053060	006301		ASL	R1	; *2
10731	053062	006102		ROL	R2	
10732	053064	006301		ASL	R1	; *4
10733	053066	006102		ROL	R2	
10734	053070	006301		ASL	R1	; *8
10735	053072	006102		ROL	R2	
10736	053074	042716	177770	BIC	#1C7,(SP)	;STRIP THE ASCII JUNK
10737	053100	062601		ADD	(SP)+,R1	;ADD THIS DIGIT
10738	053102	000753		BR	2\$;LOOP
10739	053104	005726		3\$: TST	(SP)+	;CLEAN PARTIAL FROM STACK
10740	053106	010166	000010	MOV	R1,10(SP)	;SAVE RESULT
10741	053112	010237	053146	MOV	R2,\$HI OCT	
10742	053116	012602		MOV	(SP)+,R2	;RESTORE R2
10743	053120	012601		MOV	(SP)+,R1	;RESTORE R1
10744	053122	012600		MOV	(SP)+,R0	;RESTORE R0
10745	053124	062716	000002	ADD	#2,(SP)	;ADJUST RETURN
10746	053130	000207		RTS	PC	;RETURN
10747						
10748	053132	005726		4\$: TST	(SP)+	;CLEAN UP PARTIAL FROM STACK
10749	053134	012602		MOV	(SP)+,R2	;RESTORE R2
10750	053136	012601		MOV	(SP)+,R1	;RESTORE R1
10751	053140	012600		MOV	(SP)+,R0	;RESTORE R0
10752	053142	013616		MOV	2(SP)+,(SP)	;PUT ADDRESS OF ERROR ROUTINE ON STACK
10753	053144	000207		RTS	PC	;GO PROCESS ERROR
10754	053146	000000		\$HI OCT: .WORD	0	;HIGH ORDER BITS GO HERE
10755				.SBTTL	RANDOM NUMBER GENERATOR ROUTINE	
10756						
10757						
10758						
10759						
10760						
10761						
10762						
10763						
10764						
10765						
10766	053150			\$RAND:		
10767	053150	010046		MOV	R0,-(SP)	;PUSH R0 ON STACK
10768	053152	010146		MOV	R1,-(SP)	;PUSH R1 ON STACK
10769	053154	010246		MOV	R2,-(SP)	;PUSH R2 ON STACK
10770	053156	013700	053250	MOV	\$LONUM,R0	;SET R0 WITH LOW
10771	053162	013701	053246	MOV	\$HINUM,R1	;SET R1 WITH HIGH
10772	053166	012702	177771	MOV	#-7,R2	;SET SHIFT COUNT
10773	053172	006300		1\$: ASL	R0	;SHIFT R0 LEFT AND
10774	053174	006101		ROL	R1	;ROTATE CARRY INTO R1 AND
10775	053176	005202		INC	R2	;CHECK FOR DONE
10776	053200	001374		BNE	1\$;CONTINUE SHIFT LOOP
10777	053202	063700	053250	ADD	\$LONUM,R0	;ADD NUMBER TO MAKE X 129
10778	053206	005501		ADC	R1	;PROPOGATE CARRY
10779	053210	063701	053246	ADD	\$HINUM,R1	;ADD NUMBER TO MAKE X 129
10780	053214	062700	001057	ADD	#1057,R0	;ADD LOW CONSTANT
10781	053220	005501		ADC	R1	;PROPOGATE CARRY
10782	053222	062701	047401	ADD	#47401,R1	;ADD HIGH CONSTANT
10783	053226	010037	053250	MOV	R0,\$LONUM	;SAVE R0
10784	053232	010137	053246	MOV	R1,\$HINUM	;SAVE R1
10785	053236	012602		MOV	(SP)+,R2	;POP STACK INTO R2

```

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

10788 053240 012601
10789 053242 012600
10790 053244 000207
10791 053246 176543
10792 053250 123456

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

*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
*

10800 053252 105737 001157
10801 053256 100002
10802 053260 000000
10803 053262 000430
10804 053264 010046
10805 053266 017600 000002
10806 053272 122737 000001 001340
10807 053300 001011
10808 053302 132737 000100 001341
10809 053310 001405
10810 053312 010037 053322
10811 053316 004737 055750
10812 053322 000000
10813 053324 132737 000040 001341
10814 053332 001003
10815 053334 112046
10816 053336 001005
10817 053340 005726
10818 053342 012600
10819 053344 062716 000002
10820 053350 000002
10821 053352 122716 000011
10822 053356 001430
10823 053360 122716 000200
10824 053364 001006
10825 053366 005726
10826 053370 104401
10827 053372 001315
10828 053374 105037 053530
10829 053400 000755
10830 053402 004737 053464
10831 053406 123726 001156
10832 053412 001350
10833 053414 013746 001154

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

```

10864 053420 105366 000001 7$: DECB 1(SP) ::AND THE NULL CHAR.
10865 053424 002770 BLT 6$ ::DOES A NULL NEED TO BE TYPED?
10866 053426 004737 053464 JSR PC,$TYPEC ::BR IF NO--GO POP THE NULL OFF OF STACK
10867 053432 105337 053530 DECB $CHARCNT ::GO TYPE A NULL
10868 053436 000770 BR 7$ ::DO NOT COUNT AS A COUNT
::LOOP

```

:HORIZONTAL TAB PROCESSOR

```

10869 053440 112716 000040 8$: MOVB #' (SP) ::REPLACE TAB WITH SPACE
10870 053444 004737 053464 9$: JSR PC,$TYPEC ::TYPE A SPACE
10871 053448 132737 000007 053530 BITB #7,$CHARCNT ::BRANCH IF NOT AT
10872 053452 001372 BNE 9$ ::TAB STOP
10873 053456 005726 TST (SP)+ ::POP SPACE OFF STACK
10874 053460 000724 BR 2$ ::GET NEXT CHARACTER
10875 053464 105777 125460 $TYPEC: TSTB $STPB ::WAIT UNTIL PRINTER IS READY
10876 053470 100375 BPL $TYPEC
10877 053472 116677 000002 125452 MOVB 2(SP), $STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
10878 053500 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
10879 053506 001003 BNE 1$ ::BRANCH IF NO
10880 053510 105037 053530 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
10881 053514 000406 BR $TYPEX ::EXIT
10882 053516 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
10883 053520 001402 BEQ $TYPEX ::BRANCH IF YES
10884 053524 105227 INCB (PC)+ ::COUNT THE CHARACTER
10885 053530 000000 $CHARCNT: WORD 0 ::CHARACTER COUNT STORAGE
10886 053532 000207 $TYPEX: RTS PC

```

```

:*****
:SBTTL DOUBLE-PRECISION MULTIPLY SUBROUTINE
:* SUBROUTINE TO MULTIPLY TWO DOUBLE PRECISION INTEGERS
:* USES ALL REGISTERS (R0-R5)
:*
:* ENTER WITH JSR PC,M.DPIM
:* MULTIPLIER IN R2-R3
:* MULTIPLICAND IN R4-R5
:* PRODUCT RETURNED IN R0-R1-R2-R3
:*****

```

```

10884 053534 005000 M.DPIM: CLR R0 ;CLEAR HI ORDER WORDS
10885 053536 005001 CLR R1
10886 053540 012746 000041 MOV #41,-(SP) ;MOVE 33 (DEC) TO COUNTER
10887 053544 006000 M.DP01: ROR R0
10888 053546 006001 ROR R1
10889 053550 006002 ROR R2 ;SHIFT TO ADD
10890 053552 006003 ROR R3
10891 053554 103003 BCC M.DP02 ;NO CARRY NO ADD
10892 053556 060501 ADD R5,R1
10893 053560 005500 ADC R0 ;ADD DOUBLE PRECISION TO OBTAIN NEW PARTIAL
10894 053562 060400 ADD R4,R0 ;PRODUCT
10895 053564 005316 M.DP02: DEC $SP ;DECREMENT COUNTER
10896 053566 001366 BNE M.DP01
10897 053570 005726 TST (SP)+ ;REMOVE THE COUNTER

```

10899	053572	000207	
10900			
10901			
10902			
10903			
10904			
10905			
10906			
10907			
10908			
10909			
10910			
10911			
10912			
10913	053574	012746	000040
10914	053600	010446	
10915	053602	010546	
10916	053604	005466	000002
10917	053610	005416	
10918	053612	005666	000002
10919	053616	061601	
10920	053620	005500	
10921	053622	066600	000002
10922	053626	103445	
10923	053630	005046	
10924	053632	006103	
10925	053634	006102	
10926	053636	006101	
10927	053640	006100	
10928	053642	005716	
10929	053644	001410	
10930	053646	005016	
10931	053650	066601	000002
10932	053654	005500	
10933	053656	005516	
10934	053660	066600	000004
10935	053664	000404	
10936	053666	060501	
10937	053670	005500	
10938	053672	005516	
10939	053674	060400	
10940	053676	005516	
10941	053700	005716	
10942	053702	001401	
10943	053704	005203	
10944	053706	005366	000006
10945	053712	003347	
10946	053714	006003	
10947	053716	103404	
10948	053720	060501	
10949	053722	005500	
10950	053724	060400	
10951	053726	000241	
10952	053730	006103	
10953	053732	062706	000010

```

RTS    PC

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

```

EO1

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 212
 DOUBLE-PRECISION DIVIDE SUBROUTINE

SEQ 0211

```

10954 053736 000242
10955 053740 000207
10956 053743 062706 000006
10957 053746 000262
10958 053750 000207
  
```

```

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

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

```

;*****
;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
;UNSIGNED OCTAL ASCII NUMBER.
;CALL
;*      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
;*      JSR      PC, @#$DB20      ;; CALL THE ROUTINE
;*      RETURN                      ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
  
```

```

10973 053752 104407
10974 053754 016601 000002
10975 053760 012705 054071
10976 053764 012704 000014
10977 053770 012703 177770
10978 053774 012100
10979 053776 012101
10980 054000 005002
10981 054002 110245
10982 054004 010002
10983 054006 005304
10984 054010 003007
10985 054012 001405
10986 054014 005205
10987 054016 010566 000002
10988 054022 104410
10989 054024 000207
10990 054026 006203
10991 054030 006001
10992 054032 006000
10993 054034 006001
10994 054036 006000
10995 054040 006001
10996 054042 006000
10997 054044 040302
10998 054046 062702 000060
10999 054052 000753
11000 054054 000016
  
```

```

$DB20:  SAVREG                      ;; SAVE ALL REGISTERS
        MOV      2(SP),R1          ;; PICKUP THE POINTER TO LOW WORD
        MOV      #SOCTVL+13.,R5   ;; POINTER TO DATA TABLE
        MOV      #12.,R4          ;; DO ELEVEN CHARACTERS
        MOV      #7,R3           ;; MASK
        MOV      (R1)+,R0         ;; LOWER WORD
        MOV      (R1)+,R1         ;; HIGH WORD
        CLR      R2              ;; TERMINATOR
1$:     MOVB     R2, -(R5)         ;; PUT CHARACTER IN DATA TABLE
        MOV      R0,R2           ;; GET THIS DIGIT
        DEC     R4              ;; COUNT THIS CHARACTER
        BGT     3$,             ;; BR IF NOT THE LAST DIGIT
        BEQ     2$,             ;; BR IF IT IS THE LAST DIGIT
        INC     R5              ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
        MOV     R5, 2(SP)        ;; ASCII CHAR. & PUT IT ON THE STACK
        RESREG                    ;; RESTORE ALL REGISTERS
        RTS      PC              ;; RETURN TO USER
2$:     ASR     R3                ;; POSITION THE MASK FOR THE LAST DIGIT
3$:     ROR     R1                ;; POSITION THE BINARY NUMBER FOR
        ROR     R0                ;; THE NEXT OCTAL DIGIT
        ROR     R1
        ROR     R0
        ROR     R1
        ROR     R0
        BIC     R3,R2            ;; MASK OUT ALL JUNK
        ADD     #'0,R2          ;; MAKE THIS CHAR. ASCII
        BR      1$              ;; GO PUT IT IN THE DATA TABLE
SOCTVL: .BLKB 14.              ;; RESERVE DATA TABLE
.SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
  
```

```

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

```

11007
11008
11009
        MOV     #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
        JSR     PC, @#$DB20
  
```

F01

MD-11-DZREN-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZRENC.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 213
DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

SEQ 0212

```

11010          ;*      RETURN          ;;THE FIRST ADDRESS OF ASCIZ
11011          ;;IS ON THE STACK
11012
11013
11014          054072 104407          $DB2D: SAVREG          ;;SAVE REGISTERS
11015          054074 016602 000002      MOV      2(SP),R2          ;;PICKUP THE DATA POINTER
11016          054100 012700 054252      MOV      #SDECVL,R0          ;;GET ADDRESS OF "SDECVL" STRING
11017          054104 010066 000002      MOV      R0,2(SP)          ;;PUT ADDRESS OF ASCIZ STRING ON STACK
11018          054110 012201          MOV      (R2)+,R1          ;;PICKUP THE BINARY NUMBER
11019          054112 012202          MOV      (R2)+,R2
11020          054114 012737 000012 054170      MOV      #10,4$          ;;SET UP TO DO 10 CONVERSIONS
11021          054122 012704 054202      MOV      #STNPWR,R4          ;;ADDRESS OF TEN POWER
11022          054126 012705 054204      MOV      #STNPWR+2,R5
11023          054132 005003          1$: CLR      R3          ;;CLEAR PARTIAL
11024          054134 161401          2$: SUB      (R4),R1          ;;SUBTRACT TEN POWER
11025          054136 005602          SBC      R2
11026          054140 161502          SUB      (R5),R2
11027          054142 002402          BLT      3$          ;;BR IF TEN POWER TO LARGE
11028          054144 005203          INC      R3          ;;ADD 1 TO PARTIAL
11029          054146 000772          BR       2$          ;;LOOP
11030          054150 062401          3$: ADD      (R4)+,R1          ;;RESTORE SUBTRACTED VALUE
11031          054152 005502          ADC      R2
11032          054154 062402          ADD      (R4)+,R2
11033          054156 022525          CMP      (R5)+,(R5)+          ;;MOVE TO NEXT TEN POWER
11034          054160 052703 000060      BIS      #'0,R3          ;;CHANGE PARTIAL TO ASCII
11035          054164 110320          MOV      R3,(R0)+          ;;SAVE IT
11036          054166 005327          DEC      (PC)+          ;;DONE?
11037          054170 000000          4$: .WORD 0
11038          054172 001357          BNE      1$          ;;BR IF NO
11039          054174 105020          CLRB    (R0)+          ;;TERMINATOR
11040          054176 104410          RESREG          ;;RESTORE REGISTERS
11041          054200 000207          RTS      PC          ;;RETURN
11042          054202 145000          STNPWR: 145000          ;;1.0E09
11043          054204 035632          35632
11044          054206 160400          160400          ;;1.0E08
11045          054210 002765          2765
11046          054212 113200          113200          ;;1.0E07
11047          054214 000230          230
11048          054216 041100          041100          ;;1.0E06
11049          054220 000017          17
11050          054222 103240          103240          ;;1.0E05
11051          054224 000001          1
11052          054226 023420          23420          ;;1.0E04
11053          054230 000000          0
11054          054232 001750          1750          ;;1.0E03
11055          054234 000000          0
11056          054236 000144          144          ;;1.0E02
11057          054240 000000          0
11058          054242 000012          12          ;;1.0E01
11059          054244 000000          0
11060          054246 000001          1          ;;1.0E00
11061          054250 000000          0
11062          054252 000014          SDECVL: .BLKB 12          ;;RESERVE STORAGE FOR ASCIZ STRING
11063          .SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
11064
11065          ;;*****

```

GO1

