

RM03

FUNCTIONAL TEST PART 2 MD-11-DZRMD-A

EP-DZRMD-A-DL-A
COPYRIGHT © 1977

OCT 1977
digital
MADE IN USA

FICHE 1 OF 2

This image shows a grid of 144 small test cards, arranged in 12 rows and 12 columns. Each card contains technical data, including alphanumeric strings and small diagrams. The cards are organized into a structured layout, likely representing a functional test sequence for the MD-11-DZRMD-A system. The data on the cards includes various alphanumeric strings, possibly representing test parameters, component identifiers, or test results. Some cards also feature small diagrams or flowcharts, which may illustrate the test procedure or the internal logic of the system being tested. The overall appearance is that of a comprehensive test manual or a set of test instructions for a specific piece of equipment.



B01

EOF1DZRMCASEQ
DZRMCA.P11

00010000 770804
DZRMCA - RMD3 FUNCTIONAL TEST, PART 2

PDP10 411

HR1DZRMASEQ

00010000

770804

MACY11 30(1046) 29-JUL-77 15:01 PAGE 1
29-JUL-77 14:07

.REM

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZRMCA-A-D
PRODUCT NAME:	RMD3 FUNCTIONAL TEST, PART 2
DATE CREATED:	1 AUGUST 77
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	DOUG RIIKONEN

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURSHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1977, DIGITAL EQUIPMENT CORPORATION

.PAGE

CONTENTS

- 1. INTRODUCTION
 - 1. ABSTRACT
 - 2. UNIT UNDER TEST
- 2. OPERATING REQUIREMENTS
 - 1. HARDWARE REQUIREMENTS
 - 2. MEDIA REQUIREMENTS
 - 3. PREREQUISITE DIAGNOSTIC PROGRAMS
- 3. OPERATING PROCEDURE
 - 1. LOADING
 - 2. SWITCH OPTIONS
 - 3. STARTING
 - 4. HALTING
 - 5. RESTARTING
- 4. OPERATOR INTERFACE
 - 1. PROGRAM I.D.
 - 2. CONSOLE DIALOGUE
 - 3. PROGRESS REPORTS
 - 4. PERFORMANCE REPORTS
 - 5. PROGRAM HALTS
 - 6. ERROR REPORTS
- 5. ENVIRONMENTAL SUPPORT
 - 1. PROCESSOR COMPATIBILITY

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

DO1

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 3
DZRMDA.P11 29-JUL-77 14:07

SEQ 0005

109

2. DUAL PORT CONFIGURATIONS

EO1

110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

- 3. MEMORY PARITY HARDWARE
 - 4. MEMORY MANAGEMENT HARDWARE
 - 5. ACT,APT COMPATIBILITY
 - 6. XXDP COMPATIBILITY
 - 7. OPERATING SYSTEM COMPATIBILITY
-
- 6. TEST DESCRIPTION

126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181

1.0 INTRODUCTION

1.1 ABSTRACT

THE RMD3 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RMD3 DISK SUBSYSTEM. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

TO EXPLICITLY ESTABLISH CONFIDENCE IN THE BASIC OPERATIONS OF THE DISK DRIVE, INCLUDING MECHANICAL POSITIONING AND DATA TRANSFER OPERATIONS;

TO IMPLICITLY ESTABLISH CONFIDENCE IN THE DRIVE/ADAPTER ELECTRICAL INTERFACE;

TO VERIFY THE FUNCTIONALITY OF THE RMD3 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

THE TEST IS COMPRISED OF 3 PARTS, WHICH WOULD NORMALLY BE RUN IN SEQUENCE, STARTING WITH PART 1. BRIEFLY, PART 1 TESTS HOUSEKEEPING AND MECHANICAL POSITIONING OPERATIONS; PART 2 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING HEADER AND DATA; PART 3 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING DATA.

```
*****
*
* NOTE: PARTS 2 AND 3 OF THE FUNCTIONAL TEST LEAVE
* 2 HEADER ERRORS ON THE MEDIA. THE PACK SHOULD BE
* REFORMATTED AFTER RUNNING THESE TESTS.
*
*****
```

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RMD3 DISK SUBSYSTEM WHICH CONSISTS OF THE RHXX MASSBUS CONTROLLER, THE RMD3 MASSBUS ADAPTER AND THE STORAGE MODULE DISK DRIVE. NOTE THAT A DISK PACK IS REQUIRED FOR TESTING AND IS CONSIDERED AN INTEGRAL OF THE STORAGE MODULE DISK DRIVE.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RMD3 SUBSYSTEM FUNCTIONAL TEST:

PDP-11 PROCESSOR

GO1

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 6
DZRMDA.P11 29-JUL-77 14:07

SEQ 0008

182
183
184
185
186
187
188
189
190

16K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
UNIT UNDER TEST,

WHERE THE UNIT UNDER TEST CONSISTS OF ONE TO EIGHT RMD3 ADAPTERS, DISK
DRIVES AND DISK PACKS.

191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246

2.2 MEDIA REQUIREMENTS

EACH UNIT BEING TESTED MUST BE LOADED WITH A SCRATCH DISK PACK BEFORE TESTING BEGINS ON THAT UNIT. THE PACK MAY BE FORMATTED OR UNFORMATTED, BUT MUST NOT CONTAIN NEEDED INFORMATION BECAUSE THE PACK WILL BE WRITTEN DURING THE TEST.

NOTE

WHEN THE PROGRAM IS LOADED VIA THE RMO3, DRIVE 0 IS NOT TESTED UNLESS THE OPERATOR SPECIFICALLY ENTERS THE DRIVE NUMBER DURING CONSOLE DIALOGUE.

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RMO3 DISKLESS DIAGNOSTIC, DZRMJ-A

3.0 OPERATING PROCEDURE

3.1 LOADING

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:

- . PAPER TAPE, USING THE STANDARD PAPER TAPE LOADING PROCEDURE;
- .XXDP MEDIA, USING THE APPROPRIATE LOADING DEVICE.

NOTE

WHEN THE PROGRAM IS LOADED VIA THE RMO3, DRIVE 0 IS NOT TESTED UNLESS THE OPERATOR SPECIFICALLY ENTERS THE DRIVE NUMBER DURING CONSOLE DIALOGUE.

3.2 SWITCH OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE PROVIDED TO ENHANCE THE UTILITY OF THE PROGRAM.

I01

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 8
DZRMDA.P11 29-JUL-77 14:07

SEQ 0010

247
248

SW15 HALT ON ERROR
SW14 LOOP ON TEST (CURRENTLY BEING EXECUTED)

249
250
251
252 SW13 INHIBIT ERROR TYPEOUTS
253 SW12 UNUSED
254 SW11 INHIBIT TEST ITERATIONS
255 SW10 BELL ON ERROR
256 SW09 LOOP ON ERROR
257 SW08 LOOP ON TEST IN SW07-00
258

259 THE LOW ORDER 8 SWITCHES (SW07-SW00) ARE USED IN CONJUNCTION WITH
260 SW08 TO SPECIFY THE OCTAL NUMBER OF THE TEST WHICH THE PROGRAM WILL
261 LOOP ON.
262
263
264
265

266 3.3 STARTING

267 THE PROGRAM STARTS AT LOCATION 200, WHICH PROVIDES WORST CASE
268 TEST CONDITIONS IF RUNNING IN AN AUTOMATIC ENVIRONMENT. IF RUNNING IN
269 A STAND-ALONE ENVIRONMENT, THE PROGRAM USES CONSOLE DIALOGUE TO ALLOW
270 THE OPERATOR TO CONTROL TEST CONDITIONS.
271

272 3.4 HALTING

273 THE PROGRAM CAN BE HALTED BY TYPING CONTROL C FROM THE CONSOLE.
274
275

276 3.5 RESTARTING

277 THE PROGRAM CAN BE RESTARTED AT ADDRESS 200.
278
279

280 4.0 OPERATOR INTERFACE

281 4.1 PROGRAM ID

282 THE PROGRAM TYPES ITS TITLE AND MAINDEC NUMBER THE FIRST TIME IT
283 IS STARTED AFTER BEING LOADED. PROGRAM IDENTIFICATION DOES NOT OCCUR
284 IF THE PROGRAM IS RESTARTED.
285

286 4.2 CONSOLE DIALOGUE

287 WHEN THE PROGRAM IS RUNNING IN STAND ALONE MODE, IT ENTERS A
288 CONSOLE DIALOGUE SEQUENCE AFTER TYPING THE PROGRAM I.D..
289

290 THE FIRST QUESTION TYPED OUT IS: "TYPE HELP TEXT (Y OR N)??".
291 IF THE OPERATOR RESPONDS WITH A Y, THE PROGRAM WILL TYPE A BRIEF HELP
292 MESSAGE WHICH WILL LIST SWITCH OPTIONS, ETC.
293
294
295
296
297
298
299
300
301
302
303
304

305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360

THE SECOND QUESTION TYPED OUT IS, "CHANGE RMD3 UNIBUS ADDRESS OR VECTOR ADDRESS (Y OR N)??". IF THE UNIBUS ADDRESS OF THE RMD3 IS NON STANDARD, THE OPERATOR SHOULD RESPOND Y, THEN ANSWER SUBSEQUENT QUESTIONS TO SPECIFY THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY. IF THE OPERATOR DOES NOT RESPOND WITH A Y, THE PROGRAM SKIPS TO THE THIRD QUESTION.

THE THIRD QUESTION TYPED OUT IS, "TYPE (A) TO TEST ALL DEVICES, OR TYPE DEVICE NUMBER(S). TERMINATE INPUT WITH CARRIAGE RETURN". IF THE OPERATOR TYPES A, ALL POSSIBLE DEVICES ARE TESTED. OTHERWISE, THE OPERATOR CAN TYPE THE NUMBER(S) OF THE DEVICE(S) HE WANTS TESTED, AND TERMINATE HIS INPUT WITH A CARRIAGE RETURN.

IF THE PROGRAM IS RESTARTED, THE FIRST QUESTION TYPED IS, "USE SAME DEVICES (Y OR N)??". IF THE OPERATOR TYPES Y, THE TEST IS RESTARTED USING THE SAME DEVICES AS THE LAST TIME, OTHERWISE, THE TEST RESTARTS THE DIALOGUE AS IF THE PROGRAM WERE STARTED FOR THE FIRST TIME.

4.3 PROGRESS REPORTS

AN END OF PASS REPORT OCCURS EACH TIME THE PROGRAM IS EXECUTED FOR ALL DEVICES IN THE TEST QUEUE. THE END OF PASS REPORT INCLUDES A MESSAGE AND AN ERROR SUMMARY.

4.4 PERFORMANCE REPORT

NO PERFORMANCE REPORTS ARE GIVEN DURING THE EXECUTION OF THE PROGRAM.

4.5 PROGRAM HALTS

THERE ARE NO SCHEDULED HALTS DURING THE EXECUTION OF THE PROGRAM. PROCESSOR HALTS ARE DUE TO THE TRAP CATCHER.

4.6 ERROR REPORTS

THE FIRST LINE OF THE ERROR REPORT CONTAINS THE NUMBER OF THE UNIT BEING TESTED, THE TEST NUMBER, THE ERROR NUMBER AND THE VALUE OF THE PROGRAM COUNTER WHERE THE ERROR WAS CALLED. THIS LINE IS FOLLOWED BY THE ERROR MESSAGE: ONE OR MORE LINES OF TEXT WHICH GIVE A BRIEF, YET COMPREHENSIVE DESCRIPTION OF THE ERROR. THE ERROR MESSAGE IS NORMALLY FOLLOWED BY ONE OR MORE PAIRS OF LINES CONTAINING DATA HEADERS AND DATA PERTINENT TO THE ERROR, INCLUDING EXPECTED AND ACTUAL TEST RESULTS.

361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET, AND PROVIDING THAT DATA THROUGHPUT ON THE SYSTEM IS SUFFICIENT TO SUSTAIN DATA TRANSFER OPERATIONS.

5.2 DUAL PORT CONFIGURATIONS

THE RMO3 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RMO3 ADAPTER BUT IS EXECUTABLE ON RMO3 SUBSYSTEMS HAVING THE DUAL PORT OPTION PROVIDING THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE IS NOT USED DURING THE EXECUTION OF THE RMO3 SUBSYSTEM FUNCTIONAL TEST.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE IS NOT USED DURING THE RMO3 SUBSYSTEM FUNCTIONAL TEST. CAPABILITIES OF THE MASSBUS CONTROLLER.

5.5 ACT11, APT11 COMPATIBILITY

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES. FURTHER, THE PROGRAM WILL EXECUTE A QUICK PASS DURING THE FIRST PASS IN SUPPORT OF QUICK VERIFY MODE.

5.6 XXDP COMPATIBILITY

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RMO3 IS THE XXDP LOADING DEVICE.

MO1

412
413
414
415
416
417

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473

6.0 TEST DESCRIPTION

CONTROLLER ACCESS TEST

PURPOSE:

TO VERIFY THAT THE UNIBUS ADDRESS OF THE RMO3 SUBSYSTEM IS CORRECT, AS DEFINED AT LOCATION \$BASE.

PROCEDURE:

THE TEST TRIES TO ACCESS ALL MASSBUS CONTROLLER REGISTERS USING THE \$BASE ADDRESS. REGISTER CONTENTS ARE IGNORED DURING THE TEST, AND THE TEST FAILS IF A BUS TIMEOUT OCCURS FOR ANY REGISTER TRANSFER.

IF THE TEST FAILS AND THE PROGRAM IS RUNNING IN A STAND ALONE ENVIRONMENT, I.E., LOCATION 42 IS 0, THE PROGRAM WILL JUMP TO LOCATION 204 WHICH ALLOWS THE OPERATOR TO CHANGE THE \$BASE ADDRESS VIA CONSOLE DIALOGUE. OTHERWISE, THE PROGRAM ESCAPES TO THE END OF PASS HANDLER.

DEVICE AVAILABLE TEST

PURPOSE:

TO VERIFY THAT THE UNIT UNDER TEST IS AVAILABLE FOR TESTING, AND IS NOT LOCKED OR PROGRAMMED TO THE ALTERNATE PORT.

PROCEDURE:

THIS TEST SELECTS THE DEVICE AND READS CONTROL STATUS REGISTERS 1 AND 2 TO VERIFY THAT THE SELECTED DEVICE IS AVAILABLE FOR TESTING, AS INDICATED BY DVA STATUS, BIT 11 OF RMCS1 AND NED STATUS, BIT 12 OF RMCS2. THE RESULTS OF THE TEST CAN VARY AS FOLLOWS:

.NONEXISTANT DEVICE - THE DEVICE IS NONEXISTENT OR IS LOCKED ON THE ALTERNATE PORT AND IS THEREFORE NOT AVAILABLE FOR TESTING;

.DEVICE NOT AVAILABLE - THE DEVICE EXISTS BUT IS SEIZED BY THE ALTERNATE PORT AND IS NOT AVAILABLE FOR TESTING;

.BUS TIMEOUT - THE MASSBUS CONTROLLER FAILED TO DETECT A

B02

DZRM0A - RMD3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 14
DZRM0A.P11 29-JUL-77 14:07

SEQ 0016

474

NONEXISTENT DEVICE;

475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530

.DEVICE AVAILABLE - THE DEVICE IS AVAILABLE FOR TESTING.

THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH SELECTS THE NEXT DRIVE TO BE TESTED IF THE DEVICE IS NOT AVAILABLE.

DRIVE TYPE TEST

PURPOSE:

TO VERIFY THAT THE UNIT UNDER TEST IS AN RMD3 SINGLE PORT OR DUAL PORT SUBSYSTEM.

PROCEDURE:

THIS TEST READS THE DRIVE TYPE REGISTER, RMDT, OF THE SELECTED DEVICE AND VERIFIES THAT THE DEVICE IS A SINGLE PORT OR DUAL PORT RMD3 SUBSYSTEM. IF THE SELECTED DEVICE IS NOT AN RMD3, THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH WILL SELECT THE NEXT DEVICE FOR TESTING.

WRITE/READ HEADER AND DATA (FORMAT) TESTS

PURPOSE:

TO TEST WRITE HEADER AND DATA AND READ HEADER AND DATA FUNCTIONALITY OF THE RMD3 SUBSYSTEM USING A SET OF VARIABLES WHICH INCLUDE WORD COUNT, HEAD MOTION, HEAD SWITCHING AND ERROR CONDITIONS.

PROCEDURE:

ALTHOUGH EACH TEST EXERCISES A DIFFERENT VARIABLE, THE GENERAL PROCEDURE OF EACH TEST IS THE SAME. THE DRIVE IS INITIALIZED AND RECALIBRATED IF "PIP" OR "SKI" ARE ACTIVE SO THAT THERE ARE NO ERRORS WHEN A TEST BEGINS. FOLLOWING THAT, THE TEST PERFORMS ANY EXPLICIT SEEKS REQUIRED FOR THE CONDITIONS OF THE TEST. REGISTERS ARE PRESET AND THE WRITE HEADER AND DATA COMMAND IS EXECUTED. WHEN THE WRITE COMMAND IS COMPLETE, THE TEST STORES ALL SUBSYSTEM STATUS AND CHECKS FOR PRIMARY ERRORS WHICH PRECLUDE OTHER STATUS CHECKS. IF THERE ARE NO PRIMARY ERRORS, THE TEST VERIFIES THE RESULTS OF THE WRITE COMMAND AND THEN CHECKS FOR SECONDARY ERRORS. LOOP ADDRESSES ARE MODIFIED FOLLOWING THE SUCCESSFUL COMPLETION OF THE WRITE COMMAND IN ORDER TO SHORTEN EXECUTION TIMES AND ENHANCE SCOPING LOOPS. THEN THE PROGRAM EXECUTES THE READ HEADER AND DATA PORTION OF THE TEST, VERIFYING

531
532
533
534
535
536
537
538
539
540
541
542
543
544
545

THE SAME TYPE OF ERRORS AS IN THE WRITE COMMAND.

NOTE THAT THE SECTOR USED DURING A TEST MAY DIFFER FROM THE PROGRAM LISTING BECAUSE THE PROGRAM SUBSTITUTES A GOOD SECTOR IF THE ONE SELECTED IS LISTED IN THE BAD BLOCK TABLE OF THE LAST TRACK. UNLESS SPECIFIED OTHERWISE, ALL TESTS ARE IN 16 BIT (32 SECTOR) FORMAT.

FORMAT ZEROS - 18

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

THE TEST SEEKS TO CYLINDER 0, THEN WRITES HEADER AND DATA ON SECTOR 0 IN 18 BIT FORMAT. THE HEADER AND DATA FIELDS ARE ALL ZEROS, CAUSING THE DEVICE TO USE NORMAL WRITE GATE. THE HEADER AND DATA ARE READ AND COMPARED WITH THE WRITE BUFFER. THE INITIAL SEEK POSITIONS THE HEAD SUCH THAT THERE IS NO IMPLIED SEEK.

FORMAT ZEROS - 16

THIS TEST IS THE SAME AS THE PREVIOUS TEST, EXCEPT THAT DATA IS WRITTEN IN 16 BIT FORMAT.

ZERO FILL TEST

THE TEST EXECUTES A SEEK TO CYLINDER 0 TO INSURE THAT THERE IS NO HEAD MOTION DURING DATA TRANSFER. THIS IS FOLLOWED BY A WRITE HEADER AND DATA COMMAND WITH THE WORD COUNT EQUAL TO THE SIZE OF THE HEADER WHICH CAUSES THE RM70 TO ZERO FILL THE DATA FIELD. THE READ HEADER AND DATA COMMAND THAT FOLLOWS READS A FULL SECTOR AND VERIFIES THAT DATA WAS ZERO FILLED.

FORMAT CHECK ZEROS

THE TEST WRITES HEADER AND AN ALL ZEROS DATA FIELD, THEN PERFORMS A WRITE CHECK HEADER AND DATA COMMAND AND VERIFIES THERE ARE NO ERRORS.

FORMAT CHECK ZEROS W/ WCE ERROR

THE TEST WRITES HEADER AND AN ALL ZEROS DATA FIELD. AFTER COMPLEMENTING THE LAST WORD OF THE WRITE BUFFER, THE TEST PERFORMS A WRITE CHECK HEADER AND DATA COMMAND AND VERIFIES THAT THE CORRECT WRITE CHECK ERROR IS DETECTED.

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
651
652
653
654
655
656

FORMAT ONES

THE TEST WRITES HEADER AND AN ALL ONES DATA FIELD, THEN READS THE HEADER AND DATA, VERIFYING THE READ BUFFER WITH THE WRITE BUFFER. THE ALL ONES FIELD IS A CONSTANT FREQUENCY, AND THE DRIVE SHOULD USE NORMAL WRITE GATE.

FORMAT CHECK ONES

THE TEST FORMATS AN ALL ONES DATA FIELD, THEN PERFORMS A WRITE CHECK HEADER AND DATA COMMAND, VERIFYING THAT THERE ARE NO ERRORS.

FORMAT CHECK ONES W/ WCE ERROR

THE TEST FORMATS AN ALL ONES DATA FIELD, THEN COMPLEMENTS THE LAST WORD OF THE WRITE BUFFER. A WRITE CHECK HEADER AND DATA COMMAND IS EXECUTED, AND THE TEST VERIFIES THAT THE CORRECT WRITE CHECK ERROR IS DETECTED.

FORMAT MULTIPLE SECTORS

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE WRITE HEADER AND DATA COMMAND FOLLOWS, WITH THE WORD COUNT EQUAL TO MULTIPLE SECTORS. THE WORD COUNT DURING THE READ HEADER AND DATA PORTION OF THE TEST IS SET FOR ALL OF THE FIRST SECTOR AND THE HEADER OF THE SECOND SECTOR.

FORMAT WITH HEAD SWITCHING

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE WRITE HEADER AND DATA COMMAND STARTS WITH CYLINDER 0, TRACK 0, SECTOR 31. THE WORD COUNT IS EQUAL TO MULTIPLE SECTORS WHICH CAUSES THE SUBSYSTEM TO SWITCH

G02

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 19
DZRMDA.P11 29-JUL-77 14:07

SEQ 0021

657
658

FROM TRACK 0 TO TRACK 1 AFTER THE FIRST SECTOR IS WRITTEN. THE
READ HEADER AND DATA COMMAND USES THE SAME WORD COUNT AND

659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712

STARTING SECTOR.