```

11066      *THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
11067      *LEADING NUMBERS.
11068      *CALL
11069      *      MOV      #NUMADR,-(SP)      ;;FIRST ADDRESS OF ASCIZ STRING
11070      *      JSR      PC,2#$$SUPRS
11071
11072
11073      $$SUPRS: MOV      RO,-(SP)          ;;SAVE RO
11074      MOV      4(SP),RO                ;;PICKUP THE POINTER
11075      1$:      TSTB      (RO)            ;;TERMINATEOR?
11076      BEQ      2$                    ;;BR IF YES
11077      CMPB      #'0,(RO)+            ;;IS THIS AN ASCII "0" ?
11078      BEQ      1$                    ;;BR IF YES
11079      2$:      DEC      RO              ;;BACKUP BY "1"
11080      MOV      RO,3$                  ;;SAVE FOR TYPING
11081      TYPE
11082      3$:      .WORD      0              ;;GO TYPE
11083      MOV      (SP)+,RO                ;;ASCIZ POINTER GOES HERE
11084      MOV      (SP)+,(SP)              ;;RESTORE RO
11085      RTS      PC                      ;;RESTORE THE STACK
11086      .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
11087
11088      *-----
11089      *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
11090      *OCTAL (ASCII) NUMBER AND TYPE IT.
11091      *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
11092      *CALL:
11093      *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
11094      *      TYPOS
11095      *      .BYTE      N                ;;CALL FOR TYPEOUT
11096      *      .BYTE      M                ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
11097      *                                          ;;M=1 OR 0
11098      *                                          ;;1=TYPE LEADING ZEROS
11099      *                                          ;;0=SUPPRESS LEADING ZEROS
11100      *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
11101      *$TYPOS OR $TYPOC
11102      *CALL:
11103      *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
11104      *      TYPON                      ;;CALL FOR TYPEOUT
11105      *
11106      *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
11107      *CALL:
11108      *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
11109      *      TYPOC                      ;;CALL FOR TYPEOUT
11110
11111      $TYPOS: MOV      2(SP),-(SP)        ;;PICKUP THE MODE
11112      MOV      1(SP),$OFILL              ;;LOAD ZERO FILL SWITCH
11113      MOV      (SP)+,$SOMODE+1          ;;NUMBER OF DIGITS TO TYPE
11114      ADD      #2,(SP)                  ;;ADJUST RETURN ADDRESS
11115      BR      $TYPON
11116      $TYPOC: MOV      #1,$OFILL          ;;SET THE ZERO FILL SWITCH
11117      MOV      #6,$SOMODE+1            ;;SET FOR SIX(6) DIGITS
11118      $TYPON: MOV      #5,$SOCNT        ;;SET THE ITERATION COUNT
11119      MOV      R3,-(SP)                  ;;SAVE R3
11120      MOV      R4,-(SP)                  ;;SAVE R4
11121      MOV      R5,-(SP)                  ;;SAVE R5
    
```

```

11066
11067
11068
11069
11070
11071
11072
11073 054266 010046
11074 054270 016600 000004
11075 054274 105710
11076 054276 001403
11077 054300 122720 000060
11078 054304 001773
11079 054306 005300
11080 054310 010037 054316
11081 054314 104401
11082 054316 000000
11083 054320 012600
11084 054322 012616
11085 054324 000207
11086
11087
11088
11089
11090
11091
11092
11093
11094
11095
11096
11097
11098
11099
11100
11101
11102
11103
11104
11105
11106
11107
11108
11109
11110
11111 054326 017646 000000
11112 054332 116637 000001 054551
11113 054340 112637 054553
11114 054344 062716 000002
11115 054350 000406
11116 054352 112737 000001 054551
11117 054360 112737 000006 054553
11118 054366 112737 000005 054550
11119 054374 010346
11120 054376 010446
11121 054400 010546
    
```

H01

MD-11-DZREN-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZREN.C.F11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 215
 BINARY TO OCTAL (ASCII) AND TYPE

SEQ 0214

111122	054402	113704	054553		MOVB	\$OMODE+1,R4	;;GET THE NUMBER OF DIGITS TO TYPE
111123	054406	005404			NEG	R4	
111124	054410	062704	000006		ADD	#6,R4	;;SUBTRACT IT FOR MAX. ALLOWED
111125	054414	110437	054552		MOVB	R4,\$OMODE	;;SAVE IT FOR USE
111126	054420	113704	054551		MOVB	\$OFILL,R4	;;GET THE ZERO FILL SWITCH
111127	054424	016605	000012		MOV	12(SP),R5	;;PICKUP THE INPUT NUMBER
111128	054430	005003			CLR	R3	;;CLEAR THE OUTPUT WORD
111129	054432	006105		1\$:	ROL	R5	;;ROTATE MSB INTO "C"
111130	054434	000404			BR	3\$;;GO DO MSB
111131	054436	006105		2\$:	ROL	R5	;;FORM THIS DIGIT
111132	054440	006105			ROL	R5	
111133	054442	006105			ROL	R5	
111134	054444	010503			MOV	R5,R3	
111135	054446	006103		3\$:	ROL	R3	;;GET LSB OF THIS DIGIT
111136	054450	105337	054552		DECB	\$OMODE	;;TYPE THIS DIGIT?
111137	054454	100016			BPL	7\$;;BR IF NO
111138	054456	042703	177770		BIC	#177770,R3	;;GET RID OF JUNK
111139	054462	001002			BNE	4\$;;TEST FOR 0
111140	054464	005704			TST	R4	;;SUPPRESS THIS 0?
111141	054466	001403			BEQ	5\$;;BR IF YES
111142	054470	005204		4\$:	INC	R4	;;DON'T SUPPRESS ANYMORE 0'S
111143	054472	052703	000060		BIS	#'0,R3	;;MAKE THIS DIGIT ASCII
111144	054476	052703	000040		BIS	#',R3	;;MAKE ASCII IF NOT ALREADY
111145	054502	110337	054546		MOVB	R3,8\$;;SAVE FOR TYPING
111146	054506	104401	054546		TYPE	8\$;;GO TYPE THIS DIGIT
111147	054512	105337	054550		DECB	\$OCNT	;;COUNT BY 1
111148	054516	003347			BGT	2\$;;BR IF MORE TO DO
111149	054520	002402			BLT	6\$;;BR IF DONE
111150	054522	005204			INC	R4	;;INSURE LAST DIGIT ISN'T A BLANK
111151	054524	000744			BR	2\$;;GO DO THE LAST DIGIT
111152	054526	012605		6\$:	MOV	(SP)+,R5	;;RESTORE R5
111153	054530	012604			MOV	(SP)+,R4	;;RESTORE R4
111154	054532	012603			MOV	(SP)+,R3	;;RESTORE R3
111155	054534	016666	000002 000004		MOV	2(SP),4(SP)	;;SET THE STACK FOR RETURNING
111156	054542	012616			MOV	(SP)+,(SP)	
111157	054544	000002			RTI		;;RETURN
111158	054546	000		8\$:	.BYTE	0	;;STORAGE FOR ASCII DIGIT
111159	054547	000			.BYTE	0	;;TERMINATOR FOR TYPE ROUTINE
111160	054550	000		\$OCNT:	.BYTE	0	;;OCTAL DIGIT COUNTER
111161	054551	000		\$OFILL:	.BYTE	0	;;ZERO FILL SWITCH
111162	054552	000000		\$OMODE:	.WORD	0	;;NUMBER OF DIGITS TO TYPE
111163				.SBTTL	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE		
111164					*****		
111165					*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT		
111166					*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE		
111167					*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED		
111168					*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE		
111169					*REPLACED WITH SPACES.		
111170					*CALL:		
111171					* MOV NUM,-(SP) ;;PUT THE BINARY NUMBER ON THE STACK		
111172					* TYPDS ;;GO TO THE ROUTINE		
111173							
111174							
111175	054554			\$TYPDS:			
111176	054554	010046			MOV	R0,-(SP)	;;PUSH R0 ON STACK
111177	054556	010146			MOV	R1,-(SP)	;;PUSH R1 ON STACK

I01

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 216
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0215

11178	054560	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK
11179	054562	010346		MOV	R3,-(SP)	::PUSH R3 ON STACK
11180	054564	010546		MOV	R5,-(SP)	::PUSH R5 ON STACK
11181	054566	012746	020200	MOV	#20200,-(SP)	::SET BLANK SWITCH AND SIGN
11182	054572	016605	000020	MOV	20(SP),R5	::GET THE INPUT NUMBER
11183	054576	100004		BPL	1\$::BR IF INPUT IS POS.
11184	054600	005405		NEG	R5	::MAKE THE BINARY NUMBER POS.
11185	054602	112766	000055 000001	MOVB	#'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
11186	054610	005000		CLR	R0	::ZERO THE CONSTANTS INDEX
11187	054612	012703	054770	MOV	#\$DBLK,R3	::SETUP THE OUTPUT POINTER
11188	054616	112723	000040	MOVB	#',(R3)+	::SET THE FIRST CHARACTER TO A BLANK
11189	054622	005002		CLR	R2	::CLEAR THE BCD NUMBER
11190	054624	016001	054760	MOV	\$DTBL(R0),R1	::GET THE CONSTANT
11191	054630	160105		SUB	R1,R5	::FORM THIS BCD DIGIT
11192	054632	002402		BLT	4\$::BR IF DONE
11193	054634	005202		INC	R2	::INCREASE THE BCD DIGIT BY 1
11194	054636	000774		BR	3\$	
11195	054640	060105		ADD	R1,R5	::ADD BACK THE CONSTANT
11196	054642	005702		TST	R2	::CHECK IF BCD DIGIT=0
11197	054644	001002		BNE	5\$::FALL THROUGH IF 0
11198	054646	105716		TSTB	(SP)	::STILL DOING LEADING 0'S?
11199	054650	100407		BMI	7\$::BR IF YES
11200	054652	106316		ASLB	(SP)	::MSD?
11201	054654	103003		BCC	6\$::BR IF NO
11202	054656	116663	000001 177777	MOVB	1(SP),-1(R3)	::YES--SET THE SIGN
11203	054664	052702	000060	BIS	#'0,R2	::MAKE THE BCD DIGIT ASCII
11204	054670	052702	000040	BIS	#',R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
11205	054674	110223		MOVB	R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
11206	054676	005720		TST	(R0)+	::JUST INCREMENTING
11207	054700	020027	000010	CMP	R0,#10	::CHECK THE TABLE INDEX
11208	054704	002746		BLT	2\$::GO DO THE NEXT DIGIT
11209	054706	003002		BGT	8\$::GO TO EXIT
11210	054710	010502		MOV	R5,R2	::GET THE LSD
11211	054712	000764		BR	6\$::GO CHANGE TO ASCII
11212	054714	105726		TSTB	(SP)+	::WAS THE LSD THE FIRST NON-ZERO?
11213	054716	100003		BPL	9\$::BR IF NO
11214	054720	116663	177777 177776	MOVB	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
11215	054726	105013		CLRB	(R3)	::SET THE TERMINATOR
11216	054730	012605		MOV	(SP)+,R5	::POP STACK INTO R5
11217	054732	012603		MOV	(SP)+,R3	::POP STACK INTO R3
11218	054734	012602		MOV	(SP)+,R2	::POP STACK INTO R2
11219	054736	012601		MOV	(SP)+,R1	::POP STACK INTO R1
11220	054740	012600		MOV	(SP)+,R0	::POP STACK INTO R0
11221	054742	104401	054770	TYPE	\$DBLK	::NOW TYPE THE NUMBER
11222	054746	016666	000002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
11223	054754	012616		MOV	(SP)+,(SP)	
11224	054756	000002		RTI		::RETURN TO USER
11225	054760	023420		\$DTBL:	10000.	
11226	054762	001750			1000.	
11227	054764	000144			100.	
11228	054766	000012			10.	
11229	054770	000004		\$DBLK:	.BLKW 4	
11230				.SBTTL	ERROR HANDLER ROUTINE	
11231						
11232						
11233						

::*****
 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,

```

11234 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
11235 ;*AND GO TO TYPERR ON ERROR
11236 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
11237 ;*SW15=1 HALT ON ERROR
11238 ;*SW13=1 INHIBIT ERROR TYPEOUTS
11239 ;*SW10=1 BELL ON ERROR
11240 ;*SW09=1 LOOP ON ERROR
11241 ;*CALL
11242 ;* ERROR N ;:ERROR=EMT AND N=ERROR ITEM NUMBER
11243 $ERROR:
11244 055000 7$: INCB $ERFLG ;:SET THE ERROR FLAG
11245 055000 105237 001103 BEQ 7$ ;:DON'T LET THE FLAG GO TO ZERO
11246 055004 001775 MOV $TSTNM,@DISPLAY ;:DISPLAY TEST NUMBER AND ERROR FLAG
11247 055006 013777 001102 124126 BIT #BIT10,@SWR ;:BELL ON ERROR?
11248 055014 032777 002000 124116 BEQ 1$ ;:NO - SKIP
11249 055022 001402 TYPE $BELL ;:RING BELL
11250 055024 104401 001310 1$: INC $ERTTL ;:COUNT THE NUMBER OF ERRORS
11251 055030 005237 001112 MOV (SP),$ERRPC ;:GET ADDRESS OF ERROR INSTRUCTION
11252 055034 011637 001116 SUB #2,$ERRPC
11253 055040 162737 000002 001116 MOVB @ERRPC,$ITEMB ;:STRIP AND SAVE THE ERROR ITEM CODE
11254 055046 117737 124044 001114 BIT #BIT13,@SWR ;:SKIP TYPEOUT IF SET
11255 055054 032777 020000 124056 BNE 20$ ;:SKIP TYPEOUTS
11256 055062 001004 JSR PC,TYPERR ;:GO TO USER ERROR ROUTINE
11257 055064 004737 041300 TYPE ,SRLF
11258 055070 104401 001315 20$: CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
11259 055074 BNE 2$ ;:NO SKIP APT ERROR REPORT
11260 055074 122737 000001 001340 MOVB $ITEMB,21$ ;:SET ITEM NUMBER AS ERROR NUMBER
11261 055102 001007 JSR PC,$ATY4 ;:REPORT FATAL ERROR TO APT
11262 055104 113737 001114 055116 21$: .BYTE 0
11263 055112 004737 055760 .BYTE 0
11264 055116 000 .BR 22$ ;:APT ERROR LOOP
11265 055117 000 22$: TST @SWR ;:HALT ON ERROR
11266 055120 000777 2$: BPL 3$ ;:SKIP IF CONTINUE
11267 055122 005777 124012 HALT ;:HALT ON ERROR!
11268 055126 100001 3$: BIT #BIT09,@SWR ;:LOOP ON ERROR SWITCH SET?
11269 055130 000000 BEQ 4$ ;:BR IF NO
11270 055132 032777 001000 124000 4$: MOV $LPERR,(SP) ;:FUDGE RETURN FOR LOOPING
11271 055140 001402 TST $ESCAPE ;:CHECK FOR AN ESCAPE ADDRESS
11272 055142 013716 001110 BEQ 5$ ;:BR IF NONE
11273 055146 005737 001306 MOV $ESCAPE,(SP) ;:FUDGE RETURN ADDRESS FOR ESCAPE
11274 055152 001402 5$: CMP #SENDAD,@#42 ;:ACT-11 AUTO-ACCEPT?
11275 055154 013716 001306 BNE 6$ ;:BRANCH IF NO
11276 055160 5$: HALT ;:YES
11277 055160 022737 023600 000042 6$: RTI ;:RETURN
11278 055166 001001 .SBTTL TTY INPUT ROUTINE
11279 055170 000000 ;:*****
11280 055172 .ENABL LSB
11281 055172 000002 .DSABL LSB
11282
11283
11284
11285
11286
11287
11288
11289

```