FORMAT WITH IMPLIED SEEK

THIS TEST SEEKS TO THE LAST CYLINDER PRIOR TO WRITING HEADER AND DATA ON CYLINDER 0. THE EXPLICIT SEEK INSURES THAT THERE WILL BE MAXIMUM HEAD MOTION DURING THE IMPLIED SEEK OF THE WRITE COMMAND. THE SAME OPERATION, INCLUDING THE EXPLICIT SEEK IS REPEATED FOR READ HEADER AND DATA.

FORMAT WITH MIDTRANSFER SEEK

THIS TEST WRITES MULTIPLE SECTORS STARTING WITH CYLINDER 0, TRACK 4, SECTOR 31, CAUSING A MIDTRANSFER SEEK AFTER THE FIRST SECTOR IS WRITTEN. THE SAME SECTORS ARE READ WITH READ HEADER AND DATA COMMAND.

FORMAT EACH SECTOR ADDRESS

HEADERS AND DATA OF EACH SECTOR ON CYLINDER 0, TRACK 0 ARE FORMATTED AND READ WITH THE PROGRAM VERIFYING HEADERS AND DATA.

FORMAT EACH TRACK ADDRESS

THIS TEST FORMATS SECTOR 0 OF EACH TRACK ON CYLINDER 0 AND READS EACH SECTOR WITH THE PROGRAM VERIFYING HEADERS AND DATA.

FORMAT PRIME CYLINDERS

THIS TEST FORMATS AND READS SECTOR 0, TRACK 0 ON EACH PRIME CYLINDER, I.E., CYLINDERS 1,2,4,8,....,512.

713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766

FORMAT LAST SECTOR

THIS TEST READS HEADER AND DATA ON THE LAST SECTOR, I.E. CYLINDER 822, TRACK 4, SECTOR 31, AND VERIFIES THAT LBT STATUS SETS.

FOR W/ AOE ERROR

THIS TEST READS MULTIPLE SECTORS STARTING WITH THE LAST SECTOR AND VERIFIES THAT AOE STATUS SETS.

FORMAT INVALID SECTOR ADDRESS

THIS TEST USES AN ILLEGAL SECTOR ADDRESS AND VERIFIES THAT IAE STATUS SETS.

FORMAT INVALID TRACK ADDRESS

THIS TEST USES AN ILLEGAL TRACK ADDRESS AND VERIFIES THAT IAE STATUS SETS.

FORMAT INVALID CYLINDER ADDRESS

THIS TEST USES AN ILLEGAL CYLINDER ADDRESS AND VERIFIES THAT IAE STATUS SETS.

FORMAT AT OFFSET

THE PROGRAM SETS OFFSET MODE AND EXECUTES A WRITE HEADER AND DATA COMMAND, VERIFYING THAT OFFSET MODE IS RESET BY THE WRITE COMMAND.

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

IVC FORMAT TEST

VOLUME VALID IS RESET BY SETTING AND RESETTING DIAGNOSTIC MODE. THE TEST THEN EXECUTES A WRITE HEADER AND DATA COMMAND AND VERIFIES THAT INVALID COMMAND STATUS SETS. THE TEST IS REPEATED FOR READ HEADER AND DATA COMMAND.

FORMAT ERROR TEST(18 AND 16)

A SINGLE SECTOR IS FORMATTED WITH THE OFFSET REGISTER SET FOR 16 BIT FORMAT AFTER WHICH THE SAME SECTOR IS READ IN 18 BIT FORMAT WITH THE PROGRAM VERIFYING THAT FORMAT ERROR STATUS IS SET. THE SAME PROCEDURE IS REPEATED WITH THE SECTOR WRITTEN IN 18 BIT FORMAT AND READ IN 16 BIT FORMAT.

FORMAT HCE (FIRST AND SECOND HEADER WORDS)

THESE TWO TESTS WRITE AN INCORRECT HEADER THEN READ THE HEADER AND VERIFY THAT THE CORRECT HEADER ERROR IS DETECTED. THE TESTS SETUP THE CORRECT HEADER, THEN COMPLEMENT BIT 0 AND USE THE MODIFIED BUFFER TO WRITE THE HEADER. THE PROCESS IS REPEATED UNTIL EACH BIT POSITION HAS BEEN SEPARATELY TESTED.

K02

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 23
DZRMDA.P11 29-JUL-77 14:07

SEQ 0025

806

```

807 ;PROGRAM REVISION #001
808
809 .TITLE DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
810 ;*COPYRIGHT (C) 1977
811 ;*DIGITAL EQUIPMENT CORP.
812 ;*MAYNARD, MASS. 01754
813 ;*
814 ;*PROGRAM BY DOUG RIIKONEN
815 ;*
816 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
817 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
818
819 000001 $TN=1
820 .SBTTL OPERATIONAL SWITCH SETTINGS
821 ;*
822 ;* SWITCH USE
823 ;* -----
824 ;* 15 HALT ON ERROR
825 ;* 14 LOOP ON TEST
826 ;* 13 INHIBIT ERROR TYPEOUTS
827 ;* 12 ENABLE EXTENDED STATUS
828 ;* 11 INHIBIT ITERATIONS
829 ;* 10 BELL ON ERROR
830 ;* 9 LOOP ON ERROR
831 ;* 8 LOOP ON TEST IN SWR<7:0>
832 ;* 7 TN128
833 ;* 6 TN64
834 ;* 5 TN32
835 ;* 4 TN16
836 ;* 3 TN8
837 ;* 2 TN4
838 ;* 1 TN2
839 ;* 0 TN1
840 .SBTTL BASIC DEFINITIONS
841
842 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
843 001100 STACK= 1100
844 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
845 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
846
847 ;*MISCELLANEOUS DEFINITIONS
848 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
849 000012 LF= 12 ;;CODE FOR LINE FEED
850 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
851 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
852 177776 PS= 177776 ;;PROCESSOR STATUS WORD
853 .EQUIV PS,PSW
854 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
855 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
856 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
857 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
858
859 ;*GENERAL PURPOSE REGISTER DEFINITIONS
860 000000 R0= %0 ;;GENERAL REGISTER
861 000001 R1= %1 ;;GENERAL REGISTER
862 000002 R2= %2 ;;GENERAL REGISTER
    
```

863	000003	R3=	%3	::	GENERAL REGISTER
864	000004	R4=	%4	::	GENERAL REGISTER
865	000005	R5=	%5	::	GENERAL REGISTER
866	000006	R6=	%6	::	GENERAL REGISTER
867	000007	R7=	%7	::	GENERAL REGISTER
868	000006	SP=	%6	::	STACK POINTER
869	000007	PC=	%7	::	PROGRAM COUNTER
870					
871		:*PRIORITY LEVEL DEFINITIONS			
872	000000	PR0=	0	::	PRIORITY LEVEL 0
873	000040	PR1=	40	::	PRIORITY LEVEL 1
874	000100	PR2=	100	::	PRIORITY LEVEL 2
875	000140	PR3=	140	::	PRIORITY LEVEL 3
876	000200	PR4=	200	::	PRIORITY LEVEL 4
877	000240	PR5=	240	::	PRIORITY LEVEL 5
878	000300	PR6=	300	::	PRIORITY LEVEL 6
879	000340	PR7=	340	::	PRIORITY LEVEL 7
880					
881		:*"SWITCH REGISTER" SWITCH DEFINITIONS			
882	100000	SW15=	100000		
883	040000	SW14=	40000		
884	020000	SW13=	20000		
885	010000	SW12=	10000		
886	004000	SW11=	4000		
887	002000	SW10=	2000		
888	001000	SW09=	1000		
889	000400	SW08=	400		
890	000200	SW07=	200		
891	000100	SW06=	100		
892	000040	SW05=	40		
893	000020	SW04=	20		
894	000010	SW03=	10		
895	000004	SW02=	4		
896	000002	SW01=	2		
897	000001	SW00=	1		
898		.EQUIV	SW09, SW9		
899		.EQUIV	SW08, SW8		
900		.EQUIV	SW07, SW7		
901		.EQUIV	SW06, SW6		
902		.EQUIV	SW05, SW5		
903		.EQUIV	SW04, SW4		
904		.EQUIV	SW03, SW3		
905		.EQUIV	SW02, SW2		
906		.EQUIV	SW01, SW1		
907		.EQUIV	SW00, SW0		
908					
909		:*DATA BIT DEFINITIONS (BIT00 TO BIT15)			
910	100000	BIT15=	100000		
911	040000	BIT14=	40000		
912	020000	BIT13=	20000		
913	010000	BIT12=	10000		
914	004000	BIT11=	4000		
915	002000	BIT10=	2000		
916	001000	BIT09=	1000		
917	000400	BIT08=	400		
918	000200	BIT07=	200		

919 000100
 920 000040
 921 000020
 922 000010
 923 000004
 924 000002
 925 000001
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938 000004
 939 000010
 940 000014
 941 000014
 942 000014
 943 000020
 944 000024
 945 000030
 946 000034
 947 000060
 948 000064
 949 000240
 950
 951
 952
 953
 954
 955 004000
 956 000040
 957 000020
 958 000010
 959 000004
 960 000002
 961 000001
 962 000077
 963
 964
 965
 966 000000
 967 000002
 968 000004
 969 000006
 970 000010
 971 000012
 972 000014
 973 000016
 974 000020

BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 BIT00= 1
 .EQUIV BIT09,BIT9
 .EQUIV BIT08,BIT8
 .EQUIV BIT07,BIT7
 .EQUIV BIT06,BIT6
 .EQUIV BIT05,BIT5
 .EQUIV BIT04,BIT4
 .EQUIV BIT03,BIT3
 .EQUIV BIT02,BIT2
 .EQUIV BIT01,BIT1
 .EQUIV BIT00,BIT0

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

.SBTTL RM03 REGISTER BIT DEFINITIONS

;RMCS1 CONTROL STATUS REGISTER

DVA = BIT11 ; DEVICE AVAILABLE-READ ONLY
 F4 = BIT05 ; FUNCTION CODE
 F3 = BIT04 ; FUNCTION CODE
 F2 = BIT03 ; FUNCTION CODE
 F1 = BIT02 ; FUNCTION CODE
 F0 = BIT01 ; FUNCTION CODE
 GO = BIT00 ; GO BIT
 FNCMSK = 000077 ; FUNCTION CODE MASK

;FUNCTION CODES (BITS 01-05 OF RMCS1)

NOP = 000000 ; NOP COMMAND
 ILF02 = 000002 ; ILLEGAL COMMAND
 SEEK = 000004 ; SEEK COMMAND
 RECAL = 000006 ; RECALIBRATE COMMAND
 DRVCLR = 000010 ; DRIVE CLEAR COMMAND
 RELEASE = 000012 ; RELEASE COMMAND
 OFFSET = 000014 ; OFFSET COMMAND
 RTC = 000016 ; RETURN TO CENTERLINE COMMAND
 RIP = 000020 ; READ IN PRESET COMMAND