```

11290          ;:*****
11291          ;:THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
11292          ;:CALL:
11293          ;:      RDCHR          ;: INPUT A SINGLE CHARACTER FROM THE TTY
11294          ;:      RETURN HERE ;: CHARACTER IS ON THE STACK
11295          ;:                  ;: WITH PARITY BIT STRIPPED OFF
11296          ;:
11297          ;:
11298 055174 011646 $RDCHR: MOV      (SP), -(SP) ;: PUSH DOWN THE PC
11299 055176 016666 000004 000002 MOV      4(SP), 2(SP) ;: SAVE THE PS
11300 055204 105777 123734 1$: TSTB   @STKS ;: WAIT FOR
11301 055210 100375 BPL      1$ ;: A CHARACTER
11302 055212 117766 123730 000004 MOVB   @STKB, 4(SP) ;: READ THE TTY
11303 055220 042766 177600 000004 BIC    #177, 4(SP) ;: GET RID OF JUNK IF ANY
11304 055226 026627 000004 000023 CMP     4(SP), #23 ;: IS IT A CONTROL-S?
11305 055234 001013 BNE     3$ ;: BRANCH IF NO
11306 055236 105777 123702 2$: TSTB   @STKS ;: WAIT FOR A CHARACTER
11307 055242 100375 BPL     2$ ;: LOOP UNTIL ITS THERE
11308 055244 117746 123676 MOVB   @STKB, -(SP) ;: GET CHARACTER
11309 055250 042716 177600 BIC    #177, (SP) ;: MAKE IT 7-BIT ASCII
11310 055254 022627 000021 CMP     (SP)+, #21 ;: IS IT A CONTROL-Q?
11311 055260 001366 BNE     2$ ;: IF NOT DISCARD IT
11312 055262 000750 BR      1$ ;: YES, RESUME
11313 055264 026627 000004 000140 3$: CMP     4(SP), #140 ;: IS IT UPPER CASE?
11314 055272 002407 BLT     4$ ;: BRANCH IF YES
11315 055274 026627 000004 000175 CMP     4(SP), #175 ;: IS IT A SPECIAL CHAR?
11316 055302 003003 BGT     4$ ;: BRANCH IF YES
11317 055304 042766 000040 000004 BIC    #40, 4(SP) ;: MAKE IT UPPER CASE
11318 055312 000002 4$: RTI ;: GO BACK TO USER
11319 055314 052536 005015 000 $CNTLU: .ASCIZ /↑U<15><12> ;: CONTROL "U"
11320 055321 136 006507 000012 $CNTLG: .ASCIZ /↑G<15><12> ;: CONTROL "G"
11321 055326 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
11322 055334 020075 000
11323 055337 040 047040 053505 $MNEW: .ASCIZ / NEW = /
11324 055344 036440 000040
11325
11326
11327
11328          .SBTTL POWER DOWN AND UP ROUTINES
11329
11330          ;POWER DOWN ROUTINE
11331 055350 012737 055362 000024 $PWRDN: MOV      #PWRUP, PWRVEC ;: SET VECTOR FOR POWER UP
11332 055356 000000 HALT ;: HANG UP
11333 055360 000776 BR      -2
11334
11335          ;POWER UP ROUTINE
11336 055362 005037 055434 $PWRUP: CLR      $PWRCT ;: WAIT LOOP FOR TTY TO COME UP
11337 055366 005237 055434 4$: INC     $PWRCT
11338 055372 001375 BNE     4$
11339 055374 012737 055350 000024 MOV     #PWRDN, PWRVEC ;: SET VECTOR FOR POWER DOWN
11340 055402 012737 000340 000026 MOV     #PR7, PWRVEC+2 ;: RE-ESTABLISH POWER AND TRAP PRIORITIES
11341 055410 012737 000340 000036 MOV     #PR7, TRAPVEC+2
11342 055416 012706 001100 MOV     #STACK, SP ;: RE-INITIALIZE THE STACK
11343 055422 104401 055436 TYPE    ,PWRMSG ;: TYPE "POWER FAILED"
11344 055426 000005 RESET ;: CLEAR THE UNIBUS
11345 055430 000177 123452 JMP     @SLPADR ;: RESTART THE CURRENT TEST
    
```

LO1

```

11346 055434 000000 $PWRC: .WORD 0
11347 055436 005015 047520 042527 PWRMSG: .ASCIZ <15><12>/POWER FAILED/<15><12>
11348 055444 020122 040506 046111
11349 055452 042105 005015 000
11350 055460 .EVEN
11351
11352
11353
11354
11355
11356
11357
11358 055460 105737 001103 SCOPE1: TSTB $ERFLG ;SEE IF AN ERROR HAS OCCURRED
11359 055464 001406 BEQ 6$ ;BR IF NOT
11360 055466 032777 001000 123444 BIT #BIT9, $SWR ;SEE IF LOOP ON ERROR DESIRED
11361 055474 001402 BEQ 6$ ;BR IF NOT
11362 055476 013716 001110 MOV $LPERR, (SP) ;SET ERROR LOOP ADDRESS ON STACK
11363 055502 000002 6$: RTI ;RETURN
11364
11365
11366
11367
11368
11369
11370
11371
11372
11373
11374
11375
11376
11377
11378
11379
11380 055504 $SCOPE:
11381 055504 005737 001304 TST $TIMES ;CHECK CURRENT ITERATION NUMBER
11382 055510 001404 BEQ $XTSTR ;BR IF 0, TO SKIP CHECK OF SWR BIT 14
11383 055512 032777 040000 123420 1$: BIT #BIT14, $SWR ;LOOP ON PRESENT TEST?
11384 055520 001101 BNE $OVER ;YES IF SW14=1
11385
11386 055522 000416 $XTSTR: BR 6$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
11387 ;#####START OF CODE FOR THE XOR TESTER##### ;THIS INSTRUCTION TO A "NOP" (NOP=240)
11388 055524 013746 000004 MOV @#ERRVEC, -(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
11389 055530 012737 055550 000004 MOV #5$, @#ERRVEC ;SET FOR TIMEOUT
11390 055536 005737 177060 TST @#177060 ;TIME OUT ON XOR?
11391 055542 012637 000004 MOV (SP)+, @#ERRVEC ;RESTORE THE ERROR VECTOR
11392 055546 000450 BR $SVLAD ;GO TO THE NEXT TEST
11393 055550 022626 5$: CMP (SP)+, (SP)+ ;CLEAR THE STACK AFTER A TIME OUT
11394 055552 012637 000004 MOV (SP)+, @#ERRVEC ;RESTORE THE ERROR VECTOR
11395 055556 000413 BR 7$ ;LOOP ON THE PRESENT TEST
11396 055560 6$; ;#####END OF CODE FOR THE XOR TESTER#####
11397 055560 105737 001103 2$: TSTB $ERFLG ;HAS AN ERROR OCCURRED?
11398 055564 001421 BEQ 3$ ;BR IF NO
11399 055566 123737 001115 001103 CMPB $ERMAX, $ERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
11400 055574 101015 BHI 3$ ;BR IF NO
11401 055576 032777 001000 123334 BIT #BIT09, $SWR ;LOOP ON ERROR?

```

```

11402 055604 001404          BEQ      4$          ;;BR IF NO
11403 055606 013737 001110 001106 7$:  MOV     $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
11404 055614 000443          BR      $OVER
11405 055616 105037 001103          4$:  CLRB   $ERFLG      ;;ZERO THE ERROR FLAG
11406 055622 005037 001304          CLR     $TIMES     ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
11407 055626 000412          BR      1$          ;;ESCAPE TO THE NEXT TEST
11408 055630 032777 004000 123302 3$:  BIT     #BIT11,$SWR  ;;INHIBIT ITERATIONS?
11409 055636 001006          BNE     1$          ;;BR IF YES
11410 055640 005237 001104          INC     $ICNT      ;;INCREMENT ITERATION COUNT
11411 055644 023737 001304 001104  CMP     $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
11412 055652 002024          BGE     $OVER      ;;BR IF MORE ITERATION REQUIRED
11413 055654 012737 000001 001104 1$:  MOV     #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
11414 055662 013737 055740 001304  MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
11415 055670 105237 001102  $SVLAD: INCB   $STSTNM    ;;COUNT TEST NUMBERS
11416 055674 113737 001102 001324  MOVB   $STSTNM,$STSTN ;;SET TEST NUMBER IN APT MAILBOX
11417 055702 011637 001106          MOV     (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
11418 055706 011637 001110          MOV     (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
11419 055712 005037 001306          CLR     $ESCAPE    ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
11420 055716 112737 000001 001115  MOVB   #1,$ERMAX   ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
11421 055724 013777 001102 123210 $OVER: MOV     $STSTNM,$DISPLAY ;;DISPLAY TEST NUMBER
11422 055732 013716 001106          MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
11423 055736 000002          RTI
11424 055740 003720          $MXCNT: 2000.      ;;FIXES PS
11425          .SBTTL  APT COMMUNICATIONS ROUTINE  ;;MAX. NUMBER OF ITERATIONS
11426
11427          ;;*****
11428 055742 112737 000001 056206 $ATY1: MOVB   #1,$FFLG  ;;TO REPORT FATAL ERROR
11429 055750 112737 000001 056204 $ATY3: MOVB   #1,$MFLG  ;;TO TYPE A MESSAGE
11430 055756 000403          BR      $ATYC
11431 055760 112737 000001 056206 $ATY4: MOVB   #1,$FFLG  ;;TO ONLY REPORT FATAL ERROR
11432 055766          $ATYC:
11433 055766 010046          MOV     RD,-(SP)   ;;PUSH RD ON STACK
11434 055770 010146          MOV     R1,-(SP)   ;;PUSH R1 ON STACK
11435 055772 105737 056204          TSTB   $MFLG      ;;SHOULD TYPE A MESSAGE?
11436 055776 001450          BEQ     5$          ;;IF NOT: BR
11437 056000 122737 000001 001340  CMPB   #APTENV,$ENV ;;OPERATING UNDER APT?
11438 056006 001031          BNE     3$          ;;IF NOT: BR
11439 056010 132737 000100 001341  BITB   #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
11440 056016 001425          BEQ     3$          ;;IF NOT: BR
11441 056020 017600 000004          MOV     @4(SP),RO  ;;GET MESSAGE ADDR.
11442 056024 062766 000002 000004  ADD     #2,4(SP)   ;;BUMP RETURN ADDR.
11443 056032 005737 001320 1$:  TST     $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
11444 056036 001375          BNE     1$          ;;IF NOT: WAIT
11445 056040 010037 001334          MOV     RO,$MSGAD  ;;PUT ADDR IN MAILBOX
11446 056044 105720 2$:  TSTB   (RO)+      ;;FIND END OF MESSAGE
11447 056046 001376          BNE     2$
11448 056050 163700 001334          SUB     $MSGAD,RO  ;;SUB START OF MESSAGE
11449 056054 006200          ASR     RO        ;;GET MESSAGE LNTH IN WORDS
11450 056056 010037 001336          MOV     RO,$MSGLGT ;;PUT LENGTH IN MAILBOX
11451 056062 012737 000004 001320  MOV     #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
11452 056070 000413          BR      5$
11453 056072 017637 000004 056116 3$:  MOV     @4(SP),4$  ;;PUT MSG ADDR IN JSR LINKAGE
11454 056100 062766 000002 000004  ADD     #2,4(SP)   ;;BUMP RETURN ADDRESS
11455 056106 013746 177776          MOV     177776,-(SP) ;;PUSH 177776 ON STACK
11456 056112 004737 053252          JSR     PC,$TYPE   ;;CALL TYPE MACRO
11457 056116 000000          4$:  .WORD  0

```

```

11458 056120
11459 056120 105737 056206
11460 056124 001416
11461 056126 005737 001340
11462 056132 001413
11463 056134 005737 001320
11464 056140 001375
11465 056142 017637 000004 001322
11466 056150 062766 000002 000004
11467 056156 005237 001320
11468 056162 105037 056206
11469 056166 105037 056205
11470 056172 105037 056204
11471 056176 012601
11472 056200 012600
11473 056202 000207
11474 056204 000
11475 056205 000
11476 056206 000
11477 056210
11478 000200
11479 000001
11480 000100
11481 000040
11482
11483
11484
11485
11486
11487
11488
11489
11490
11491
11492
11493
11494
11495
11496
11497
11498 056210 010046
11499 056212 010146
11500 056214 010246
11501 056216 010346
11502 056220 013746 000004
11503 056224 013746 000006
11504 056230 010600
11505
11506 056232 104400
11507 056234 012637 000006
11508 056240 012701 003776
11509 056244 105727
11510 056246 000200
11511 056250 100062
11512 056252 012737 056410 000004
11513 056260 005737 177572

5$:
10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
      BEQ 12$ ;; IF NOT: BR
      TST $ENV ;; RUNNING UNDER APT?
      BEQ 12$ ;; IF NOT: BR
11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
      BNE 11$ ;; IF NOT: WAIT
      MOV @4(SP), $FATAL ;; GET ERROR #
      ADD #2, 4(SP) ;; BUMP RETURN ADDR.
12$: INC $MSGTYPE ;; TELL APT TO TAKE ERROR
      CLRB $FFLG ;; CLEAR FATAL FLAG
      CLRB $LFLG ;; CLEAR LOG FLAG
      CLRB $MFLG ;; CLEAR MESSAGE FLAG
      MOV (SP)+, R1 ;; POP STACK INTO R1
      MOV (SP)+, R0 ;; POP STACK INTO R0
      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 ROUTINE TO SIZE MEMORY

*****
*CALL:
* JSR PC, $SIZE
* RETURN
*$LSTAD WILL CONTAIN:
* WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
* WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
*$KT11 IS THE MEMORY MANAGEMENT KEY
*$BIT07 = 0 DON'T USE MEMORY MANAGEMENT
* MUST BE SETUP BEFORE THE CALL
*$BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
* DETERMINED BY ROUTINE

$SIZE: MOV R0, -(SP) ;; SAVE R0 ON THE STACK
      MOV R1, -(SP) ;; SAVE R1 ON THE STACK
      MOV R2, -(SP) ;; SAVE R2 ON THE STACK
      MOV R3, -(SP) ;; SAVE R3 ON THE STACK
      MOV @#ERRVEC, -(SP) ;; SAVE PRESENT ERROR VECTOR PS & PC
      MOV @#ERRVEC+2, -(SP)
      MOV SP, R0 ;; SAVE THE STACK POINTER
;; SET THE ERRVEC PS TO THE PRESENT PS
      TRAP ;; PUSH OLD PSW AND PC ON STACK
      MOV (SP)+, @#ERRVEC+2 ;; SAVE THE PSW IN @#ERRVEC+2
      MOV #3776, R1 ;; SETUP ADDRESS
      TSTB (PC)+ ;; USE MEMORY MANAGEMENT?
$KT11: .WORD 200 ;; SET TO USE MEMORY MANAGEMENT
      BPL $SCORE ;; BR IF NO
      MOV # $KTNEX, @#ERRVEC ;; SET FOR TIMEOUT
      TST @#SRO ;; KT11 ARE YOU THERE?

```