975	000022	PAKACK	=	000022	;PACK ACKNOWLEDGE COMMAND
976	000022	PACACK	=	PAKACK	
977	000024	ILF24	=	000024	; ILLEGAL COMMAND
978	000026	ILF26	=	000026	; ILLEGAL COMMAND
979	000030	SEARCH	=	000030	;SEARCH COMMAND
980	000030	ILF30	=	000030	; ILLEGAL COMMAND
981	000032	ILF32	=	000032	; ILLEGAL COMMAND
982	000034	ILF34	=	000034	; ILLEGAL COMMAND
983	000036	ILF36	=	000036	; ILLEGAL COMMAND
984	000040	ILF40	=	000040	; ILLEGAL COMMAND
985	000042	ILF42	=	000042	; ILLEGAL COMMAND
986	000044	ILF44	=	000044	; ILLEGAL COMMAND
987	000046	ILF46	=	000046	; ILLEGAL COMMAND
988	000050	WCD	=	000050	;WRITE CHECK DATA COMMAND
989	000052	WCH	=	000052	;WRITE CHECK HEADER AND DATA
990	000054	ILF54	=	000054	; ILLEGAL COMMAND
991	000056	ILF56	=	000056	; ILLEGAL COMMAND
992	000060	WD	=	000060	;WRITE DATA COMMAND
993	000062	WH	=	000062	;WRITE HEADER AND DATA COMMAND
994	000064	ILF64	=	000064	; ILLEGAL COMMAND
995	000066	ILF66	=	000066	; ILLEGAL COMMAND
996	000070	PD	=	000070	;READ DATA COMMAND
997	000072	RH	=	000072	;READ HEADER AND DATA COMMAND
998	000074	ILF74	=	000074	; ILLEGAL COMMAND
999	000076	ILF76	=	000076	; ILLEGAL COMMAND
1000					
1001		;RMDA		DISK ADDRESS REGISTER	
1002					
1003	002000	TA4	=	BIT10	; TRACK ADDRESS 4
1004	001000	TA2	=	BIT09	; TRACK ADDRESS 2
1005	000400	TA1	=	BIT08	; TRACK ADDRESS 1
1006	000020	SA16	=	BIT04	; SECTOR ADDRESS 16
1007	000010	SAB	=	BIT03	; SECTOR ADDRESS 8
1008	000004	SA4	=	BIT02	; SECTOR ADDRESS 4
1009	000002	SA2	=	BIT01	; SECTOR ADDRESS 2
1010	000001	SA1	=	BIT00	; SECTOR ADDRESS 1
1011					
1012		; TRACK, SECTOR MASKS			
1013					
1014	003400	TAMSK	=	003400	; TRACK ADDRESS MASK
1015	000037	SAMSK	=	000037	; SECTOR ADDRESS MASK
1016					
1017		;RMD5		DRIVE STATUS REGISTER	
1018					
1019	100000	ATA	=	BIT15	; ATTENTION ACTIVE
1020	040000	ERR	=	BIT14	; COMPOSITE ERROR
1021	020000	PIP	=	BIT13	; POSITIONING IN PROGRESS
1022	010000	MOL	=	BIT12	; MEDIUM ON LINE
1023	004000	WRL	=	BIT11	; WRITE LOCK
1024	002000	LBT	=	BIT10	; LAST BLOCK TRANSFERRED
1025	001000	PGM	=	BIT09	; PROGRAMMABLE
1026	000400	DPR	=	BIT08	; DRIVE PRESENT
1027	000200	DRY	=	BIT07	; DRIVE READY
1028	000100	VV	=	BIT06	; VOLUME VALID
1029	000001	OM	=	BIT00	; OFFSET MODE ACTIVE
1030					

```

1031 ;RMR1 ERROR REGISTER #1
1032
1033 100000 DCK = BIT15 ;DATA CHECK ERROR
1034 040000 UNS = BIT14 ;DRIVE UNSAFE
1035 020000 OPI = BIT13 ;OPERATION INCOMPLETE
1036 010000 DTE = BIT12 ;DRIVE TIMING ERROR
1037 004000 WLE = BIT11 ;WRITE LOCK ERROR
1038 002000 IAE = BIT10 ;INVALID ADDRESS ERROR
1039 001000 AOE = BIT09 ;ADDRESS OVERFLOW ERROR
1040 000400 HCRC = BIT08 ;HEADER CRC ERROR
1041 000200 HCE = BIT07 ;HEADER COMPARE ERROR
1042 000100 ECH = BIT06 ;ECC "HARD" ERROR
1043 000040 WCF = BIT05 ;WRITE CLOCK FAILURE
1044 000020 FER = BIT04 ;FORMAT ERROR
1045 000010 PAR = BIT03 ;PARITY ERROR
1046 000004 RMR = BIT02 ;REGISTER MODIFICATION REFUSED
1047 000002 ILR = BIT01 ;ILLEGAL REGISTER
1048 000001 ILF = BIT00 ;ILLEGAL FUNCTION
1049
1050 115760 NDTMSK = DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER
1051 ;"NDTMSK" IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
1052 ;COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
1053
1054 ;RMAS ATTENTION SUMMARY REGISTER
1055
1056 000377 ATNMSK = 377 ;MASK FOR ATTENTION BITS
1057
1058 ;RMLA LOOK AHEAD REGISTER
1059
1060 002000 SC4 = BIT10 ;SECTOR COUNT = 16
1061 001000 SC3 = BIT09 ;SECTOR COUNT = 8
1062 000400 SC2 = BIT08 ;SECTOR COUNT = 4
1063 000200 SC1 = BIT07 ;SECTOR COUNT = 2
1064 000100 SC0 = BIT06 ;SECTOR COUNT = 1
1065
1066 003700 SCTMSK = 003700 ;SECTOR COUNT MASK
1067
1068 ;RMR MAINTENANCE REGISTER
1069
1070 ; WRITE ONLY BITS
1071
1072 100000 DBCK = BIT15 ;DEBUG CLOCK
1073 040000 DBEN = BIT14 ;DEBUG CLOCK ENABLE
1074 020000 DEBL = BIT13 ;DIAGNOSTIC END OF BLOCK
1075 010000 DTO = BIT12 ;DIAGNOSTIC TIMEOUT
1076 004000 MCLK = BIT11 ;MAINTENANCE CLOCK
1077 002000 MRD = BIT10 ;READ DATA
1078 001000 MUR = BIT09 ;UNIT READY
1079 000400 MOC = BIT08 ;ON CYLINDER
1080 000200 MSER = BIT07 ;SEEK ERROR
1081 000100 MDF = BIT06 ;DRIVE FAULT
1082 000040 MS = BIT05 ;SECTOR PULSE
1083 000010 MWP = BIT03 ;WRITE PROTECT
1084 000004 MI = BIT02 ;INDEX PULSE
1085 000002 MSC = BIT01 ;SECTOR COMPARE
1086 000001 DMD = BIT00 ;DIAGNOSTIC MODE

```

```

1087
1088
1089
1090      100000      OCC      =      BIT15      ;OCCUPIED
1091      040000      RG      =      BIT14      ;RUN AND GO
1092      020000      EBL      =      BIT13      ;END OF BLOCK
1093      010000      REX      =      BIT12      ;EXCEPTION
1094      004000      ESRC      =      BIT11      ;ENABLE SEARCH
1095      002000      PLFS      =      BIT10      ;LOOKING FOR SYNC
1096      001000      ECRC      =      BIT09      ;ENABLE CRC OUT
1097      000400      PDA      =      BIT08      ;DATA AREA
1098      000200      PHA      =      BIT07      ;HEADER AREA
1099      000100      CONT      =      BIT06      ;CONTINUE
1100      000040      WC      =      BIT05      ;WORD CLOCK
1101      000020      EECC      =      BIT04      ;ENABLE ECC OUT
1102      000010      MWD      =      BIT03      ;WRITE DATA BIT
1103      000004      LS      =      BIT02      ;LAST SECTOR
1104      000002      LST      =      BIT01      ;LAST SECTOR AND TRACK
1105      000001      DMD      =      BIT00      ;DIAGNOSTIC MODE
1106
1107      ;RMDT      DRIVE TYPE REGISTER
1108
1109      100000      NSA      =      BIT15      ;NOT SECTOR ADDRESSED=0
1110      040000      TAP      =      BIT14      ;TAPE DRIVE = 0
1111      020000      MOH      =      BIT13      ;MOVING HEAD = 1
1112      004000      DRQ      =      BIT11      ;DRIVE REQUEST REQUIRED
1113
1114      020024      SNGPRT      =      020024      ;SINGLE PORT DRIVE TYPE
1115      024024      DULPRT      =      024024      ;DUAL PORT DRIVE TYPE
1116
1117      ;RMOF      OFFSET REGISTER
1118
1119      010000      FMT16      =      BIT12      ;16 BIT WORD FORMAT
1120      004000      ECI      =      BIT11      ;ECC INHIBIT
1121      002000      HCI      =      BIT10      ;HEADER COMPARE INHIBIT
1122      000200      OFD      =      BIT07      ;OFFSET FORWARD
1123
1124
1125      ;RMDC      DESIRED CYLINDER ADDRESS REGISTER
1126      001777      CYLSK      =      1777      ;MASK FOR CYLINDER ADDRESS
1127
1128      ;RMR2      MAINTENANCE REGISTER #2
1129
1130      ;      READ ONLY BITS
1131      100000      RQA      =      BIT15      ;PORT A REQUEST
1132      040000      RQB      =      BIT14      ;PORT B REQUEST
1133      020000      TAG      =      BIT13      ;TAG CONTROL
1134      010000      TST      =      BIT12      ;COMMAND SEQUENCE TEST BIT
1135      004000      CC      =      BIT11      ;CONTROL OR CYLINDER TAG
1136      002000      CH      =      BIT10      ;CONTROL OR HEAD TAG
1137      001000      BB09      =      BIT09      ;TAG BUS
1138      000400      BB08      =      BIT08      ;TAG BUS
1139      000200      BB07      =      BIT07      ;TAG BUS
1140      000100      BB06      =      BIT06      ;TAG BUS
1141      000040      BB05      =      BIT05      ;TAG BUS
1142      000020      BB04      =      BIT04      ;TAG BUS

```

1143	000010	BB03	=	BIT03	;TAG BUS
1144	000004	BB02	=	BIT02	;TAG BUS
1145	000002	BB01	=	BIT01	;TAG BUS
1146	000001	BB00	=	BIT00	;TAG BUS
1147					
1148					
1149		;RMR2 ERROR REGISTER 2			
1150					
1151	100000	BSE	=	BIT15	;BAD SECTOR ERROR
1152	040000	SKI	=	BIT14	;SEEK INCOMPLETE
1153	020000	OPE	=	BIT13	;OPERATOR PLUG ERROR
1154	010000	IVC	=	BIT12	;INVALID COMMAND ERROR
1155	004000	LSC	=	BIT11	;LOSS OF SYSTEM CLOCK
1156	002000	LBC	=	BIT10	;LOSS OF BIT CLOCK
1157	000200	DVC	=	BIT07	;DEVICE CHECK
1158	000010	OPE	=	BIT03	;DATA PARITY ERROR
1159					
1160		.SBTTL PROGRAM MNEMONICS			
1161					
1162	100000	MSE	=	BIT15	;MANUFACTURING DETECTED SECTOR ERROR
1163	040000	USE	=	BIT14	;USER DETECTED SECTOR ERROR
1164					
1165		.SBTTL RMD3 REGISTER INDEX VALUES			
1166					
1167	000000	RMCS1	=	00	;CONTROL STATUS REGISTER
1168	000006	RMDA	=	06	;DISK ADDRESS REGISTER
1169	000012	RMD5	=	12	;DRIVE STATUS REGISTER
1170	000014	RMR1	=	14	;ERROR REGISTER 1
1171	000016	RMAS	=	16	;ATTENTION SUMMARY REGISTER
1172	000020	RMLA	=	20	;LOOK AHEAD REGISTER
1173	000024	RMR1	=	24	;MAINTENANCE REGISTER
1174	000026	RMDT	=	26	;DRIVE TYPE REGISTER
1175	000030	RMSN	=	30	;SERIAL NUMBER REGISTER
1176	000032	RMOF	=	32	;OFFSET REGISTER
1177	000034	RMDC	=	34	;DESIRED CYLINDER REGISTER
1178	000036	RMCC	=	36	;CURRENT CYLINDER REGISTER
1179	000040	RMR2	=	40	;MAINTENANCE REGISTER 2
1180	000042	RMR2	=	42	;ERROR REGISTER 2
1181	000044	RMEC1	=	44	;ECC POSITION REGISTER
1182	000046	RMEC2	=	46	;ECC PATTERN REGISTER
1183	000050	ILRG50	=	50	;ILLEGAL REGISTER 50
1184	000052	ILRG52	=	52	;ILLEGAL REGISTER 52
1185	000054	ILRG54	=	54	;ILLEGAL REGISTER 54
1186	000056	ILRG56	=	56	;ILLEGAL REGISTER 56
1187	000060	ILRG60	=	60	;ILLEGAL REGISTER 60
1188	000062	ILRG62	=	62	;ILLEGAL REGISTER 62
1189	000064	ILRG64	=	64	;ILLEGAL REGISTER 64
1190	000066	ILRG66	=	66	;ILLEGAL REGISTER 66
1191	000070	ILRG70	=	70	;ILLEGAL REGISTER 70
1192	000072	ILRG72	=	72	;ILLEGAL REGISTER 72
1193	000074	ILRG74	=	74	;ILLEGAL REGISTER 74
1194	000076	ILRG76	=	76	;ILLEGAL REGISTER 76
1195					
1196					
1197	000077	IDXMSK	=	77	;MASK FOR REGISTER INDEX NUMBER
1198					


```

1199 .SBTTL RH CONTROLLER REGISTER BIT DEFINITIONS
1200 ;RMCS1 CONTROL STATUS REGISTER #1
1201
1202 SC = BIT15 ;SPECIAL CONDITION-READ ONLY
1203 100000 TRE = BIT14 ;TRANSFER ERROR
1204 040000 MCPE = BIT13 ;MASSBUS CONTROL BUS PARITY
1205 020000 ;ERROR-READ ONLY
1206 PSEL = BIT10 ;PORT B SELECT
1207 002000 A17 = BIT09 ;ADDRESS EXTENSION
1208 001000 A16 = BIT08 ;ADDRESS EXTENSION
1209 000400 RDY = BIT07 ;READY-READ ONLY
1210 000200 IE = BIT06 ;INTERRUPT ENABLE
1211 000100
1212 ;RMCS2 RH CONTROL STATUS REGISTER #2
1213
1214 DLT = BIT15 ;DATA LATE-READ ONLY
1215 100000 WCE = BIT14 ;WRITE CHECK ERROR-READ ONLY
1216 040000 UPE = BIT13 ;UNIBUS PARITY ERROR
1217 020000 NED = BIT12 ;NONEXISTANT DRIVE-READ ONLY
1218 010000 NEM = BIT11 ;NONEXISTANT MEMORY-READ ONLY
1219 004000 PGE = BIT10 ;PROGRAM ERROR-READ ONLY
1220 002000 MXF = BIT09 ;MISSED TRANSFER
1221 001000 MDPE = BIT08 ;MASSBUS DATA BUS PARITY
1222 000400 ;ERROR-READ ONLY
1223 OR = BIT07 ;OUTPUT READY-READ ONLY
1224 000200 IR = BIT06 ;INPUT READY-READ ONLY
1225 000100 CLR = BIT05 ;CONTROLLER CLEAR
1226 000040 PAT = BIT04 ;PARITY TEST
1227 000020 BAI = BIT03 ;UNIBUS ADDRESS INCREMENT
1228 000010 ;INHIBIT
1229 U2 = BIT02 ;UNIT SELECT
1230 000004 U1 = BIT01 ;UNIT SELECT
1231 000002 U0 = BIT00 ;UNIT SELECT
1232 000001
1233 ;UNIT SELECT MASK
1234 UNTMSK = 7 ;UNIT SELECT MASK
1235 000007
1236 ;RMCS3 RH70 CONTROL STATUS REGISTER #3
1237
1238 APE = BIT15 ;ADDRESS PARITY ERROR
1239 100000 DPEHI = BIT14 ;DATA PARITY ERROR HIGH WORD
1240 040000 DPELO = BIT13 ;DATA PARITY ERROR LOW WORD
1241 020000 WCEHI = BIT12 ;WRITE CHECK ERROR HIGH WORD
1242 010000 WCELO = BIT11 ;WRITE CHECK ERROR LOW WORD
1243 004000 DBL = BIT10 ;DOUBLE WORD TRANSFER
1244 002000 IE = BIT06 ;INTERRUPT ENABLE
1245 000100 IPCK3 = BIT03 ;INVERT PARITY CHECK
1246 000040 IPCK2 = BIT02 ;INVERT PARITY CHECK
1247 000020 IPCK1 = BIT01 ;INVERT PARITY CHECK
1248 000004 IPCK0 = BIT00 ;INVERT PARITY CHECK
1249 000001
1250 .SBTTL RH CONTROLLER REGISTER INDEX VALUES
1251 RMCS1 = 00 ;CONTROL STATUS REGISTER
1252 RMWC = 02 ;WORD COUNT REGISTER
1253 RMBA = 04 ;BUS ADDRESS REGISTER
1254 RMCS2 = 10 ;CONTROLLER STATUS REGISTER

```

```

1255      000022      RMD8      =      22      ;DATA BUFFER
1256      000050      RMBAE      =      50      ;BUS ADDRESS EXTENSION
1257      000052      RMCS3      =      52      ;CONTROL STATUS REGISTER #3
1258
1259      176700      ABASE      =      176700      ;UNIBUS ADDRESS
1260      120254      AVECT1      =      120254      ;UNIBUS VECTOR ADDRESS AND PRIORITY
1261
1262
1263      .SBTTL TRAP CATCHER
1264
1265      000000      .=0
1266      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1267      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1268      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1269      000174      000174      .=174
1270      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
1271      000176      000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
1272
1273
1274      .SBTTL ACT11 HOOKS
1275
1276      ;*****
1277      ;HOOKS REQUIRED BY ACT11
1278      000200      $SVPC=.      ;SAVE PC
1279      000046      .=46
1280      000046      032452      SENDAD      ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
1281      000052      .=52
1282      000052      000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
1283      000200      .=$SVPC      ;; RESTORE PC
1284
1285      .SBTTL STARTING ADDRESS
1286
1287      ;THE PROGRAM STARTS AT LOCATION 200
1288      000200      000200      =      200
1289      000200      000137      005322      JMP      START      ;JUMP TO START OF PROGRAM
1290
1291      001100      .=1100
1292      .SBTTL APT PARAMETER BLOCK
1293
1294      ;*****
1295      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1296      ;*****
1297      001100      .SX=.      ;;SAVE CURRENT LOCATION
1298      000024      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1299      000024      000200      200      ;;FOR APT START UP
1300      000044      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
1301      000044      001100      $APTHDR ;;POINT TO APT HEADER BLOCK
1302      001100      .=$X      ;;RESET LOCATION COUNTER
1303      ;*****
1304      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1305      ;INTERFACE SPEC.
1306
1307      001100      $APTHD:
1308      001100      000000      $HIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1309      001102      001222      $MBADR: .WORD $MAIL ;; ADDRESS OF APT MAILBOX (BITS 0-15)
1310      001104      000001      $STMT:  .WORD 1      ;;RUN TIM OF LONGEST TEST
  
```

H03

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 33
APT PARAMETER BLOCK

SEQ 0035

1311 001106 000002
1312 001110 000002
1313 001112 000042
1314 001114

\$PASTM: .WORD 2 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 2 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
TAGADR = .WORD SETEND-SMAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

1315
1316
1317
1318
1319
1320
1321 001114
1322 001114 001114
1323 001114 000000
1324 001116 000
1325 001117 000
1326 001120 000000
1327 001122 000000
1328 001124 000000
1329 001126 000000
1330 001130 000
1331 001131 001
1332 001132 000000
1333 001134 000000
1334 001136 000000
1335 001140 000000
1336 001142 000000
1337 001144 000000
1338 001146 000000
1339 001150 000
1340 001151 000
1341 001152 000000
1342 001154 177570
1343 001156 177570
1344 001160 177560
1345 001162 177562
1346 001164 177564
1347 001166 177566
1348 001170 000
1349 001171 002
1350 001172 012
1351 001173 000
1352 001174 000000
1353 001176 000000
1354 001200 000000
1355 001202 000000
1356 001204 000000
1357 001206 000000
1358 001210 000000
1359 001212 177607 000377
1360 001216 077
1361 001217 015
1362 001220 000012
1363
1364
1365
1366
1367
1368 001222
1369 001222 000000
1370 001224 000000

```

.SBTTL COMMON TAGS

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

      . =TAGADR
SCMTAG:      ;; START OF COMMON TAGS
              .WORD 0
STSTNM:     .BYTE 0      ;; CONTAINS THE TEST NUMBER
SERFLG:     .BYTE 0      ;; CONTAINS ERROR FLAG
$ICNT:      .WORD 0      ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR:     .WORD 0      ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR:     .WORD 0      ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL:     .WORD 0      ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB:     .BYTE 0      ;; CONTAINS ITEM CONTROL BYTE
$ERMAX:     .BYTE 1      ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC:     .WORD 0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR:     .WORD 0      ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR:     .WORD 0      ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT:     .WORD 0      ;; CONTAINS 'GOOD' DATA
$BDDAT:     .WORD 0      ;; CONTAINS 'BAD' DATA
              .WORD 0      ;; RESERVED--NOT TO BE USED
              .WORD 0
$AUTOB:     .BYTE 0      ;; AUTOMATIC MODE INDICATOR
$INTAG:     .BYTE 0      ;; INTERRUPT MODE INDICATOR
              .WORD 0
$SWR:       .WORD DSWR   ;; ADDRESS OF SWITCH REGISTER
$DISPLAY:   .WORD DDISP  ;; ADDRESS OF DISPLAY REGISTER
$TKS:       177560      ;; TTY KBD STATUS
$TKB:       177562      ;; TTY KBD BUFFER
$TPS:       177564      ;; TTY PRINTER STATUS REG. ADDRESS
$TPB:       177566      ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL:     .BYTE 0      ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS:    .BYTE 2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC:    .BYTE 12     ;; INSERT FILL CHARS. AFTER A "LINE FEED"
$TPFLG:    .BYTE 0      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$TMP0:     .WORD 0      ;; USER DEFINED
$TMP1:     .WORD 0      ;; USER DEFINED
$TMP2:     .WORD 0      ;; USER DEFINED
$TMP3:     .WORD 0      ;; USER DEFINED
$TMP4:     .WORD 0      ;; 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