```

11514 056264 052737 100000 056246 BIS #100000,$KT11 ;;YES--SET KT11 KEY
11515 056272 005046 CLR -(SP) ;;INITIALIZE FOR "PAR" LOADING
11516 056274 012702 172340 MOV #KIPAR0,R2 ;;ADDRESS OF FIRST "PAR"
11517 056300 012703 000010 MOV #108,R3 ;;LOAD EIGHT "PAR.'S" AND EIGHT "PDR.'S"
11518 056304 012762 077406 177740 1$: MOV #77406,-40(R2) ;;PDR = 4K, UP, READ/WRITE
11519 056312 011622 MOV (SP),(R2)+ ;;LOAD "PAR"
11520 056314 062716 000200 ADD #200,(SP) ;;UPDATE FOR NEXT "PAR"
11521 056320 077307 SOB R3,1$ ;;LOOP UNTIL ALL EIGHT ARE LOADED
11522 056322 012742 177600 MOV #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
11523 056326 005042 CLR -(R2) ;;SETUP KIPAR6 FOR TESTING
11524 056330 012737 056346 000004 MOV #2$,2$ERRVEC ;;CATCH TIMEOUT IF NO SR3
11525 056336 012737 000020 172516 MOV #20,2$SR3 ;;ENABLE 22 BIT MODE
11526 056344 000401 BR 3$ ;;THIS PDP-11 HAS A SR3 REGISTER
11527 056346 022626 2$: CMP (SP)+,(SP)+ ;;CLEAN OFF THE STACK--NO SR3
11528 056350 005237 177572 3$: INC 2$SR0 ;;TURN ON MEMORY MANAGEMENT
11529 056354 012737 056400 000004 MOV #SKTOUT,2$ERRVEC ;;SET FOR TIME OUT
11530 056362 005737 143776 4$: TST 2$143776 ;;TRAP ON NON-EX-MEM
11531 056366 062712 000040 ADD #40,(R2) ;;MAKE A 1K STEP
11532 056372 023712 172356 CMP 2$KIPAR7,(R2) ;;LAST ONE?
11533 056376 101371 BHI 4$ ;;NO--TRY IT
11534 056400 011202 SKTOUT: MOV (R2),R2 ;;GET LAST BANK+1
11535 056402 005037 177572 CLR 2$SR0 ;;TURN OFF MEMORY MANAGEMENT
11536 056406 000421 BR $$SIZEX
11537 056410 042737 100000 056246 SKTNEX: BIC #100000,$KT11 ;;KT11 NON-EXISTENT
11538 056416 012737 056446 000004 SCORE: MOV #SCROUT,2$ERRVEC ;;SET FOR TIMEOUT
11539 056424 005002 CLR R2 ;;SET UP BANK
11540 056426 062701 004000 1$: ADD #4000,R1 ;;INCREMENT BY 1K
11541 056432 062702 000040 ADD #40,R2 ;;1K STEP
11542 056436 005711 TST (R1) ;;TRAP ON TIME OUT
11543 056440 022701 177776 CMP #177776,R1 ;;LAST ONE
11544 056444 001370 BNE 1$ ;;NO--TRY AGAIN
11545 056446 162701 004000 SCROUT: SUB #4000,R1 ;;DROP BACK
11546 056452 162702 000040 $$SIZEX: SUB #40,R2 ;;RESTORE THE STACK
11547 056456 010006 MOV R0,SP ;;RESTORE ERROR VECTOR
11548 056460 012637 000006 MOV (SP)+,2$ERRVEC+2 ;;RESTORE ERROR VECTOR
11549 056464 012637 000004 MOV (SP)+,2$ERRVEC
11550 056470 010137 056512 MOV R1,$LSTAD ;;LAST ADDRESS
11551 056474 010237 056514 MOV R2,$LSTBK ;;LAST BANK
11552 056500 012603 MOV (SP)+,R3 ;;RESTORE R3
11553 056502 012602 MOV (SP)+,R2 ;;RESTORE R2
11554 056504 012601 MOV (SP)+,R1 ;;RESTORE R1
11555 056506 012600 MOV (SP)+,R0 ;;RESTORE R0
11556 056510 000207 RTS PC
11557 056512 000000 $LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
11558 056514 000000 $LSTBK: .WORD 0 ;;CONTAINS THE LAST BANK
11559
11560
11561
11562
11563
11564
11565
11566
11567
11568
11569

```

```

*****
*SAVE R0-R5
*CALL:
* SAVREG
*UPON RETURN FROM $$SAVREG THE STACK WILL LOOK LIKE:
*
*TOP---(+16)
* +2---(+18)
* +4---R5

```

```

11570
11571
11572
11573
11574
11575
11576 056516
11577 056516 010046
11578 056520 010146
11579 056522 010246
11580 056524 010346
11581 056526 010446
11582 056530 010546
11583 056532 016646 000022
11584 056536 016646 000022
11585 056542 016646 000022
11586 056546 016646 000022
11587 056552 000002
11588
11589
11590
11591
11592 056554
11593 056554 012666 000022
11594 056560 012666 000022
11595 056564 012666 000022
11596 056570 012666 000022
11597 056574 012605
11598 056576 012604
11599 056600 012603
11600 056602 012602
11601 056604 012601
11602 056606 012600
11603 056610 000002
11604
11605
11606
11607
11608
11609
11610
11611
11612 056612 010046
11613 056614 016600 000002
11614 056620 005740
11615 056622 111000
11616 056624 006300
11617 056626 016000 056646
11618 056632 000200
11619
11620
11621
11622
11623 056634 011646
11624 056636 016666 000004 000002
11625 056644 000002

```

```

;* +6---R4
;* +8---R3
;* +10---R2
;* +12---R1
;* +14---R0

```

SSAVREG:

```

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

```

;;RESTORE RO-R5

;;CALL:

;;RESREG

\$RESREG:

```

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

```

.SBTTL TRAP DECODER

```

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

```

\$TRAP:

```

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

```

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

\$TRAP2:

```

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

```


11626
11627
11628
11629
11630
11631
11632
11633
11634
11635
11636
11637
11638
11639
11640
11641
11642
11643
11644
11645
11646
11647
11648
11649
11650
11651
11652
11653
11654
11655
11656
11657
11658
11659
11660
11661
11662
11663
11664
11665
11666
11667
11668
11669
11670
11671
11672
11673
11674
11675
11676
11677
11678
11679
11680
11681

056646 056634
056650 053252
056652 054352
056654 054326
056656 054366
056660 054554

056662 055174
056664 056516
056666 056554
056670 055460

056672 020040 020040 042524
056700 052123 000040
056704 025052 020040 000
056711 125 044516 052502
056716 020123 040520 044522
056724 054524 042440 051122
056732 051117 000
056735 116 047117 042455
056742 044530 052123 047101
056750 020124 042515 047515
056756 054522 000
056761 116 047117 042455
056766 044530 052123 047101
056774 020124 051104 053111
057002 000105
057004 047125 052111 043040
057012 042511 042114 042440
057020 051122 051117 000
057025 123 041125 054523
057032 020123 044524 042515
057040 052517 000124
057044 020104 047524 041440
057052 050040 051101 052111
057060 020131 051105 047522
057066 000122
057070 051104 053111 020105
057076 042504 042524 052103
057104 042105 050040 051101
057112 052111 020131 051105
057120 047522 000122
057124 041501 046040 053517
057132 000
057133 123 042520 042105
057140 046040 051517 000123
057146 046111 042514 040507

.SBTTL TRAP TABLE

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

: ROUTINE

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

\$RDCHR ::CALL=RDCHR TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
\$SAVREG ::CALL=SAVREG TRAP+7(104407) SAVE RO-R5 ROUTINE
\$RESREG ::CALL=RESREG TRAP+10(104410) RESTORE RO-R5 ROUTINE
SCOPE1 ::CALL=SCOPE1 TRAP+11(104411) INTERNAL LOOP ON ERROR ROUTINE

TSTMSG: .ASCIZ / TEST /

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

EM2: .ASCIZ /NON-EXISTANT MEMORY/

EM3: .ASCIZ /NON-EXISTANT DRIVE/

EM4: .ASCIZ /UNIT FIELD ERROR/

EM5: .ASCIZ /SUBSYS TIMEOUT/

EM6: .ASCIZ /D TO C PARITY ERROR/

EM7: .ASCIZ /DRIVE DETECTED PARITY ERROR/

EM10: .ASCIZ /AC LOW/
EM11: .ASCIZ /SPEED LOSS/
EM12: .ASCIZ /ILLEGAL FUNCTION/

11682	057154	020114	052506	041516		
11683	057162	044524	047117	000		
11684	057167	120	047522	051107	EM13:	.ASCIZ /PROGRAMMING ERROR/
11685	057174	046501	044515	043516		
11686	057202	042440	051122	051117		
11687	057210	000				
11688	057211	116	047117	042455	EM14:	.ASCIZ /NON-EXISTANT FUNCTION/
11689	057216	044530	052123	047101		
11690	057224	020124	052506	041516		
11691	057232	044524	047117	000		
11692	057237	104	044522	042526	EM15:	.ASCIZ /DRIVE TYPE ERROR/
11693	057244	052040	050131	020105		
11694	057252	051105	047522	000122		
11695	057260	047506	046522	052101	EM16:	.ASCIZ /FORMAT ERROR/
11696	057266	042440	051122	051117		
11697	057274	000				
11698	057275	127	044522	042524	EM17:	.ASCIZ /WRITE LOCK ERROR/
11699	057302	046040	041517	020113		
11700	057310	051105	047522	000122		
11701	057316	051104	053111	020105	EM20:	.ASCIZ /DRIVE UNSAFE/
11702	057324	047125	040523	042506		
11703	057332	000				
11704	057333	123	042505	020113	EM21:	.ASCIZ /SEEK INCOMPLETE/
11705	057340	047111	047503	050115		
11706	057346	042514	042524	000		
11707	057353	103	046131	047111	EM22:	.ASCIZ /CYLINDER OVERFLOW/
11708	057360	042504	020122	053117		
11709	057366	051105	046106	053517		
11710	057374	000				
11711	057375	111	046114	043505	EM23:	.ASCIZ /ILLEGAL CYLINDER ADDRESS/
11712	057402	046101	041440	046131		
11713	057410	047111	042504	020122		
11714	057416	042101	051104	051505		
11715	057424	000123				
11716	057426	051104	053111	020105	EM24:	.ASCIZ /DRIVE OFF TRACK/
11717	057434	043117	020106	051124		
11718	057442	041501	000113			
11719	057446	051104	053111	020105	EM25:	.ASCIZ /DRIVE TIMING ERROR/
11720	057454	044524	044515	043516		
11721	057462	042440	051122	051117		
11722	057470	000				
11723	057471	104	052101	020101	EM26:	.ASCIZ /DATA LATE/
11724	057476	040514	042524	000		
11725	057503	103	047117	051124	EM27:	.ASCIZ /CONTROLLER TIMEOUT/
11726	057510	046117	042514	020122		
11727	057516	044524	042515	052517		
11728	057524	000124				
11729	057526	050117	051105	052101	EM30:	.ASCIZ /OPERATION INCOMPLETE/
11730	057534	047511	020116	047111		
11731	057542	047503	050115	042514		
11732	057550	042524	000			
11733	057553	110	040505	042504	EM31:	.ASCIZ /HEADER VRC ERROR/
11734	057560	020122	051126	020103		
11735	057566	051105	047522	000122		
11736	057574	040504	040524	041440	EM32:	.ASCIZ /DATA CHECK ERROR/
11737	057602	042510	045503	042440		

11738	057610	051122	051117	000		
11739	057615	127	044522	042524	EM33:	.ASCIZ /WRITE CHECK ERROR/
11740	057622	041440	042510	045503		
11741	057630	042440	051122	051117		
11742	057636	000				
11743	057637	104	052101	020101	EM34:	.ASCIZ /DATA MISCOMPARE/
11744	057644	044515	041523	046517		
11745	057652	040520	042522	000		
11746	057657	116	020117	051104	EM35:	.ASCIZ /NO DRIVE RESPONSE-UFE & NXD/
11747	057664	053111	020105	042522		
11748	057672	050123	047117	042523		
11749	057700	052455	042506	023040		
11750	057706	047040	042130	000		
11751	057713	104	044522	042526	EM36:	.ASCIZ /DRIVE ERROR WILL NOT CLEAR/
11752	057720	042440	051122	051117		
11753	057726	053440	046111	020114		
11754	057734	047516	020124	046103		
11755	057742	040505	000122			
11756	057746	051104	053111	020105	EM37:	.ASCIZ /DRIVE STATUS CHANGE WILL NOT CLEAR/
11757	057754	052123	052101	051525		
11758	057762	041440	040510	043516		
11759	057770	020105	044527	046114		
11760	057776	047040	052117	041440		
11761	060004	042514	051101	000		
11762	060011	101	052124	047047	EM40:	.ASCIZ /ATT'N BUT NO STATUS CHANGE OR FAULT/
11763	060016	041040	052125	047040		
11764	060024	020117	052123	052101		
11765	060032	051525	041440	040510		
11766	060040	043516	020105	051117		
11767	060046	043040	052501	052114		
11768	060054	000				
11769	060055	101	052124	047047	EM41:	.ASCIZ /ATT'N BUT DRIVE NOT AVAILABLE/
11770	060062	041040	052125	042040		
11771	060070	044522	042526	047040		
11772	060076	052117	040440	040526		
11773	060104	046111	041101	042514		
11774	060112	000				
11775	060113	101	052124	047047	EM42:	.ASCIZ /ATT'N WHEN NOT EXPECTED/
11776	060120	053440	042510	020116		
11777	060126	047516	020124	054105		
11778	060134	042520	052103	042105		
11779	060142	000				
11780	060143	105	051122	051117	EM43:	.ASCIZ /ERROR GATHERING DRIVE STATUS/
11781	060150	043440	052101	042510		
11782	060156	044522	043516	042040		
11783	060164	044522	042526	051440		
11784	060172	040524	052524	000123		
11785	060200	052515	052114	050111	EM52:	.ASCIZ /MULTIPLE DRIVE SELECT/
11786	060206	042514	042040	044522		
11787	060214	042526	051440	046105		
11788	060222	041505	000124			
11789	060226	042510	042101	051105	EM53:	.ASCIZ /HEADER COMPARE ERROR/
11790	060234	041440	046517	040520		
11791	060242	042522	042440	051122		
11792	060250	051117	000			
11793	060253	123	041125	054523	EM56:	.ASCIZ /SUBSYS TIMEOUT/

11794	060260	020123	044524	042515	
11795	060266	052517	000124		
11796	060272	051105	047522	020122	EM60: .ASCIZ /ERROR IN RECAL FOR RECOVERY/
11797	060300	047111	051040	041505	
11798	060306	046101	043040	051117	
11799	060314	051040	041505	053117	
11800	060322	051105	000131		
11801	060326	051120	043517	040522	EM61: .ASCIZ /PROGRAM ABORTING FATAL ERROR IN RETRY/
11802	060334	020115	041101	051117	
11803	060342	044524	043516	043040	
11804	060350	052101	046101	042440	
11805	060356	051122	051117	044440	
11806	060364	020116	042522	051124	
11807	060372	000131			
11808	060374	054503	044514	042116	EM62: .ASCIZ /CYLINDER MISCOMPARE/
11809	060402	051105	046440	051511	
11810	060410	047503	050115	051101	
11811	060416	000105			
11812	060420	046103	040505	020122	EM63: .ASCIZ /CLEAR CONTROLLER DID NOT CLEAR ERROR/
11813	060426	047503	052116	047522	
11814	060434	046114	051105	042040	
11815	060442	042111	047040	052117	
11816	060450	041440	042514	051101	
11817	060456	042440	051122	051117	
11818	060464	000			
11819	060465	116	020117	052101	EM64: .ASCIZ /NO ATT'N IN ATT'N SUMMARY REGISTER/
11820	060472	023524	020116	047111	
11821	060500	040440	052124	047047	
11822	060506	051440	046525	040515	
11823	060514	054522	051040	043505	
11824	060522	051511	042524	000122	
11825	060530	047125	047523	044514	EM65: .ASCIZ /UNSOLICITED ATTENTION/
11826	060536	044503	042524	020104	
11827	060544	052101	042524	052116	
11828	060552	047511	000116		
11829	060556	047125	054105	042520	EM66: .ASCIZ /UNEXPECTED DATA TYPE ERROR/
11830	060564	052103	042105	042040	
11831	060572	052101	020101	054524	
11832	060600	042520	042440	051122	
11833	060606	051117	000		
11834	060611	101	052124	047047	EM67: .ASCIZ /ATT'N DID NOT RESET WITH CLEAR/
11835	060616	042040	042111	047040	
11836	060624	052117	051040	051505	
11837	060632	052105	053440	052111	
11838	060640	020110	046103	040505	
11839	060646	000122			
11840	060650	052523	051502	051531	EM70: .ASCIZ /SUBSYS CLEAR DID NOT CLEAR DRIVE ATT'N/
11841	060656	041440	042514	051101	
11842	060664	042040	042111	047040	
11843	060672	052117	041440	042514	
11844	060700	051101	042040	044522	
11845	060706	042526	040440	052124	
11846	060714	047047	000		
11847	060717	104	052101	020101	EM71: .ASCIZ /DATA LATE WHEN UNLOADING HEADER/
11848	060724	040514	042524	053440	
11849	060732	042510	020116	047125	

11850	060740	047514	042101	047111		
11851	060746	020107	042510	042101		
11852	060754	051105	000			
11853	060757	103	047117	051124	EM72:	.ASCIZ /CONTROLLER ERROR WHEN DRIVER SERVICING/
11854	060764	046117	042514	020122		
11855	060772	051105	047522	020122		
11856	061000	044127	047105	042040		
11857	061006	044522	042526	020122		
11858	061014	042523	053122	041511		
11859	061022	047111	000107			
11860	061026	051104	053111	020105	EM73:	.ASCIZ /DRIVE DETECTED PARITY ERROR/
11861	061034	042504	042524	052103		
11862	061042	042105	050040	051101		
11863	061050	052111	020131	051105		
11864	061056	047522	000122			
11865	061062	047125	042504	044506	EM74:	.ASCIZ /UNDEFINED ERROR/
11866	061070	042516	020104	051105		
11867	061076	047522	000122			
11868	061102	040515	045522	047111	EM75:	.ASCIZ /MARKING THIS SECTOR BAD/
11869	061110	020107	044124	051511		
11870	061116	051440	041505	047524		
11871	061124	020122	040502	000104		
11872	061132	040502	020104	040504	EM76:	.ASCIZ /BAD DATA VERIF'N WITH READ. ECC OF LAST RETRY IS:/
11873	061140	040524	053040	051105		
11874	061146	043111	047047	053440		
11875	061154	052111	020110	042522		
11876	061162	042101	020056	041505		
11877	061170	020103	043117	046040		
11878	061176	051501	020124	042522		
11879	061204	051124	020131	051511		
11880	061212	000072				
11881	061214	042522	051124	020131	EM77:	.ASCIZ /RETRY SUCCESSFUL/
11882	061222	052523	041503	051505		
11883	061230	043123	046125	000		
11884	061235	122	052105	054522	EM100:	.ASCIZ /RETRY UNSUCCESSFUL/
11885	061242	052440	051516	041525		
11886	061250	042503	051523	052506		
11887	061256	000114				
11888	061260	040503	047116	052117	EM101:	.ASCIZ /CANNOT FIND A VALID HEADER IN TRACK JUST READ/
11889	061266	043040	047111	020104		
11890	061274	020101	040526	044514		
11891	061302	020104	042510	042101		
11892	061310	051105	044440	020116		
11893	061316	051124	041501	020113		
11894	061324	052512	052123	051040		
11895	061332	040505	000104			
11896	061336	040502	020104	042523	EM102:	.ASCIZ /BAD SECTOR ERROR ON SECTOR NOT LISTED BAD/
11897	061344	052103	051117	042440		
11898	061352	051122	051117	047440		
11899	061360	020116	042523	052103		
11900	061366	051117	047040	052117		
11901	061374	046040	051511	042524		
11902	061402	020104	040502	000104		
11903	061410	044524	042515	026504	EM103:	.ASCIZ /TIMED-OUT ON READ HDR/
11904	061416	052517	020124	047117		
11905	061424	051040	040505	020104		

11906	061432	042110	000122			
11907	061436	044524	042515	026504	EM104:	.ASCIZ /TIMED-OUT ON SEEK/
11908	061444	052517	020124	047117		
11909	061452	051440	042505	000113		
11910	061460	051104	053111	020105	EM105:	.ASCIZ /DRIVE SIEZED BY OTHER PORT/
11911	061466	044523	055105	042105		
11912	061474	041040	020131	052117		
11913	061502	042510	020122	047520		
11914	061510	052122	000			
11915	061513	104	052101	020101	EM106:	.ASCIZ /DATA MISCMPR WHILE BAI SET/
11916	061520	044515	041523	050115		
11917	061526	020122	044127	046111		
11918	061534	020105	040502	020111		
11919	061542	042523	000124			
11920	061546	047516	047040	046505	EM107:	.ASCIZ /NO NEM ERROR WHEN REF'ING LOC 760000/
11921	061554	042440	051122	051117		
11922	061562	053440	042510	020116		
11923	061570	042522	023506	047111		
11924	061576	020107	047514	020103		
11925	061604	033067	030060	030060		
11926	061612	000				
11927	061613	111	052116	050122	EM110:	.ASCIZ /INTRPT WHEN CNTRLR NOT RDY/
11928	061620	020124	044127	047105		
11929	061626	041440	052116	046122		
11930	061634	020122	047516	020124		
11931	061642	042122	000131			
11932	061646	047516	040440	052124	EM111:	.ASCIZ /NO ATT'N ON SEEK/
11933	061654	047047	047440	020116		
11934	061662	042523	045505	000		
11935	061667	104	044522	042526	EM112:	.ASCIZ /DRIVE'S CYLINDER INCORRECT/
11936	061674	051447	041440	046131		
11937	061702	047111	042504	020122		
11938	061710	047111	047503	051122		
11939	061716	041505	000124			
11940	061722	041101	051117	026524	EM113:	.ASCIZ /ABORT- CAN'T READ BSF/
11941	061730	041440	047101	052047		
11942	061736	051040	040505	020104		
11943	061744	051502	000106			
11944	061750	052113	030461	043040	EM114:	.ASCIZ /KT11 FAILURE/
11945	061756	044501	052514	042522		
11946	061764	000				
11947	061765	115	046505	050040	EM115:	.ASCIZ /MEM PARITY ERR/
11948	061772	051101	052111	020131		
11949	062000	051105	000122			
11950	062004	052503	051122	047105	DH100:	.ASCIZ /CURRENT COMMAND :/
11951	062012	020124	047503	046515		
11952	062020	047101	020104	000072		
11953	062026	051120	053105	047511	DH105:	.ASCIZ /PREVIOUS COMMAND :/
11954	062034	051525	041440	046517		
11955	062042	040515	042116	035040		
11956	062050	000				
11957	062051	122	033113	030461	DH200:	.ASCIZ /RK611 REGISTERS :/
11958	062056	051040	043505	051511		
11959	062064	042524	051522	035040		
11960	062072	000				
11961	062073	122	046505	044501	DH500:	.ASCIZ /REMAINING REGISTERS NOT VALID/

12018	062560	020062	020040	020040	
12019	062566	030101	020063	020040	
12020	062574	020040	030102	000063	
12021	062602	042510	042101	051105	DH601: .ASCIZ /HEADER SHOULD BE: /
12022	062610	051440	047510	046125	
12023	062616	020104	042502	000072	
12024	062624	042510	042101	051105	DH602: .ASCIZ /HEADER OF SECTOR 0 THIS CYLINDER IS: /
12025	062632	047440	020106	042523	
12026	062640	052103	051117	030040	
12027	062646	052040	044510	020123	
12028	062654	054503	044514	042116	
12029	062662	051105	044440	035123	
12030	062670	000			
12031	062671	120	041501	020113	DH604: .ASCIZ /PACK ADDRESS OF ERROR(S) : /
12032	062676	042101	051104	051505	
12033	062704	020123	043117	042440	
12034	062712	051122	051117	05150	
12035	062720	020051	000072		
12036	062724	054503	047114	051104	DH6041: .ASCIZ /CYLNDR TRACK SECTOR/
12037	062732	020040	051124	041501	
12038	062740	020113	020040	042523	
12039	062746	052103	051117	000	
12040	062753	103	046131	042116	DH6042: .ASCIZ /CYLNDR TRACK/
12041	062760	020122	052040	040522	
12042	062766	045503	000		
12043	062771	105	041503	042040	DH605: .ASCIZ /ECC DATA IS: /
12044	062776	052101	020101	051511	