```

1371 001226 000000
 1372 001230 000000
 1373 001232 000000
 1374 001234 000000
 1375 001236 000000
 1376 001240 000000
 1377 001242
 1378 001242 000
 1379 001243 000
 1380 001244 000000
 1381 001246 000000
 1382 001250 000000
 1383
 1384
 1385
 1386
 1387
 1388
 1389 001252 000
 1390 001253 000
 1391
 1392
 1393
 1394
 1395 001254 000000
 1396
 1397 001256 000
 1398 001257 000
 1399 001260 000000
 1400 001262 000
 1401 001263 000
 1402 001264 000000
 1403 001266 000
 1404 001267 000
 1405 001270 000000
 1406 001272 120254
 1407 001274 000000
 1408 001276 176700
 1409 001300 000000
 1410 001302 000000
 1411 001304 000000
 1412 001306 000000
 1413 001310 000000
 1414 001312 000000
 1415 001314 000000
 1416 001316 000000
 1417 001320 000000
 1418 001322 000000
 1419 001324 000000
 1420 001326
 1421
 1422
 1423
 1424
 1425
 1426 001326

STESTN: .WORD ATESTN ;; TEST NUMBER
 SPASS: .WORD APASS ;; PASS COUNT
 SDEVCT: .WORD ADEVCT ;; DEVICE COUNT
 SUNIT: .WORD AUNIT ;; I/O UNIT NUMBER
 SMSGAD: .WORD AMSGAD ;; MESSAGE ADDRESS
 SMSGLG: .WORD AMSGLG ;; MESSAGE LENGTH
 SETABLE: .WORD ;; APT ENVIRONMENT TABLE
 SENV: .BYTE AENV ;; ENVIRONMENT BYTE
 SENVM: .BYTE AENVM ;; ENVIRONMENT MODE BITS
 SSWREG: .WORD ASWREG ;; APT SWITCH REGISTER
 SUSMR: .WORD AUSMR ;; USER SWITCHES
 SCPUOP: .WORD ACPUOP ;; CPU TYPE, OPTIONS
 ;;
 ;; BITS 15-11=CPU TYPE
 ;; 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
 ;; 11/70=06, P00=07, Q=10
 ;; BIT 10=REAL TIME CLOCK
 ;; BIT 9=FLOATING POINT PROCESSOR
 ;; BIT 8=MEMORY MANAGEMENT
 \$MAMS1: .BYTE AMAMS1 ;; HIGH ADDRESS, M.S. BYTE
 \$MTYP1: .BYTE AMTYP1 ;; MEM. TYPE, BLK#1
 ;; MEM. TYPE BYTE -- (HIGH BYTE)
 ;; 900 NSEC CORE=001
 ;; 300 NSEC BIPOLAR=002
 ;; 500 NSEC MOS=003
 \$MADR1: .WORD AMADR1 ;; HIGH ADDRESS, BLK#1
 ;; MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
 \$MAMS2: .BYTE AMAMS2 ;; HIGH ADDRESS, M.S. BYTE
 \$MTYP2: .BYTE AMTYP2 ;; MEM. TYPE, BLK#2
 \$MADR2: .WORD AMADR2 ;; MEM. LAST ADDRESS, BLK#2
 \$MAMS3: .BYTE AMAMS3 ;; HIGH ADDRESS, M.S. BYTE
 \$MTYP3: .BYTE AMTYP3 ;; MEM. TYPE, BLK#3
 \$MADR3: .WORD AMADR3 ;; MEM. LAST ADDRESS, BLK#3
 \$MAMS4: .BYTE AMAMS4 ;; HIGH ADDRESS, M.S. BYTE
 \$MTYP4: .BYTE AMTYP4 ;; MEM. TYPE, BLK#4
 \$MADR4: .WORD AMADR4 ;; MEM. LAST ADDRESS, BLK#4
 SVECT1: .WORD AVECT1 ;; INTERRUPT VECTOR#1, BUS PRIORITY#1
 SVECT2: .WORD AVECT2 ;; INTERRUPT VECTOR#2, BUS PRIORITY#2
 SBASE: .WORD ABASE ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
 SDEVN: .WORD ADEVN ;; DEVICE MAP
 SCDW1: .WORD ACDW1 ;; CONTROLLER DESCRIPTION WORD#1
 SCDW2: .WORD ACDW2 ;; CONTROLLER DESCRIPTION WORD#2
 SDDW0: .WORD ADDW0 ;; DEVICE DESCRIPTOR WORD#0
 SDDW1: .WORD ADDW1 ;; DEVICE DESCRIPTOR WORD#1
 SDDW2: .WORD ADDW2 ;; DEVICE DESCRIPTOR WORD#2
 SDDW3: .WORD ADDW3 ;; DEVICE DESCRIPTOR WORD#3
 SDDW4: .WORD ADDW4 ;; DEVICE DESCRIPTOR WORD#4
 SDDW5: .WORD ADDW5 ;; DEVICE DESCRIPTOR WORD#5
 SDDW6: .WORD ADDW6 ;; DEVICE DESCRIPTOR WORD#6
 SDDW7: .WORD ADDW7 ;; DEVICE DESCRIPTOR WORD#7
 SETEND:
 .MEXIT
 ;; THE REGISTER INPUT BUFFER IS USED FOR
 ;; STORING DRIVE STATUS
 GETBUF:

1427
1428
1429 001326 000000
1430 001330 000000
1431 001332 000000
1432 001334 000000
1433 001336 000000
1434 001340 000000
1435 001342 000000
1436 001344 000000
1437 001346 000000
1438 001350 000000
1439 001352 000000
1440 001354 000000
1441 001356 000000
1442 001360 000000
1443 001362 000000
1444 001364 000000
1445 001366 000000
1446 001370 000000
1447 001372 000000
1448 001374 000000

.REGISTER INPUT BUFFER

RMCS1I: .WORD ;CONTROL STATUS REGISTER
 RMWCI: .WORD ;WORD COUNT REGISTER
 RMBAI: .WORD ;BUS ADDRESS REGISTER
 RMDAI: .WORD ;DISK ADDRESS REGISTER
 RMCS2I: .WORD ;CONTROLLER STATUS REGISTER
 RMDSI: .WORD ;DRIVE STATUS REGISTER
 RMER1I: .WORD ;ERROR REGISTER 1
 RMA5I: .WORD ;ATTENTION SUMMARY REGISTER
 RMLAI: .WORD ;LOOK AHEAD REGISTER
 RMDBI: .WORD ;DATA BUFFER
 RMMR1I: .WORD ;MAINTENANCE REGISTER #1
 RMDTI: .WORD ;DRIVE TYPE REGISTER
 RMSNI: .WORD ;SERIAL NUMBER REGISTER
 RMOFI: .WORD ;OFFSET REGISTER
 RMDCI: .WORD ;DESIRED CYLINDER REGISTER
 RMCCI: .WORD ;CURRENT CYLINDER REGISTER
 RMMR2I: .WORD ;MAINTENANCE REGISTER #2
 RMER2I: .WORD ;ERROR REGISTER 2
 RMEC1I: .WORD ;ECC POSITION REGISTER
 RMEC2I: .WORD ;ECC PATTERN REGISTER

;THE REGISTER OUTPUT BUFFER IS USED FOR
;ASSEMBLING DATA GOING TO REGISTER

1449
1450
1451
1452
1453 001376

PUTBUF:

.REGISTER OUTPUT BUFFER

1454
1455
1456 001376 000000
1457 001400 000000
1458 001402 000000
1459 001404 000000
1460 001406 000000
1461 001410 000000
1462 001412 000000
1463 001414 000000
1464 001416 000000
1465 001420 000000
1466 001422 000000
1467 001424 000000
1468 001426 000000
1469 001430 000000
1470 001432 000000
1471 001434 000000
1472 001436 000000
1473 001440 000000
1474 001442 000000
1475 001444 000000

RMCS1O: .WORD ;CONTROL STATUS REGISTER
 RMWCO: .WORD ;WORD COUNT REGISTER
 RMBAO: .WORD ;BUS ADDRESS REGISTER
 RMDAO: .WORD ;DISK ADDRESS REGISTER
 RMCS2O: .WORD ;CONTROLLER STATUS REGISTER
 RMDSO: .WORD ;DRIVE STATUS REGISTER
 RMER1O: .WORD ;ERROR REGISTER 1
 RMA5O: .WORD ;ATTENTION SUMMARY REGISTER
 RMLAO: .WORD ;LOOK AHEAD REGISTER
 RMDBO: .WORD ;DATA BUFFER
 RMMR1O: .WORD ;MAINTENANCE REGISTER #1
 RMDTO: .WORD ;DRIVE TYPE REGISTER
 RMSNO: .WORD ;SERIAL NUMBER REGISTER
 RMOFO: .WORD ;OFFSET REGISTER
 RMDCO: .WORD ;DESIRED CYLINDER REGISTER
 RMCCO: .WORD ;CURRENT CYLINDER REGISTER
 RMMR2O: .WORD ;MAINTENANCE REGISTER #2
 RMER2O: .WORD ;ERROR REGISTER 2
 RMEC1O: .WORD ;ECC POSITION REGISTER
 RMEC2O: .WORD ;ECC PATTERN REGISTER

;EACH WORD OF THE TEST QUE CONTAINS THE DEVICE NUMBER IN
;THE LOW BYTE AND THE ATTENTION BIT IN THE HIGH BYTE. A ZERO
;WORD IS A BLANK AND REPRESENTS THE END OF THE QUE.

1476
1477
1478
1479
1480
1481 001446 000012
1482

TSTQUE: .BLKW 10. ;TEST QUE

1483				
1484				
1485	001472	000000		
1486				
1487				
1488				
1489	001474	000000		
1490	001476	000000		
1491	001500	000000		
1492	001502	000000		
1493				
1494				
1495				
1496				
1497	001504	000027		
1498				
1499				
1500				
1501				
1502	001533	000027		
1503				
1504				
1505				

;MEDIA ENABLE IS SET IF THE BAD SECTOR FILES HAVE BEEN RECOVERED
 ;FOR THE UNIT UNDER TEST, OTHERWISE IT IS ZERO.
 MEDENB: .WORD ;MEDIA ENABLE

;LOCATIONS "ASNDC" AND "ASNDC" CONTAIN THE CYLINDER, TRACK AND SECTOR
 ;ADDRESS ASSIGNED BY THE BAD SECTOR MODULE.
 ASNDC: .WORD ;ASSIGNED DESIRED CYLINDER
 ASNDA: .WORD ;ASSIGNED TRACK, AND SECTOR
 CLKADR: .WORD ;UNIBUS ADDRESS OF KW11 CLOCK
 CLKVCT: .WORD ;VECTOR ADDRESS OF KW11 CLOCK

;THE GET INDEX TABLE CONTAINS A BYTE LIST OF REGISTERS WHICH
 ;ARE READ BY THE GET SUBROUTINE. THE LIST IS TERMINATED BY
 ;A NEGATIVE BYTE.
 GETINX: .BLKB 23. ;GET INDEX TABLE

;THE PUT INDEX TABLE ICONTAINS A BYTE LIST OF REGISTERS WHICH
 ;ARE WRITTEN BY THE PUT SUBROUTINE. THE LIST IS TERMINATED BY
 ;A NEGATIVE BYTE.
 PUTINX: .BLKB 23. ;PUT INDEX TABLE

;PUT TAGS HERE

1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522

001562

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (\$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:

1523			;ERROR	1	WRONG UNIT SELECTED
1524	001562	064434		EMT1	
1525	001564	070526		EHT1	
1526	001566	070652		EDT1	
1527	001570	070742		EFT1	
1528					
1529					
1530			;ERROR	2	DEVICE WENT UNAVAILABLE
1531	001572	064440		EMT2	
1532	001574	070526		EHT1	
1533	001576	070652		EDT1	
1534	001600	070742		EFT1	
1535					
1536					
1537			;ERROR	3	DEVICE WENT NONEXISTENT
1538	001602	064446		EMT3	
1539	001604	070526		EHT1	
1540	001606	070652		EDT1	
1541	001610	070742		EFT1	
1542					
1543					
1544			;ERROR	4	CONTROLLER NOT READY
1545	001612	064454		EMT4	
1546	001614	070526		EHT1	
1547	001616	070652		EDT1	
1548	001620	070742		EFT1	
1549					
1550					
1551			;ERROR	5	DRIVE NOT READY AND GO NOT RESET
1552	001622	064462		EMT5	
1553	001624	070526		EHT1	
1554	001626	070652		EDT1	
1555	001630	070742		EFT1	
1556					
1557					
1558			;ERROR	6	UNEXPECTED VALUE FOR "ATA" STATUS
1559	001632	064470		EMT6	
1560	001634	070526		EHT1	
1561	001636	070652		EDT1	
1562	001640	070742		EFT1	
1563					
1564					
1565			;ERROR	7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER
1566	001642	064476		EMT7	
1567	001644	000000		0	
1568	001646	000000		0	
1569	001650	000000		0	
1570					
1571					
1572			;ERROR	10	DRIVE NOT READY BUT GO IS RESET
1573	001652	064504		EMT10	
1574	001654	070526		EHT1	
1575	001656	070652		EDT1	
1576	001660	070742		EFT1	
1577					
1578					

1579			;ERROR	11	GO NOT RESET BUT DRIVE IS READY
1580	001662	064510		EMT11	
1581	001664	070526		EHT1	
1582	001666	070652		EDT1	
1583	001670	070742		EFT1	
1584					
1585					
1586			;ERROR	12	INCORRECT FUNCTION CODE
1587	001672	064514		EMT12	
1588	001674	070526		EHT1	
1589	001676	070652		EDT1	
1590	001700	070742		EFT1	
1591					
1592					
1593			;ERROR	13	PARITY ERROR READING REMOTE REGISTERS
1594	001702	064522		EMT13	
1595	001704	070526		EHT1	
1596	001706	070652		EDT1	
1597	001710	070742		EFT1	
1598					
1599					
1600			;ERROR	14	TRANSFER ERROR IS INCORRECT
1601	001712	064534		EMT14	
1602	001714	070526		EHT1	
1603	001716	070652		EDT1	
1604	001720	070742		EFT1	
1605					
1606					
1607			;ERROR	15	INCORRECT WORD COUNT
1608	001722	064542		EMT15	
1609	001724	070526		EHT1	
1610	001726	070652		EDT1	
1611	001730	070742		EFT1	
1612					
1613					
1614			;ERROR	16	INCORRECT BUS ADDRESS
1615	001732	064550		EMT16	
1616	001734	070526		EHT1	
1617	001736	070652		EDT1	
1618	001740	070742		EFT1	
1619					
1620					
1621			;ERROR	17	INCORRECT LBT STATUS
1622	001742	064560		EMT17	
1623	001744	070526		EHT1	
1624	001746	070652		EDT1	
1625	001750	070742		EFT1	
1626					
1627					
1628			;ERROR	20	INCORRECT AOE
1629	001752	064570		EMT20	
1630	001754	070526		EHT1	
1631	001756	070652		EDT1	
1632	001760	070742		EFT1	
1633					
1634					

1635			;ERROR	21	INCORRECT DISK ADDRESS
1636	001762	064600		EMT21	
1637	001764	070526		EHT1	
1638	001766	070652		EDT1	
1639	001770	070742		EFT1	
1640					
1641					
1642			;ERROR	22	INCORRECT CYLINDER ADDRESS
1643	001772	064610		EMT22	
1644	001774	070526		EHT1	
1645	001776	070652		EDT1	
1646	002000	070742		EFT1	
1647					
1648					
1649			;ERROR	23	INCORRECT WLE STATUS
1650	002002	064620		EMT23	
1651	002004	070526		EHT1	
1652	002006	070652		EDT1	
1653	002010	070742		EFT1	
1654					
1655					
1656			;ERROR	24	INCORRECT UPE STATUS
1657	002012	064630		EMT24	
1658	002014	070526		EHT1	
1659	002016	070652		EDT1	
1660	002020	070742		EFT1	
1661					
1662					
1663			;ERROR	25	INCORRECT WCF STATUS
1664	002022	064640		EMT25	
1665	002024	070526		EHT1	
1666	002026	070652		EDT1	
1667	002030	070742		EFT1	
1668					
1669					
1670			;ERROR	26	INCORRECT WCE STATUS
1671	002032	064650		EMT26	
1672	002034	070526		EHT1	
1673	002036	070652		EDT1	
1674	002040	070742		EFT1	
1675					
1676					
1677			;ERROR	27	INCORRECT MDPE STATUS
1678	002042	064660		EMT27	
1679	002044	070526		EHT1	
1680	002046	070652		EDT1	
1681	002050	070742		EFT1	
1682					
1683					
1684			;ERROR	30	INCORRECT DCK STATUS
1685	002052	064670		EMT30	
1686	002054	070526		EHT1	
1687	002056	070652		EDT1	
1688	002060	070742		EFT1	
1689					
1690					

1691			;ERROR 31	INCORRECT ECH STATUS
1692	002062	064700	EMT31	
1693	002064	070526	EHT1	
1694	002066	070652	EDT1	
1695	002070	070742	EFT1	
1696				
1697				
1698			;ERROR 32	DLT SHOULD NOT BE SET
1699	002072	064710	EMT32	
1700	002074	070526	EHT1	
1701	002076	070652	EDT1	
1702	002100	070742	EFT1	
1703				
1704				
1705			;ERROR 33	MXF SHOULD NOT BE SET
1706	002102	064720	EMT33	
1707	002104	070526	EHT1	
1708	002106	070652	EDT1	
1709	002110	070742	EFT1	
1710				
1711				
1712			;ERROR 34	DTE SHOULD NOT BE SET
1713	002112	064730	EMT34	
1714	002114	070526	EHT1	
1715	002116	070652	EDT1	
1716	002120	070742	EFT1	
1717				
1718				
1719			;ERROR 35	INCORRECT HCRC STATUS
1720	002122	064740	EMT35	
1721	002124	070526	EHT1	
1722	002126	070652	EDT1	
1723	002130	070742	EFT1	
1724				
1725				
1726			;ERROR 36	INCORRECT HCE STATUS
1727	002132	064750	EMT36	
1728	002134	070526	EHT1	
1729	002136	070652	EDT1	
1730	002140	070742	EFT1	
1731				
1732				
1733			;ERROR 37	INCORRECT FER STATUS
1734	002142	064760	EMT37	
1735	002144	070526	EHT1	
1736	002146	070652	EDT1	
1737	002150	070742	EFT1	
1738				
1739				
1740			;ERROR 40	DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
1741	002152	064770	EMT40	
1742	002154	070526	EHT1	
1743	002156	070652	EDT1	
1744	002160	070742	EFT1	
1745				
1746				

E04

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 43
ERROR POINTER TABLE

SEQ 0045

1747			;ERROR 41	LOST "MOL" DURING PACK ACKNOWLEDGE
1748	002162	064776	EMT41	
1749	002164	070526	EHT1	
1750	002166	070652	EDT1	
1751	002170	070742	EFT1	
1752				
1753				
1754			;ERROR 42	UNSAFE ERROR DURING PACK ACKNOWLEDGE
1755	002172	065006	EMT42	
1756	002174	070526	EHT1	
1757	002176	070652	EDT1	
1758	002200	070742	EFT1	
1759				
1760				
1761			;ERROR 43	"OPI" ERROR DURING PACK ACKNOWLEDGE
1762	002202	065020	EMT43	
1763	002204	070526	EHT1	
1764	002206	070652	EDT1	
1765	002210	070742	EFT1	
1766				
1767				
1768			;ERROR 44	"RMR" ERROR DURING PACK ACKNOWLEDGE
1769	002212	065030	EMT44	
1770	002214	070526	EHT1	
1771	002216	070652	EDT1	
1772	002220	070742	EFT1	
1773				
1774				
1775			;ERROR 45	"ILR" ERROR DURING PACK ACKNOWLEDGE
1776	002222	065040	EMT45	
1777	002224	070526	EHT1	
1778	002226	070652	EDT1	
1779	002230	070742	EFT1	
1780				
1781				
1782			;ERROR 46	"ILF" ERROR DURING PACK ACKNOWLEDGE
1783	002232	065050	EMT46	
1784	002234	070526	EHT1	
1785	002236	070652	EDT1	
1786	002240	070742	EFT1	
1787				
1788				
1789			;ERROR 47	COMPOSITE ERROR STATUS IS INCORRECT
1790	002242	065060	EMT47	
1791	002244	070526	EHT1	
1792	002246	070652	EDT1	
1793	002250	070742	EFT1	
1794				
1795				
1796			;ERROR 50	PARITY ERROR WRITING REMOTE REGISTERS
1797	002252	065066	EMT50	
1798	002254	070526	EHT1	
1799	002256	070652	EDT1	
1800	002260	070742	EFT1	
1801				
1802				