12045	063004	000072			
12046	063006	047520	020123	020040	DH6051: .ASCIZ /POS PAT CORRECTABLE? /
12047	063014	020040	040520	020124	
12048	063022	020040	020040	047503	
12049	063030	051122	041505	040524	
12050	063036	046102	037505	000	
12051	063043	127	051117	030504	DH606: .ASCIZ /WORD1 WORD2 WORD3/
12052	063050	020040	053440	051117	
12053	063056	031104	020040	053440	
12054	063064	051117	031504	000	
12055	063071	102	042101	044040	DH607: .ASCIZ /BAD HEADER IS : /
12056	063076	040505	042504	020122	
12057	063104	051511	035040	000	
12058	063111	127	020104	020043	DH701: .ASCIZ /WD # GOOD BAD HI PHY LO PHY VRT AD KIPAR6/
12059	063116	020040	043440	047517	
12060	063124	020104	020040	041040	
12061	063132	042101	020040	020040	
12062	063140	044040	020111	044120	
12063	063146	020131	046040	020117	
12064	063154	044120	020131	053040	
12065	063162	052122	040440	020104	
12066	063170	045440	050111	051101	
12067	063176	000066			
12068	063200	047507	042117	020040	DH702: .ASCIZ /GOOD BAD/
12069	063206	020040	040502	000104	
12070	063214	040506	046111	047111	DH703: .ASCIZ /FAILING DATA WORD : /
12071	063222	020107	040504	040524	
12072	063230	053440	051117	020104	
12073	063236	000072			

12074	063240	051123	004460	051123	DH704:	.ASCIZ	/SR0	SR1	SR2	SR3/
12075	063246	004461	051123	004462						
12076	063254	051123	000063							
12077	063260	052113	030461	051040	DH705:	.ASCIZ	/KT11 REGS :/			
12078	063266	043505	020123	000072						
12079	063274	042515	020115	042522	DH706:	.ASCIZ	/MEM REGS :/			
12080	063302	051507	035040	000						
12081	063307	120	004503	047514	DH707:	.ASCIZ	/PC	LOERAD	HIERAD	MEMSYS/
12082	063314	051105	042101	044011						
12083	063322	042511	040522	004504						
12084	063330	042515	051515	051531						
12085	063336	000								
12086	063337	120	004503	051503	DH710:	.ASCIZ	/PC	CSR AD	CSR/	
12087	063344	020122	042101	041411						
12088	063352	051123	000							
12089	063355	103	046131	047111	DH711:	.ASCIZ	/CYLINDERS :/			
12090	063362	042504	051522	035040						
12091	063370	000								
12092	063371	122	042113	020123	DH204:	.ASCIZ	/RKDS	RKER	RKMR2	RKMR3/
12093	063376	020040	051040	042513						
12094	063404	020122	020040	051040						
12095	063412	046513	031122	020040						
12096	063420	051040	046513	031522						
12097	063426	000								
12098	063427	105	051122	051117	DH502:	.ASCIZ	/ERROR TRYING TO GET FAILURE INFO/			
12099	063434	052040	054522	047111						
12100	063442	020107	047524	043440						
12101	063450	052105	043040	044501						
12102	063456	052514	042522	044440						
12103	063464	043116	000117							
12104	063470	042523	047503	042116	DH503:	.ASCIZ	/SECOND TIMEOUT OCCURRED GETTING DRIVE STATUS/			
12105	063476	052040	046511	047505						
12106	063504	052125	047440	041503						
12107	063512	051125	042522	020104						
12108	063520	042507	052124	047111						
12109	063526	020107	051104	053111						
12110	063534	020105	052123	052101						
12111	063542	051525	000							
12112	063545	116	046525	042502	DH800:	.ASCIZ	/NUMBER OF RETRIES:/			
12113	063552	020122	043117	051040						
12114	063560	052105	044522	051505						
12115	063566	000072								
12116										
12117	063570	001116	005500	005476	DT100:	.EVEN	\$ERRPC,DRIVE,ERRCOM,\$REG0,\$REG1,\$REG2,\$REG3			
12118	063576	001162	001164	001166						
12119	063604	001170								
12120	063606	001256	001260		DT102:	.WORD	\$REG36,\$REG37			
12121	063612	001174	001176	001200	DT201:	.WORD	\$REG5,\$REG6,\$REG7,\$REG10,\$REG11,\$REG12,\$REG13			
12122	063620	001202	001204	001206						
12123	063626	001210								
12124	063630	001212	001214		DT202:	.WORD	\$REG14,\$REG15			
12125	063634	001216	001220	001222	DT203:	.WORD	\$REG16,\$REG17,\$REG20,\$REG21,\$REG22,\$REG23,\$REG24,\$REG25			
12126	063642	001224	001226	001230						
12127	063650	001232	001234							
12128	063654	001174	001176	001200	DT601:	.WORD	\$REG5,\$REG6,\$REG7			
12129	063662	001202	001204	001206	DT602:	.WORD	\$REG10,\$REG11,\$REG12,\$REG13,\$REG14,\$REG15,\$REG16			

12130	063670	001210	001212	001214					
12131	063676	001216							
12132	063700	005254	005252		DT103:	.WORD	PRMPHO,PRMPLG		
12133									
12134	063704	000006			DF01:	.WORD	6		;NUMBER OF HEADER LINES
12135	063706	000				.BYTE	0		;NUMBER OF COL FOR FIRST HDR
12136	063707	000				.BYTE	0		;ALL CHARACTERS OCTAL
12137	063710	062210				.WORD	DH101		;SECOND HDR LINE
12138	063712	007	000			.BYTE	7,0		;NUM OF COL-ALL OCTAL
12139	063714	062277				.WORD	DH102		
12140	063716	002	000			.BYTE	2,0		
12141	063720	062051				.WORD	DH200		
12142	063722	000	000			.BYTE	0,0		
12143	063724	062402				.WORD	DH201		
12144	063726	007	000			.BYTE	7,0		
12145	063730	062073				.WORD	DH500		
12146	063732	000	000			.BYTE	0,0		
12147									
12148	063734	000007			DF02:	.WORD	7		
12149	063736	000	000			.BYTE	0,0		
12150	063740	062210				.WORD	DH101		
12151	063742	007	000			.BYTE	7,0		
12152	063744	062277				.WORD	DH102		
12153	063746	002	000			.BYTE	2,0		
12154	063750	062051				.WORD	DH200		
12155	063752	000	000			.BYTE	0,0		
12156	063754	062402				.WORD	DH201		
12157	063756	007	000			.BYTE	7,0		
12158	063760	062471				.WORD	DH202		
12159	063762	002	000			.BYTE	2,0		
12160	063764	062506				.WORD	DH203		
12161	063766	010	000			.BYTE	10,0		
12162									
12163	063770	000010			DF03:	.WORD	10		
12164	063772	000	000			.BYTE	0,0		
12165	063774	062210				.WORD	DH101		
12166	063776	007	000			.BYTE	7,0		
12167	064000	062277				.WORD	DH102		
12168	064002	002	000			.BYTE	2,0		
12169	064004	062051				.WORD	DH200		
12170	064006	000	000			.BYTE	0,0		
12171	064010	062402				.WORD	DH201		
12172	064012	007	000			.BYTE	7,0		
12173	064014	062131				.WORD	DH501		; "THE FOLLOWING REG DATA MB INCORRECT"
12174	064016	000	000			.BYTE	0,0		
12175	064020	062471				.WORD	DH202		
12176	064022	002	000			.BYTE	2,0		
12177	064024	062506				.WORD	DH203		
12178	064026	010	000			.BYTE	10,0		
12179									
12180	064030	000003			DF04:	.WORD	3		
12181	064032	000	000			.BYTE	0,0		
12182	064034	062210				.WORD	DH101		
12183	064036	007	000			.BYTE	7,0		
12184	064040	062277				.WORD	DH102		
12185	064042	002	000			.BYTE	2,0		

12186					
12187	064044	000007		DF05:	.WORD 7 ;OPI, HVRC
12189	064046	000	000		.BYTE 0,0
12189	064050	062210			.WORD DH101
12190	064052	007	000		.BYTE 7,0
12191	064054	062277			.WORD DH102
12192	064056	002	000		.BYTE 2,0
12193	064060	062602			.WORD DH601
12194	064062	000	000		.BYTE 0,0
12195	064064	063043			.WORD DH606
12196	064066	003	000		.BYTE 3,0
12197	064070	062624			.WORD DH602
12198	064072	000	000		.BYTE 0,0
12199	064074	063043			.WORD DH606
12200	064076	003	000		.BYTE 3,0
12201					
12202	064100	000007		DF07:	.WORD 7
12203	064102	000	000		.BYTE 0,0
12204	064104	062210			.WORD DH101
12205	064106	007	000		.BYTE 7,0
12206	064110	062277			.WORD DH102
12207	064112	002	000		.BYTE 2,0
12208	064114	062671			.WORD DH604
12209	064116	000	000		.BYTE 0,0
12210	064120	062724			.WORD DH6041
12211	064122	003	000		.BYTE 3,0
12212	064124	062771			.WORD DH605
12213	064126	000	000		.BYTE 0,0
12214	064130	063006			.WORD DH6051
12215	064132	003	000		.BYTE 3,0
12216					
12217	064134	000005		DF10:	.WORD 5
12218	064136	000	000		.BYTE 0,0
12219	064140	062210			.WORD DH101
12220	064142	007	000		.BYTE 7,0
12221	064144	062277			.WORD DH102
12222	064146	002	000		.BYTE 2,0
12223	064150	062671			.WORD DH604
12224	064152	000	000		.BYTE 0,0
12225	064154	062724			.WORD DH6041
12226	064156	003	000		.BYTE 3,0
12227					
12228	064160	000004		DF11:	.WORD 4
12229	064162	000	000		.BYTE 0,0
12230	064164	062671			.WORD DH604
12231	064166	000	000		.BYTE 0,0
12232	064170	062724			.WORD DH6041
12233	064172	003	000		.BYTE 3,0
12234	064174	063111			.WORD DH701
12235	064176	000	000		.BYTE 0,0
12236					
12237	064200	000006		DF12:	.WORD 6
12238	064202	000	000		.BYTE 0,0
12239	064204	062210			.WORD DH101
12240	064206	007	000		.BYTE 7,0
12241	064210	062277			.WORD DH102

12242	064212	002	000	.BYTE	2,0	
12243	064214	062051		.WORD	DH200	
12244	064216	000	000	.BYTE	0,0	
12245	064220	062402		.WORD	DH201	
12246	064222	007	000	.BYTE	7,0	
12247	064224	063371		.WORD	DH204	
12248	064226	004	000	.BYTE	4,0	
12249						
12250	064230	000006		DF13: .WORD	6	:FORMAT FOR 2ND LEVEL ERROR
12251	064232	000	000	.BYTE	0,0	:IN HEADER COMPARE ERROR
12252	064234	062210		.WORD	DH101	:AND 2ND LEVEL HEADER
12253	064236	007	000	.BYTE	7,0	:VRC ERROR
12254	064240	062277		.WORD	DH102	
12255	064242	002	000	.BYTE	2,0	
12256	064244	062602		.WORD	DH601	
12257	064246	000	000	.BYTE	0,0	
12258	064250	063043		.WORD	DH606	
12259	064252	003	000	.BYTE	3,0	
12260	064254	063427		.WORD	DH502	
12261	064256	000	000	.BYTE	0,0	
12262						
12263	064260	000006		DF14: .WORD	6	:FORMAT FOR 2ND LEVEL ERROR
12264	064262	000	000	.BYTE	0,0	:IN OPERATION INCOMPLETE ERROR
12265	064264	062210		.WORD	DH101	
12266	064266	007	000	.BYTE	7,0	
12267	064270	062277		.WORD	DH102	
12268	064272	002	000	.BYTE	2,0	
12269	064274	062602		.WORD	DH601	
12270	064276	000	000	.BYTE	0,0	
12271	064300	063043		.WORD	DH606	
12272	064302	003	000	.BYTE	3,0	
12273	064304	063427		.WORD	DH502	
12274	064306	000	000	.BYTE	0,0	
12275						
12276	064310	000011		DF15: .WORD	11	
12277	064312	000	000	.BYTE	0,0	
12278	064314	062210		.WORD	DH101	
12279	064316	007	000	.BYTE	7,0	
12280	064320	062277		.WORD	DH102	
12281	064322	002	000	.BYTE	2,0	
12282	064324	062051		.WORD	DH200	
12283	064326	000	000	.BYTE	0,0	
12284	064330	062402		.WORD	DH201	
12285	064332	007	000	.BYTE	7,0	
12286	064334	062471		.WORD	DH202	
12287	064336	002	000	.BYTE	2,0	
12288	064340	063470		.WORD	DH503	
12289	064342	000	000	.BYTE	0,0	
12290	064344	062131		.WORD	DH501	
12291	064346	000	000	.BYTE	0,0	
12292	064350	062506		.WORD	DH203	
12293	064352	010	000	.BYTE	10,0	
12294						
12295	064354	000011		DF16: .WORD	11	
12296	064356	000	000	.BYTE	0,0	
12297	064360	062210		.WORD	DH101	

12298	064362	007	000	.BYTE	7,0
12299	064364	062277		.WORD	DH102
12300	064366	002	000	.BYTE	2,0
12301	064370	062051		.WORD	DH200
12302	064372	000	000	.BYTE	0,0
12303	064374	062402		.WORD	DH201
12304	064376	007	000	.BYTE	7,0
12305	064400	062471		.WORD	DH202
12306	064402	002	000	.BYTE	2,0
12307	064404	063427		.WORD	DH502
12308	064406	000	000	.BYTE	0,0
12309	064410	062131		.WORD	DH501
12310	064412	000	000	.BYTE	0,0
12311	064414	062506		.WORD	DH203
12312	064416	010	000	.BYTE	10,0
12313					
12314	064420	000005		DF17: .WORD	5
12315	064422	000	000	.BYTE	0,0
12316	064424	062210		.WORD	DH101
12317	064426	007	000	.BYTE	7,0
12318	064430	062277		.WORD	DH102
12319	064432	002	000	.BYTE	2,0
12320	064434	063355		.WORD	DH711
12321	064436	000	000	.BYTE	0,0
12322	064440	063200		.WORD	DH702
12323	064442	002	000	.BYTE	2,0
12324					
12325	064444	000002		DF20: .WORD	2
12326	064446	000	000	.BYTE	0,0
12327	064450	062340		.WORD	DH104
12328	064452	002	000	.BYTE	2,0
12329					
12330	064454	000002		DF21: .WORD	2
12331	064456	000	000	.BYTE	0,0
12332	064460	063006		.WORD	DH6051
12333	064462	003	000	.BYTE	3,0
12334					
12335	064464	000006		DF22: .WORD	6
12336	064466	000	000	.BYTE	0,0
12337	064470	062004		.WORD	DH100
12338	064472	000	000	.BYTE	0,0
12339	064474	062210		.WORD	DH101
12340	064476	007	000	.BYTE	7,0
12341	064500	062277		.WORD	DH102
12342	064502	002	000	.BYTE	2,0
12343	064504	062360		.WORD	DH106
12344	064506	000	000	.BYTE	0,0
12345	064510	062340		.WORD	DH104
12346	064512	002	000	.BYTE	2,0
12347					
12348	064514	000002		DF23: .WORD	2
12349	064516	000	000	.BYTE	0,0
12350	064520	063545		.WORD	DH800
12351	064522	001	000	.BYTE	1,0
12352					
12353	064524	000001		DF24: .WORD	1

12354	064526	002	000		.BYTE	2,0
12355						
12356	064530	000000		DF25:	.WORD	0
12357	064532	007	000		.BYTE	7,0
12358						
12359	064534	000002		DF26:	.WORD	2
12360	064536	002	000		.BYTE	2,0
12361	064540	062506			.WORD	DH203
12362	064542	010	000		.BYTE	10,0
12363						
12364	064544	000005		DF27:	.WORD	5
12365	064546	000	000		.BYTE	0,0
12366	064550	062210			.WORD	DH101
12367	064552	007	000		.BYTE	7,0
12368	064554	062277			.WORD	DH102
12369	064556	002	000		.BYTE	2,0
12370	064560	063214			.WORD	DH703
12371	064562	000	000		.BYTE	0,0
12372	064564	063200			.WORD	DH702
12373	064566	002	000		.BYTE	2,0
12374						
12375	064570	000005		DF30:	.WORD	5
12376	064572	000	000		.BYTE	0,0
12377	064574	062210			.WORD	DH101
12378	064576	007	000		.BYTE	7,0
12379	064600	062277			.WORD	DH102
12380	064602	002	000		.BYTE	2,0
12381	064604	063260			.WORD	DH705
12382	064606	000	000		.BYTE	0,0
12383	064610	063240			.WORD	DH704
12384	064612	004	000		.BYTE	4,0
12385						
12386	064614	000005		DF31:	.WORD	5
12387	064616	000	000		.BYTE	0,0
12388	064620	062210			.WORD	DH101
12389	064622	007	000		.BYTE	7,0
12390	064624	062277			.WORD	DH102
12391	064626	002	000		.BYTE	2,0
12392	064630	063274			.WORD	DH706
12393	064632	000	000		.BYTE	0,0
12394	064634	063337			.WORD	DH710
12395	064636	003	000		.BYTE	3,0
12396						
12397	064640			RWBUF:		;READ/WRITE DATA BUF <AT LEAST 512(DEC) WORDS>
12398						
12399	064640	005015	020040	020040	NOTMSG: .ASCII	<15><12>/ *** NOTE ***/<15><12><12>
12400	064646	020040	020040	020040		
12401	064654	025052	020052	047516		
12402	064662	042524	025040	025052		
12403	064670	005015	012			
12404	064673	101	046114	042040	.ASCII	/ALL DRIVES TO BE TESTED MUST HAVE :/<15><12><12>
12405	064700	044522	042526	020123		
12406	064706	047524	041040	020105		
12407	064714	042524	052123	042105		
12408	064722	046440	051525	020124		
12409	064730	040510	042526	035040		

12410	064736	005015	012		
12411	064741	061	020056	020101	.ASCII /1. A 20 OR 22 SECTOR FORMATTED CARTRIDGE/<15><12>
12412	064746	030062	047440	020122	
12413	064754	031062	051440	041505	
12414	064762	047524	020122	047506	
12415	064770	046522	052101	042524	
12416	064776	020104	040503	052122	
12417	065004	044522	043504	006505	
12418	065012	012			
12419	065013	062	020056	042510	.ASCII /2. HEADS MANUALLY LOADED/<15><12>
12420	065020	042101	020123	040515	
12421	065026	052516	046101	054514	
12422	065034	046040	040517	042504	
12423	065042	006504	012		
12424	065045	063	020056	042504	.ASCII /3. DESIRED PORT SELECTED/<15><12>
12425	065052	044523	042522	020104	
12426	065060	047520	052122	051440	
12427	065066	046105	041505	042524	
12428	065074	006504	012		
12429	065077	064	020056	051127	.ASCII /4. WRITE LOCK DISABLED/<15><12>
12430	065104	052111	020105	047514	
12431	065112	045503	042040	051511	
12432	065120	041101	042514	006504	
12433	065126	012			
12434	065127	065	020056	051104	.ASCII /5. DRIVE READY LIGHT ON/<15><12><12>
12435	065134	053111	020105	042522	
12436	065142	042101	020131	044514	
12437	065150	044107	020124	047117	
12438	065156	005015	012		
12439	065161	104	044522	042526	.ASCII /DRIVES NOT TO BE TESTED MUST HAVE BOTH/<15><12>
12440	065166	020123	047516	020124	
12441	065174	047524	041040	020105	
12442	065202	042524	052123	042105	
12443	065210	046440	051525	020124	
12444	065216	040510	042526	041040	
12445	065224	052117	006510	012	
12446	065231	120	051117	051524	.ASCIZ /PORTS DE-SELECTED./<15><12><12>
12447	065236	042040	026505	042523	
12448	065244	042514	052103	042105	
12449	065252	006456	005012	000	
12450					
12451	065257	015	044412	051516	HLPFIL: .ASCII <15><12>/INSTRUCTIONS FOR USING RK06 HEAD ALIGNMENT AID :/<15><12>
12452	065264	051124	041525	044524	
12453	065272	047117	020123	047506	
12454	065300	020122	051525	047111	
12455	065306	020107	045522	033060	
12456	065314	044040	040505	020104	
12457	065322	046101	043511	046516	
12458	065330	047105	020124	044501	
12459	065336	020104	006472	012	
12460	065343	052	025052	025052	.ASCII /*****/<15><12>
12461	065350	025052	025052	025052	
12462	065356	025052	025052	025052	
12463	065364	025052	025052	025052	
12464	065372	025052	025052	025052	
12465	065400	025052	025052	025052	

F03

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
 DZR6NC.F11 05-OCT-76 10:07 TRAP TABLE

MACY11 27(1006) 05-OCT-76 10:59 PAGE 239

SEQ 0238

12466	065406	025052	025052	025052	
12467	065414	025052	025052	006452	
12468	065422	012			
12469	065423	115	052517	052116	.ASCII /MOUNT AN RK06 ALIGNMENT CARTRIDGE ON THE DESIRED /<15><12>
12470	065430	040440	020116	045522	
12471	065436	033060	040440	044514	
12472	065444	047107	042515	052116	
12473	065452	041440	051101	051124	
12474	065460	042111	042507	047440	
12475	065466	020116	044124	020105	
12476	065474	042504	044523	042522	
12477	065502	020104	005015		
12478	065506	051104	053111	026105	.ASCII /DRIVE, AND INSURE THAT THE DRIVE IS WRITE-LOCKED. /<15><12>
12479	065514	040440	042116	044440	
12480	065522	051516	051125	020105	
12481	065530	044124	052101	052040	
12482	065536	042510	042040	044522	
12483	065544	042526	044440	020123	
12484	065552	051127	052111	026505	
12485	065560	047514	045503	042105	
12486	065566	020056	005015		
12487	065572	047503	047116	041505	.ASCII /CONNECT THE ALIGNMENT INDICATOR TO THE DESIRED DRIVE, /<15><12>
12488	065600	020124	044124	020105	
12489	065606	046101	043511	046516	
12490	065614	047105	020124	047111	
12491	065622	044504	040503	047524	
12492	065630	020122	047524	052040	
12493	065636	042510	042040	051505	
12494	065644	051111	042105	042040	
12495	065652	044522	042526	006454	
12496	065660	012			
12497	065661	126	040511	052040	.ASCII /VIA THE HEAD ALIGNMENT CABLE ONLY, AND CYCLE UP THE /<15><12>
12498	065666	042510	044040	040505	
12499	065674	020104	046101	043511	
12500	065702	046516	047105	020124	
12501	065710	040503	046102	020105	
12502	065716	047117	054514	020054	
12503	065724	047101	020104	054503	
12504	065732	046103	020105	050125	
12505	065740	052040	042510	005015	
12506	065746	051104	053111	027105	.ASCII /DRIVE. /<15><12>
12507	065754	005015			
12508	065756	042522	050123	047117	.ASCII /RESPOND TO ALL REQUESTS FOR PARAMETERS, BY ENTERING /<15><12>
12509	065764	020104	047524	040440	
12510	065772	046114	051040	050505	
12511	066000	042525	052123	020123	
12512	066006	047506	020122	040520	
12513	066014	040522	042515	042524	
12514	066022	051522	020054	054502	
12515	066030	042440	052116	051105	
12516	066036	047111	020107	005015	
12517	066044	044124	020105	042504	.ASCII /THE DESIRED PARAMETER VALUE (NO <CR> NEEDED). /<15><12>
12518	066052	044523	042522	020104	
12519	066060	040520	040522	042515	
12520	066066	042524	020122	040526	
12521	066074	052514	020105	047050	

12522	066102	020117	041474	037122	
12523	066110	047040	042505	042504	
12524	066116	024504	006456	012	
12525	066123	124	042510	042522	.ASCII /THERE ARE TWO MODES OF OPERATION : MANUAL MODE /<15><12>
12526	066130	040440	042522	052040	
12527	066136	047527	046440	042117	
12528	066144	051505	047440	020106	
12529	066152	050117	051105	052101	
12530	066160	047511	020116	020072	
12531	066166	046440	047101	040525	
12532	066174	020114	047515	042504	
12533	066202	006440	012		
12534	066205	101	046114	053517	.ASCII /ALLOWS SELECTION OF DRIVES AND HEADS BY TTY INPUT,/<15><12>
12535	066212	020123	042523	042514	
12536	066220	052103	047511	020116	
12537	066226	043117	042040	044522	
12538	066234	042526	020123	047101	
12539	066242	020104	042510	042101	
12540	066250	020123	054502	052040	
12541	066256	054524	044440	050116	
12542	066264	052125	006454	012	
12543	066271	101	042116	040440	.ASCII /AND AUTO MODE ALLOWS DRIVES AND HEADS TO BE SELECTED/<15><12>
12544	066276	052125	020117	047515	
12545	066304	042504	040440	046114	
12546	066312	053517	020123	051104	
12547	066320	053111	051505	040440	
12548	066326	042116	044040	040505	
12549	066334	051504	052040	020117	
12550	066342	042502	051440	046105	
12551	066350	041505	042524	006504	
12552	066356	012			
12553	066357	102	020131	043117	.ASCII /BY OFF-ON OPERATION OF DRIVE PORT SELECT SWITCHES./<15><12>
12554	066364	026506	047117	047440	
12555	066372	042520	040522	044524	
12556	066400	047117	047440	020106	
12557	066406	051104	053111	020105	
12558	066414	047520	052122	051440	
12559	066422	046105	041505	020124	
12560	066430	053523	052111	044103	
12561	066436	051505	006456	012	
12562	066443	111	020116	044505	.ASCII /IN EITHER MODE, UP TO 5 MINUTES OF SEEK EXERCISES/<15><12>
12563	066450	044124	051105	046440	
12564	066456	042117	026105	052440	
12565	066464	020120	047524	032440	
12566	066472	046440	047111	052125	
12567	066500	051505	047440	020106	
12568	066506	042523	045505	042440	
12569	066514	042530	041522	051511	
12570	066522	051505	005015		
12571	066526	040515	020131	042502	.ASCII /MAY BE REQUESTED FOR EACH DRIVE. /<15><12>
12572	066534	051040	050505	042525	
12573	066542	052123	042105	043040	
12574	066550	051117	042440	041501	
12575	066556	020110	051104	053111	
12576	066564	027105	006440	012	
12577	066571	101	051514	020117	.ASCII /ALSO IN EITHER MODE, A VERIFY OPTION ALLOWS HEAD/<15><12>

12578	066576	047111	042440	052111	
12579	066604	042510	020122	047515	
12580	066612	042504	020054	020101	
12581	066620	042526	044522	054506	
12582	066626	047440	052120	047511	
12583	066634	020116	046101	047514	
12584	066642	051527	044040	040505	
12585	066650	006504	012		
12586	066653	123	046105	041505	.ASCII /SELECTION WITHOUT THE UNLOADING AND LOADING OF /<<15><12>
12587	066660	044524	047117	053440	
12588	066666	052111	047510	052125	
12589	066674	052040	042510	052440	
12590	066702	046116	040517	044504	
12591	066710	043516	040440	042116	
12592	066716	046040	040517	044504	
12593	066724	043516	047440	020106	
12594	066732	005015			
12595	066734	044124	020105	051104	.ASCII /THE DRIVE BY THE PROGRAM, WHICH OTHERWISE OCCURS/<<15><12>
12596	066742	053111	020105	054502	
12597	066750	052040	042510	050040	
12598	066756	047522	051107	046501	
12599	066764	020054	044127	041511	
12600	066772	020110	052117	042510	
12601	067000	053522	051511	020105	
12602	067006	041517	052503	051522	
12603	067014	005015			
12604	067016	047524	040440	046114	.ASCII /TO ALLOW MANIPULATION OF THE ALIGNMENT TOOL./<<15><12>
12605	067024	053517	046440	047101	
12606	067032	050111	046125	052101	
12607	067040	047511	020116	043117	
12608	067046	052040	042510	040440	
12609	067054	044514	047107	042515	
12610	067062	052116	052040	047517	
12611	067070	027114	005015		
12612	067074	047524	051040	051505	.ASCII /TO RESTART EITHER MODE, TYPE <↑Z>./<<15><12>
12613	067102	040524	052122	042440	
12614	067110	052111	042510	020122	
12615	067116	047515	042504	020054	
12616	067124	054524	042520	036040	
12617	067132	055136	027076	005015	
12618	067140	047524	051040	051505	.ASCII /TO RESTART ALIGNMENT AID, TYPE <↑R>./<<15><12>
12619	067146	040524	052122	040440	
12620	067154	044514	047107	042515	
12621	067162	052116	040440	042111	
12622	067170	020054	054524	042520	
12623	067176	036040	051136	027076	
12624	067204	005015			
12625	067206	047524	051440	046105	.ASCII /TO SELECT NEW DRIVES IN MANUAL MODE, TYPE <↑C>./<<15><12><12>
12626	067214	041505	020124	042516	
12627	067222	020127	051104	053111	
12628	067230	051505	044440	020116	
12629	067236	040515	052516	046101	
12630	067244	046440	042117	026105	
12631	067252	052040	050131	020105	
12632	067260	057074	037103	006456	
12633	067266	005012			

12634	067270	047506	020122	042510
12635	067276	042101	040440	044514
12636	067304	047107	042515	052116
12637	067312	050040	047522	042503
12638	067320	052504	042522	020054
12639	067326	042522	042506	020122
12640	067334	047524	043040	042511
12641	067342	042114	006440	012
12642	067347	124	051505	020124
12643	067354	047502	020130	051050
12644	067362	030113	052066	026101
12645	067370	051040	030113	052066
12646	067376	024502	047440	042520
12647	067404	040522	047524	023522
12648	067412	020123	040515	052516
12649	067420	046101	006456	000012
12650				
12651				
12652				
12653	000001			

.ASCII /FOR HEAD ALIGNMENT PROCEDURE, REFER TO FIELD /<15><12>

.ASCIZ /TEST BOX (RK06TA, RK06TB) OPERATOR'S MANUAL./<15><12>

.END

APTSIZ=	000200	4754	6259	11478#																	
APTSPO=	000100	10816	11439	11480#																	
ASKDRV	024410	6324#	6339	6343	6358	6390	6460														
ASKHED	024630	6376#	6404	6416	6423																
ASKMDE	016156	5155#	5161	5162	5167	5182	5187	5192	5217	5231	5242	5251	5267	5286							
		5295	5321	5344	5416	7696															
ASKMOD	024322	6293	6298	6303#	6317																
ASKTMD	015412	5037#	5040	5041	5046	5070	5078	5084	5100	5111	5120	5145									
ASKTST	015406	4951	5036#																		
ASTART	023634	2220	6219#																		
ASWREG=	000000	2349	2362																		
AS2SP2	056704	8967	11649#																		
AESTN=	000000	2349	2353																		
AUNIT =	000000	2349	2356																		
AUSWR =	000000	2349	2363																		
AUTAL	025446	6486	6493	6500#	6581	6602															
AUTALN	013060	4641#	6505																		
AUTEX	026442	6489	6641#	6662	6675																
AUTEXR	013176	4656#	6644																		
AUTO	025332	6315	6473#	6521	6652	6698	6710														
AUTODR	013310	4669#	6679#																		
AUTOEX	013267	4666#	6680																		
AUTVRF	013116	4646#	6503																		
AVECT1=	120210	2256#	2349	2388	4806																
AVECT2=	000000	2349	2389																		
A.ABNL	003036	3579#	5584*	5587*	6533*	6562*	6665*	6670*	7154*	7217*	7221*	7296*	7300*	8236*							
		8532*	8549*	8558*	9181*	9447*	9460*	9462*	10382												
A.CONT	003040	3580#	10392																		
A.NORM	003034	3578#	6118*	6167*	7155*	9182*	9448*	10386													
BADDRV	007366	4286#	7224*	7225*	7226	7235*	7236*	7237	7243*	7244*	7245	7263*	7264*	7265							
BADSEC=	001000	4236#	8145	8368	8821	8829	9393	9492													
BAD632	013314	4671#	7260																		
BAI =	000020	3257#	10551																		
BA16 =	000400	3233#																			
BA17 =	001000	3234#																			
BDSCHD	037616	8532	8549	8563#																	
BDSECT	011251	4466#	8582																		
BDSRCK	040670	8143	8366	8811#	9392																
BGNRTY	044512	9105	9458	9464	9485#																
BIT0 =	000001	2148#	3230	3247	3273	3295	3443	3587	3680	4892	4990	6956	7735	8019							
		8036	8051	8090	8107	8161	8203	8308	8323	8354	8390	8434	9064	10666							
BIT00 =	000001	2138#	2148																		
BIT01 =	000002	2137#	2147																		
BIT02 =	000004	2136#	2146																		
BIT03 =	000010	2135#	2145																		
BIT04 =	000020	2134#	2144																		
BIT05 =	000040	2133#	2143																		
BIT06 =	000100	2132#	2142																		
BIT07 =	000200	2131#	2141																		
BIT08 =	000400	2130#	2140	7860	7892																
BIT09 =	001000	2129#	2139	11270	11401																
BIT1 =	000002	2147#	3248	3275	3444	3588	3681	4228	5561	5609	8278	9068									
BIT10 =	002000	2128#	3235	3264	3286	3321	3335	3348	3363	3377	3456	3597	4237	9097							
		11248																			
BIT11 =	004000	2127#	3236	3265	3287	3306	3322	3336	3349	3364	3378	3457	4238	11408							
BIT12 =	010000	2126#	3238	3266	3288	3323	3337	3350	3351	3365	3379	3458	4239	8979							

DMD = 000040	3316#													
DONE 003113	3658#	8472*	8479	8857*	8864	8868*	8874	8919*	9061*	9175*	9253*	9518*	9521*	
DOTIM 003123	3666#													
DRA = 000001	3295#	9805	10588											
DRDY = 000200	3304#													
DRIEXR 012550	4603#	6432*												
DRISEL 013151	4651#	6600												
DRIV 011173	4456#	7798												
DRIVE 005500	3720#	5006*	5433*	5440*	5441	5444	5472	6066	6105	6116	6133	6336*	6430	
	6531*	6536	6539	6541*	6563*	6568*	6595	6597	6605	6618	6663*	6666	6677	
	6705*	6706	6723	7162	7163	7207	7224	7235	7243	7263	7294	7303	7311	
	7328	9250	12117											
DRNAFG 003127	3670#	8858*	8866	8869*	9074	9247*								
DRNAVL = 010000	4239#													
DRNRDY 011677	4525#	7313*												
DROPDR 011103	4446#	8950												
DROT = 000040	3301#	9320												
DRPAR = 000010	3278#	9260												
DRPDRV = 002000	3456#	10450												
DRVCAL 040774	5453	5459	5579	5582	5586	5589	5657	6000	6069	6087	6119	6123	6140	
	6168	6384	6389	6394	6439	6447	6454	6459	6464	6520	6527	6538	6580	
	6590	6594	6609	6611	6615	6624	6669	6686	6692	6697	6704	7161	7219	
	7250	7257	7270	7298	7322	7335	7337	7349	7362	7364	7376	7392	7797	
	7839	7925	8239	8280	8283	8534	8551	8739	8854#	9461	9503	9513	9548	
	9590													
DRVDC = 000040	3448#	9428	9890	10056	10118									
DRVRS 003120	3663#	4765*	5438*	8943*	8947	8949*								
DRVHRD = 000020	3447#	9423	9897	10092	10223									
DRVLST 005606	3754#	4893	4912	4924	4971	4979*	4991*	4994*	5004	5015*	5445	6064	6103	
	6131													
DRVLUP 025524	6512#	6548	6560	6616	6637									
DRVMSK = 000007	3254#	9799	9840	9991	10436									
DRVNED 012043	4544#	7305*												
DRVNO 011167	4455#	5472*	5473*											
DRVPDT = 000004	3445#	9998	10066											
DRVPOS = 000002	3444#	9998	10025	10487										
DRVSEL 013160	4653#	6599*												
DRVSEQ 007350	4283#	4889												
DRVSR 033602	5477	6344	7794#											
DRVSZD = 010000	3458#	9241	9828	10590										
DRVTST 014532	4705	4855	4871#	4877	4878	4879	4887	4944	4945	5039	5067	5083	5110	
	5160	5214	5239	5266	5294	5343	5415	7693	8228					
DRVUSE = 000001	3443#	10450												
DRY = 000200	3303#													
DR. CLR = 000005	3226#	9878	10041	10083	10107	10221								
DR. SEL = 000001	3225#	9948	10027	10342	10349	10356	10363	10567						
DROXDP 007673	4323#	7214												
DSC = 040000	3308#													
DSWR = 177570	2069#	2286	4741	6246										
DTBAII = 100000	3461#	5982	6085	10450	10549									
DTE = 010000	3288#	5672	8563	9325	10218									
DTYE = 000040	3280#	9288	10218											
DT100 063570	2413	2419	2425	2431	2437	2443	2449	2455	2461	2467	2473	2479	2485	
	2491	2497	2503	2509	2515	2521	2527	2533	2539	2545	2551	2557	2563	
	2569	2575	2581	2599	2605	2611	2617	2623	2629	2635	2641	2647	2653	
	2659	2665	2671	2677	2683	2689	2707	2719	2725	2731	2737	2743	2749	

EM34	057637	2573	2873	11743#																		
EM35	057657	2579	11746#																			
EM36	057713	2585	11751#																			
EM37	057746	2591	11756#																			
EM4	057004	2429	11662#																			
EM40	060011	2597	11762#																			
EM41	060055	2603	11769#																			
EM42	060113	2609	11775#																			
EM43	060143	2615	11780#																			
EM5	057025	2435	11665#																			
EM52	060200	2657	11785#																			
EM53	060226	2663	11789#																			
EM56	060253	2681	2687	11793#																		
EM6	057044	2441	11668#																			
EM60	060272	2693	11796#																			
EM61	060326	2699	11801#																			
EM62	060374	2705	11808#																			
EM63	060420	2621	2717	11812#																		
EM64	060465	2627	2723	11819#																		
EM65	060530	2633	2729	11825#																		
EM66	060556	2639	2735	11829#																		
EM67	060611	2645	2741	11834#																		
EM7	057070	2447	11672#																			
EM70	060650	2651	2747	11840#																		
EM71	060717	2753	11847#																			
EM72	060757	2759	11853#																			
EM73	061026	2765	11860#																			
EM74	061062	2771	11865#																			
EM75	061102	2777	11868#																			
EM76	061132	2783	11872#																			
EM77	061214	2789	2795	11881#																		
ENBCSR	030030	4763	4874	5156	6950#	6996																
ENTDRV	011633	4518#	6324																			
ENTHED	012724	4622#	6376																			
ENTLCI	010111	4351#	5037																			
ENTMOD	012416	4586#	6348	6477																		
ENTPAS	011035	4439#	5413																			
EQUALS	013442	4693#	7595																			
ERRCNT	003116	3661#	4704*	4854*	6290*	7389*	8922*	8924	8927	8929*	9062*	9176*	9474*	9494								
		9497																				
ERRCOM	005476	3719#	9113*	9114*	12117																	
ERRFRE	041220	3578	6167	6169	7155	8919#	9448															
ERRHDL	042466	3579	5587	5674	6562	6670	6718	7154	7221	7300	8236	8558	8565	9174#								
		9447	9462																			
ERRLMT	003117	3662#	9494																			
ERRVEC=	000004	2151#	4739	4740*	4751*	4759*	4760*	6244	6245*	6256*	6935*	6936*	6951	6952								
		6953*	6962*	6963*	6973	6974	6977*	6978*	6993*	6994*	7000*	8475*	8490*	11388								
		11389*	11391*	11394*	11502	11503	11507*	11512*	11524*	11529*	11538*	11548*	11549*									
ER2ENT	042506	9178#	9240	9358	9383	9467																
EXPLAN	010243	4368#	5154																			
E.CCLR=	000001	3587#	10157																			
E.CERR=	001000	3596#	10231																			
E.CLAT=	000020	3591#	9883	10047	10088	10112																
E.CMTO=	040000	3598#																				
E.CONT	003042	3581#	9064	9068	9072	9079	9083	9087	9091	9094	9097	9100	9104*	9782*								
		9820*	9883*	9893*	9964*	9986*	10009*	10015*	10047*	10088*	10112*	10157*	10209*	10231*								

PLOFST	005544	3737*	5575*	5607*	5627	5631												
PMA	005602	3749*	5733*	5734*	5812*	5813*	5901*	5902*	7023	7025	8031	8102	8182	8184				
		8302	8344	8411	8413	8688*	8689*	8690*	8691	8692	8712*	8713*	8716*	8717*				
PRDFLT	005674	3807*	5226	5277	5280	5319	5360											
PREPAR	030334	5735	5814	5903	7021*	8697	8755											
PREPKB	027302	5060	5207	5219	5244	5521	5707	5784	5872	5973	6056	6304	6325	6349				
		6377	6436	6478	6506	6530	6643	6683	6700	6781*	7368	8255						
PRMBUF	013430	4691*	7563*	7564														
PRMINP	007216	4265*	4818															
PRMLIM	005722	3823*	7727	7729	7737	7740	7742	7745										
PRMLST	005646	3792*	5225	5278	5318	5359	7565	7569	7732*	7748*	7749*							
PRMNEM	005776	3850*	5220	5299	5355	7563	7567	7623	7720									
PRMPHO	005254	3711*	8895*	12132														
PRMPLO	005252	3710*	8896*	12132														
PRMPSP	013451	4695*	5340															
PROMPT	013446	4694*	4941	5107	5158	5291	7646											
PRVCMO	005222	3708*	4786	6285	8884	9147												
PR0	= 000000	2085*	4785	5436	6275	7393	9512	9589										
PR1	= 000040	2086*																
PR2	= 000100	2087*																
PR3	= 000140	2088*																
PR4	= 000200	2089*	4778	6271														
PR5	= 000240	2090*	3577															
PR6	= 000300	2091*																
PR7	= 000340	2092*	4714	4762	4783	6219	6274	6936	6978	11340	11341							
PS	= 177776	2065*	2066	4714*	4785*	5436*	6219*	6275*	7393*	9702	9708*	9722*	10431	10432*				
		10691*																
PSTART	013510	2218	4711*															
PSW	= 177776	2066*	9512*	9589*														
PT	005666	3801*	5228	5881	5985	7626												
PWRMSG	055436	11343	11347*															
PWRVEC	= 000024	2157*	4729*	4730*	6234*	6235*	11331*	11339*	11340*									
P.ASOF	= 000032	3474*	9134	10044*	10045	10078*	10309*	10497*	10498									
P.A00	= 000040	3477*	7233	7241	7309	7326	9137	9552	10035*	10036	10052*	10053	10366*	10574*				
P.A01	= 000044	3479*	6591	6612	7365	9139	10345*											
P.A10	= 000050	3481*	9141	10352*														
P.A11	= 000054	3483*	7800	9143	10359*													
P.BAHI	= 000007	3436*	5729*	5808*	5897*	7047*	8694*	9123										
P.BALC	= 000010	3437*	5726*	5805*	5894*	5990*	6002*	6084*	7046*	7254*	7837*	8274*	8531*	8543*				
		8691*	9126	10531	10628													
P.BAR	= 000024	3471*	9133	10075*	10305*													
P.B00	= 000042	3478*	9138	10030*	10031	10367*	10575*											
P.B01	= 000046	3480*	9140	10346*														
P.B10	= 000052	3482*	6141	9142	9574	10353*												
P.B11	= 000056	3484*	9144	9951*	10360*													
P.CMND	= 000001	3430*	5452*	5578*	5581*	5583*	5588*	5656*	5741*	5745*	5821*	5825*	5916*	5920*				
		5993*	5999*	6004*	6058*	6086*	6115*	6122*	6126*	6166*	6383*	6388*	6393*	6435*				
		6446*	6453*	6458*	6463*	6519*	6526*	6537*	6579*	6589*	6593*	6608*	6610*	6614*				
		6623*	6667*	6681*	6691*	6696*	6703*	7160*	7216*	7249*	7251*	7269*	7295*	7321*				
		7334*	7336*	7347*	7361*	7363*	7375*	7391*	7796*	7834*	7924*	8238*	8277*	8282*				
		8522*	9114	9459*	9502*	9547*	9588*	9846	10463	10467	10479	10489	10495	10501				
		10504	10510	10535	10542	10561	10608	10625	10647	10656								
P.CS1	= 000016	3468*	6161*	7179	9127	9255	9273	9612	9645	9867*	9919*	9957*	9958*	9960*				
		9961	10070*	10072*	10299*	10302*	10453	10507*	10508*	10510*	10514*	10515	10539*	10542*				
		10544*	10545*	10555*	10556	10564*	10565*	10567*	10568	10570*	10571	10576	10615*	10616*				
		10618*	10621*	10622	10638*	10639*	10641*	10644	10663*	10664	10673*	10674						

K04

MD-11-DZR6N-C - RK611/RK06 SUBSYS. VERIF. : PART 2
DZR6NC.P11 05-OCT-76 10:07

MACY11 27(1006) 05-OCT-76 10:59 PAGE 258
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0256

P.CS1H= 000007	3435#	5454	5456*	5458*	7164*	7256*	7830*	8521*	9585*	9587*	9591*	9592*	9957
P.CS2 = 000020	10198	10507	10544	10564	10615	10629	10638						
P.CYLN= 000002	3469#	5669	6716	9128	9202	9207	9212	9214	9220	9222	9280	9330	9414
	9873	10073*	10303*	10461*	10486	10551*	10552	10563	10582	10586	10612*	10613	10637
	3431#	5527*	5537*	5540*	5541	5546*	5555	5557*	5565*	5618	5646	5648*	5654*
	5975*	6061*	6078*	6108*	6117*	6438*	6685*	7252*	7348*	7350*	7351	7835*	7991
	8443*	8448*	8523*	8625	8679*	9116	9582*	9955	10491	10533	10635		
P.DCYL= 000030	3473#	8629	8675	9129	9387	9602	9634	10077*	10311*				
P.DRVN= 000000	3429#	6068*	6107*	6116*	6139*	6514*	6536*	6588*	6605*	6618*	6666*	7162*	7207*
	7294*	9843	10020	10196	10435								
P.DS = 000036	3476#	9135	9265	9269	9320	10080*	10307*	10588					
P.DTS = 000026	3472#	8631	8633	8676	8677	9130	9388	9390	9573	9605	9606	9635	9641
	10076*	10306*											
P.EPAT= 000062	3486#	9411	9869*	10455									
P.EPOS= 000060	3485#	9410	9870*										
P.ER = 000034	3475#	5672	8563	9136	9201	10079*	10308*						
P.OFST= 000006	3434#	5580*	10497										
P.PRST= 000014	3439#	5982*	6085*	9228	9241	9423	9428	9433	9438	9549	9715*	9828*	9862*
	9890*	9897	9901*	9998*	10021	10025*	10038*	10056*	10066*	10092	10096*	10106*	10118*
	10223*	10375*	10450*	10487*	10511*	10512	10537	10543*	10549	10553	10580	10590*	10598
	10619	10642	10651	10685									
P.SECT= 000004	3432#	5528*	5550	5552*	5558*	5566*	5623	5642	5644*	5649*	5977*	6079*	7833*
	7918	7929*	7968	8444*	8528*	8537*	8538	8544*	8554*	8555	8627	8681*	9119
	9581*	9956	10492	10534	10636								
P.TRCK= 000005	3433#	5529*	5543	5547*	5548	5553*	5564*	5612	5651*	5652	5976*	6409*	6630*
	7253*	7836*	7919*	7921	7926*	7927	7982	8445*	8524*	8626	8680*	9117	
P.WC = 000012	3438#	5721*	5725*	5800*	5804*	5889*	5893*	5978*	5981*	6080*	6083*	7255*	7838*
	8275*	8499	8530*	8684*	8710	8747	9121	10532					
P.WCR = 000022	3470#	9131	10074*	10304*									
QNEWSW= 000000	3189												
Q.INIT= 040000	3460#												
RCDASW= 000010	3683#												
RCLREQ= 004000	4238#	9258	9263	9308	9313	9318	9323	9457					
RDALHD= 000164	3218#	9846	10625										
RDCHR = 104406	6295	7458	11642#										
RDCHRS 032272	4876	4928	4943	5038	5082	5109	5159	5265	5293	5342	5414	6858	7452#
	7630	7648											
RDDATA= 000121	3207#	5745	5825	5920	6004	7251	7834	8522					
RDGATE= 100000	3326#												
RDHDD 045062	9345	9370	9571#										
RDHEAD= 000125	3209#	5452	9588	9960	10641								
RDSTAT= 000141	3217#	6126	6589	6610	7216	7295	7363	7796	9547	10561			
RDY = 000200	3232#	6158	9453	10518	10571								
RECAL = 000113	3204#	5656	6383	6388	6393	6446	6453	6458	6463	6519	6526	6691	6696
	6703	7269	7334	8238	9459	10504							
RECODE 005474	3718#	5997	6120	7258	7388*	8145	8368	8740	8743	8821*	8829*	8920	8930*
	9177*	9178	9205*	9210*	9231	9233	9258*	9263*	9303*	9308*	9313*	9318*	9323*
	9328*	9336*	9340	9346	9361*	9365	9371	9386*	9393	9396*	9400*	9403*	9416*
	9417*	9426*	9431*	9436*	9441*	9449	9455*	9457	9463	9465*	9485	9487*	9492
	9499*	9519*	9551*										
REDBSF 037354	5471	8519#											
REFNEM 037204	5994	6005	8470#										
REISSU 003130	3671#	9076*											
RELEAS= 000140	3216#	10608											
REPLPK 007502	4301#	4926											
REPSUP 042206	6010	6137	6151	6162	6931	6975	8082	8348	8540	9111#	9183		

SW04 = 000020	2106#	2116			
SW05 = 000040	2105#	2115			
SW06 = 000100	2104#	2114			
SW07 = 000200	2103#	2113			
SW08 = 000400	2102#	2112			
SW09 = 001000	2101#	2111			
SW1 = 000002	2119#				
SW10 = 002000	2100#				
SW11 = 004000	2099#				
SW12 = 010000	2098#				
SW13 = 020000	2097#	8952			
SW14 = 040000	2096#				
SW15 = 100000	2095#				
SW2 = 000004	2118#				
SW3 = 000010	2117#				
SW4 = 000020	2116#				
SW5 = 000040	2115#				
SW6 = 000100	2114#				
SW7 = 000200	2113#				
SW8 = 000400	2112#				
SW9 = 001000	2111#				
S.ACLO= 000100	3344#				
S.BRHM= 000100	3359#				
S.BRKE= 040000	3381#				
S.CART= 000400	3361#				
S.DCLO= 010000	3351#				
S.DIB = 002000	3377#				
S.DOOR= 000200	3360#				
S.DRA = 000040	3330#				
S.DROT= 020000	3352#				
S.DRY = 000200	3332#	7233	7309		
S.DSC = 040000	3339#	9888	10036	10053	10116
S.FLT = 000200	3345#	10031			
S.FORM= 001000	3334#				
S.FWD = 002000	3363#				
S.HDFL= 000200	3374#				
S.HDHM= 000040	3358#	6591	6612	7365	
S.ICYL= 000040	3343#				
S.ILF = 000400	3346#				
S.LIMD= 020000	3380#				
S.LOAD= 010000	3365#				
S.MHD = 000400	3375#				
S.NMOV= 010000	3379#				
S.OFF = 002000	3335#				
S.PAR = 001000	3347#	9891	10368		
S.PIP = 020000	3338#	9899	10094		
S.PLO = 004000	3378#				
S.REV = 004000	3364#				
S.RTZ = 020000	3366#				
S.SECT= 000020	3371#				
S.SKI = 002000	3348#				
S.SPIN= 010000	3337#				
S.SPLS= 010000	3350#				
S.SPOK= 001000	3362#				
S.TYPE= 000400	3333#				
S.UNLD= 040000	3367#				

F05

\$MAMS2	001354	2379#																		
\$MAMS3	001360	2382#																		
\$MAMS4	001364	2385#																		
\$MBADR	001002	2251#																		
\$MFLG	056204	11429*	11435	11470*	11474#															
\$MNEW	055337	11323#																		
\$MSGAD	001334	2357#	11445*	11448																
\$MSGLG	001336	2358#	11450*																	
\$MSGTY	001320	2351#	11443	11451*	11463	11467*														
\$MSWR	055326	11321#																		
\$MTYP1	001351	2372#																		
\$MTYP2	001355	2380#																		
\$MTYP3	001361	2383#																		
\$MTYP4	001365	2386#																		
\$MXCNT	055740	11414	11424#																	
\$NULL	001154	2292#	10841	10870																
\$NWTST=	000001	5486#	5488	5680#	5682	5760#	5762	5848#	5850	5936#	5938	6022#	6024							
\$OCNT	054550	11118*	11147*	11160#																
\$OCTVL	054054	7801	10975	11000#																
\$OMODE	054552	11113*	11117*	11122	11125*	11136*	11162#													
\$OVER	055724	11384	11404	11412	11421#															
\$PASS	001326	2354#	4753*	5431*	5475	6191*	6192*	6200	6213	6258*	7414									
\$PASTM	001006	2253#																		
\$PWRC	055434	11336*	11337*	11346#																
\$PWDRN	055350	4729	6234	11331#	11339															
\$PWDRUP	055362	11331	11336#																	
\$QUES	001314	2342#	4851	4886	4956	5045	5091	5150	5166	5274	5323	5349	5423	6798						
		6868	7516	7639	7685	10870	11282													
		7862	7887	7942	10766#															
\$RAND	053150	11298#	11642																	
\$RDCHR	055174	11643																		
\$RDDEC=	*****	11643																		
\$RDLIN=	*****	11643																		
\$RDOCT=	*****	11643																		
\$RDSZ =	000000	11319#																		
\$REGAD	001160	2296#																		
\$REGO	001162	2298#	9115	12117																
\$REG1	001164	2299#	12117																	
\$REG10	001202	2306#	6937*	6982*	8135*	8136*	8137	8175*	8178	8358*	8359*	8360	8404*	8407						
		9350*	9375*	9410*	12121	12129														
\$REG11	001204	2307#	8176*	8405*	9351*	9376*	9411*	12121	12129											
\$REG12	001206	2308#	8177*	8406*	9352*	9377*	9404*	9406*	12121	12129										
\$REG13	001210	2309#	8179*	8181*	8183*	8184*	8408*	8410*	8412*	8413*	12121	12129								
\$REG14	001212	2310#	6149*	8178*	8180*	8182*	8407*	8409*	8411*	12124	12129									
\$REG15	001214	2311#	6150*	8185*	8186*	8414*	8415*	12124	12129											
\$REG16	001216	2312#	8114*	8375*	12125	12129														
\$REG17	001220	2313#	12125																	
\$REG2	001166	2300#	12117																	
\$REG20	001222	2314#	12125																	
\$REG21	001224	2315#	12125																	
\$REG22	001226	2316#	12125																	
\$REG23	001230	2317#	12125																	
\$REG24	001232	2318#	12125																	
\$REG25	001234	2319#	12125																	
\$REG26	001236	2320#	8986	9146																
\$REG27	001240	2321#																		
\$REG3	001170	2301#	12117																	

U
U
U

TYPNUM	2163#																	
TYPOCS	2163#																	
TYPOCT	2163#																	
TYPTXT	2163#																	
SDECBN	1#																	
SOCTBN	1#	10699																
SSCMRE	2259#	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311			
	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326			
	2327	2328	2329															
SSCMTM	2259#	2330	2331	2332	2333	2334	2335	2336	2337	2338								
SSESCA	2163#																	
SSNEWT	2163#	5486	5680	5760	5848	5936	6022											
SSSET	11627#	11636	11637	11638	11639	11642	11643	11644	11645									
SSSETM	4753#	6258#																
SSSKIP	2163#																	
.EQUAT	2010#	2053																
.HEADE	2010#	2020																
.KT11	2010#	2163																
.SETUP	2010#	4698																
.SWRHI	2010#	2030																
.SWRLO	2010#	2042#	2043	2044														
.SACT1	2010#	2223																
.SAPT8	2346#																	
.SAPTH	2010#	2234																
.SAPTY	2010#	11425																
.SCATC	2010#	2204																
.SCMTA	2010#	2259																
.SDB2D	2010#	11001																
.SDB20	2010#	10962																
.SEOP	2010#	6175																
.SERRO	2010#	11230																
.SPOWE	2010#																	
.SRAND	2010#	10755																
.SREAD	2010#	11282																
.SSAVE	2010#	11559																
.SSCOP	2010#	11367																
.SSIZE	2010#	11482																
.SSUPR	2010#	11063																
.STRAP	2010#	11604																
.STYPD	2010#	11163																
.STYPE	2010#	10791																
.STYPO	2010#	11086																

. ABS. 067426 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DSKZ:DZR6NC.DSKZ:DZR6NC.SEG/SOL/CRF/NL:TOC/DOC=DRIVE9.P11/EQ:QNEWSW,DZR6NC.P11
RUN-TIME: 113 111 10 SECONDS
RUN-TIME RATIO: 900/236=3.8
CORE USED: 50K (99 PAGES)

DOCUMENT PAGES: 268

K05

MD-11-DZREN-C - RK611/RK06 SUBSYS. VERIF. : PART 2 MACY11 27(1006) 05-OCT-76 10:59 PAGE 272
DZREN.C.F11 05-OCT-76 10:07 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0269