1803			;ERROR 51	INCORRECT IAE STATUS DURING SEEK COMMAND
1804	002262	065076	EMT51	
1805	002264	070526	EHT1	
1806	002266	070652	EDT1	
1807	002270	070742	EFT1	
1808				
1809				
1810			;ERROR 52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE
1811	002272	065110	EMT52	
1812	002274	070526	EHT1	
1813	002276	070652	EDT1	
1814	002300	070742	EFT1	
1815				
1816				
1817			;ERROR 53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE, ASSUME
1818			;	ON CYLINDER LATCH DIDN'T RESET
1819	002302	065126	EMT53	
1820	002304	070526	EHT1	
1821	002306	070652	EDT1	
1822	002310	070742	EFT1	
1823				
1824				
1825			;ERROR 54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND
1826	002312	065144	EMT54	
1827	002314	070526	EHT1	
1828	002316	070652	EDT1	
1829	002320	070742	EFT1	
1830				
1831				
1832			;ERROR 55	DEVICE CHECK DURING SEEK COMMAND
1833	002322	065154	EMT55	
1834	002324	070526	EHT1	
1835	002326	070652	EDT1	
1836	002330	070742	EFT1	
1837				
1838				
1839			;ERROR 56	PIP IS STILL SET AFTER SEEK - SKI IS RESET
1840	002332	065166	EMT56	
1841	002334	070526	EHT1	
1842	002336	070652	EDT1	
1843	002340	070742	EFT1	
1844				
1845			;ERROR 57	ATA DID NOT SET DURING SEEK COMMAND
1846			EMT57	
1847	002342	065204	EHT1	
1848	002344	070526	EDT1	
1849	002346	070652	EFT1	
1850	002350	070742		
1851				
1852				
1853			;ERROR 60	IVC ERROR DURING SEEK COMMAND - LOST
1854			;	VOLUME VALID
1855	002352	065214	EMT60	
1856	002354	070526	EHT1	
1857	002356	070652	EDT1	
1858	002360	070742	EFT1	

1859				
1860				
1861			;ERROR 61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
1862			;	VOLUME VALID IS STIL SET
1863	002362	065232		EMT61
1864	002364	070526		EHT1
1865	002366	070652		EDT1
1866	002370	070742		EFT1
1867				
1868				
1869			;ERROR 62	MOL IS ZERO, BUT OPI WAS NOT
1870			;	REPORTED DURING SEEK COMMAND
1871	002372	065252		EMT62
1872	002374	070526		EHT1
1873	002376	070652		EDT1
1874	002400	070742		EFT1
1875				
1876				
1877			;ERROR 63	UNUSED
1878	002402	000000		0
1879	002404	000000		0
1880	002406	000000		0
1881	002410	000000		0
1882				
1883				
1884			;ERROR 64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
1885	002412	065270		EMT64
1886	002414	070526		EHT1
1887	002416	070652		EDT1
1888	002420	070742		EFT1
1889				
1890				
1891			;ERROR 65	DRIVE EXECUTED A SEEK WITH ERROR SET
1892	002422	065310		EMT65
1893	002424	070526		EHT1
1894	002426	070652		EDT1
1895	002430	070742		EFT1
1896				
1897				
1898			;ERROR 66	UNEXPECTED ERROR SET IN RMR1
1899	002432	065330		EMT66
1900	002434	070526		EHT1
1901	002436	070652		EDT1
1902	002440	070742		EFT1
1903				
1904				
1905			;ERROR 67	UNEXPECTED ERROR SET IN RMR2
1906	002442	065342		EMT67
1907	002444	070526		EHT1
1908	002446	070652		EDT1
1909	002450	070742		EFT1
1910				
1911				
1912			;ERROR 70	ERRONEOUS "IAE" ERROR DURING RECALIBRATE
1913	002452	065354		EMT70
1914	002454	070526		EHT1

1915	002456	070652		EDT1	
1916	002460	070742		EFT1	
1917					
1918					
1919			;ERROR	71	"ILF" ERROR DURING RECALIBRATE
1920	002462	065364		EMT71	
1921	002464	070526		EHT1	
1922	002466	070652		EDT1	
1923	002470	070742		EFT1	
1924					
1925					
1926			;ERROR	72	"OPI" ERROR DURING RECALIBRATE DUE TO "MOL" = 0
1927	002472	065374		EMT72	
1928	002474	070526		EHT1	
1929	002476	070652		EDT1	
1930	002500	070742		EFT1	
1931					
1932					
1933			;ERROR	73	"OPI" ERROR DURING RECALIBRATE BECAUSE ON
1934			;		CYLINDER DIDNT DROP
1935	002502	065412		EMT73	
1936	002504	070526		EHT1	
1937	002506	070652		EDT1	
1938	002510	070742		EFT1	
1939					
1940					
1941			;ERROR	74	"IVC" ERROR DURING RECALIBRATE - "VV" = 0
1942	002512	065430		EMT74	
1943	002514	070526		EHT1	
1944	002516	070652		EDT1	
1945	002520	070742		EFT1	
1946					
1947					
1948			;ERROR	75	ERRONEOUS "IVC" ERROR DURING RECALIBRATE - "VV" = 1
1949	002522	065440		EMT75	
1950	002524	070526		EHT1	
1951	002526	070652		EDT1	
1952	002530	070742		EFT1	
1953					
1954					
1955			;ERROR	76	"SKI" ERROR DURING RECALIBRATE
1956	002532	065460		EMT76	
1957	002534	070526		EHT1	
1958	002536	070652		EDT1	
1959	002540	070742		EFT1	
1960					
1961					
1962			;ERROR	77	"DVC" OCCURRED DURING RECALIBRATE
1963	002542	065470		EMT77	
1964	002544	070526		EHT1	
1965	002546	070652		EDT1	
1966	002550	070742		EFT1	
1967					
1968					
1969			;ERROR	100	LOST "MOL" DURING RECALIBRATE - "OPI" = 0
1970	002552	065502		EMT100	

1971	002554	070526	EHT1	
1972	002556	070652	EDT1	
1973	002560	070742	EFT1	
1974				
1975				
1976				;ERROR 101 LOST "VV" DURING RECALIBRATE - "IVC" = 0
1977	002562	065520	EMT101	
1978	002564	070526	EHT1	
1979	002566	070652	EDT1	
1980	002570	070742	EFT1	
1981				
1982				
1983				;ERROR 102 "ATA" DID NOT SET DURING RECALIBRATE
1984	002572	065536	EMT102	
1985	002574	070526	EHT1	
1986	002576	070652	EDT1	
1987	002600	070742	EFT1	
1988				
1989				
1990				;ERROR 103 "OM" DID NOT RESET DURING RECALIBRATE
1991	002602	065546	EMT103	
1992	002604	070526	EHT1	
1993	002606	070652	EDT1	
1994	002610	070742	EFT1	
1995				
1996				
1997				;ERROR 104 "PIP" IS STIL SET AFTER RECALIBRATE
1998	002612	065560	EMT104	
1999	002614	070526	EHT1	
2000	002616	070652	EDT1	
2001	002620	070742	EFT1	
2002				
2003				
2004				;ERROR 105 UNEXPECTED "ILR" ERROR DURING RECALIBRATE
2005	002622	065576	EMT105	
2006	002624	070526	EHT1	
2007	002626	070652	EDT1	
2008	002630	070742	EFT1	
2009				
2010				
2011				;ERROR 106 UNEXPECTED "RMR" ERROR DURING RECALIBRATE
2012	002632	065606	EMT106	
2013	002634	070526	EHT1	
2014	002636	070652	EDT1	
2015	002640	070742	EFT1	
2016				
2017				
2018				;ERROR 107 "UNS" ERROR DURING RECALIBRATE - AC POWER IS LOW
2019	002642	065616	EMT107	
2020	002644	070526	EHT1	
2021	002646	070652	EDT1	
2022	002650	070742	EFT1	
2023				
2024				
2025				;ERROR 110 CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
2026	002652	065636	EMT110	

2027	002654	070550		EHT110	
2028	002656	070670		EDT110	
2029	002660	070760		EFT110	
2030					
2031					
2032			;ERROR	111	NONEXISTENT DEVICE
2033	002662	065650		EMT111	
2034	002664	070554		EHT111	
2035	002666	070672		EDT111	
2036	002670	070762		EFT111	
2037					
2038					
2039			;ERROR	112	DEVICE NOT AVAILABLE
2040	002672	065656		EMT112	
2041	002674	070554		EHT111	
2042	002676	070672		EDT111	
2043	002700	070762		EFT111	
2044					
2045					
2046			;ERROR	113	BUS TIMEOUT-NED STATUS FAILURE
2047	002702	065664		EMT113	
2048	002704	000000		0	
2049	002706	000000		0	
2050	002710	000000		0	
2051					
2052					
2053			;ERROR	114	DEVICE NOT AN RM03
2054	002712	065700		EMT114	
2055	002714	070560		EHT114	
2056	002716	070674		EDT114	
2057	002720	070764		EFT114	
2058					
2059					
2060			;ERROR	115	RMCS1 NOT INITIALIZED BY UNIBUS
2061	002722	065706		EMT115	
2062	002724	070526		EHT1	
2063	002726	070652		EDT1	
2064	002730	070742		EFT1	
2065					
2066					
2067			;ERROR	116	RMBA NOT INITIALIZED BY UNIBUS
2068	002732	065716		EMT116	
2069	002734	070526		EHT1	
2070	002736	070652		EDT1	
2071	002740	070742		EFT1	
2072					
2073					
2074			;ERROR	117	RMCS2 NOT INITIALIZED BY UNIBUS
2075	002742	065726		EMT117	
2076	002744	070526		EHT1	
2077	002746	070652		EDT1	
2078	002750	070742		EFT1	
2079					
2080					
2081			;ERROR	120	RMER1 NOT INITIALIZED BY UNIBUS
2082	002752	065736		EMT120	

K04

DZRM DA - RMO3 FUNCTIONAL TEST, PART 2
DZRM DA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 49
ERROR POINTER TABLE

SEQ 0051

2083	002754	070526	EHT1	
2084	002756	070652	EDT1	
2085	002760	070742	EFT1	
2086				
2087				
2088				
2089	062762	065746	;ERROR 121	RMAS NOT INITIALIZED BY UNIBUS
2090	002764	070526	EMT121	
2091	002766	070652	EHT1	
2092	002770	070742	EDT1	
2093			EFT1	
2094				
2095				
2096	002772	065756	;ERROR 122	RMMR1 NOT INITIALIZED BY UNIBUS
2097	002774	070526	EMT122	
2098	002776	070652	EHT1	
2099	003000	070742	EDT1	
2100			EFT1	
2101				
2102				
2103	003002	065766	;ERROR 123	RMDS NOT INITIALIZED BY UNIBUS
2104	003004	070526	EMT123	
2105	003006	070652	EHT1	
2106	003010	070742	EDT1	
2107			EFT1	
2108				
2109				
2110	003012	065776	;ERROR 124	RMEC2 NOT INITIALIZED BY UNIBUS
2111	003014	070526	EMT124	
2112	003016	070652	EHT1	
2113	003020	070742	EDT1	
2114			EFT1	
2115				
2116				
2117	003022	066006	;ERROR 125	RMMR2 NOT INITIALIZED BY UNIBUS
2118	003024	070526	EMT125	
2119	003026	070652	EHT1	
2120	003030	070742	EDT1	
2121			EFT1	
2122				
2123				
2124	003032	066016	;ERROR 126	RMCS1 NOT CLEARED BY CONTROLLER CLEAR
2125	003034	070526	EMT126	
2126	003036	070652	EHT1	
2127	003040	070742	EDT1	
2128			EFT1	
2129				
2130				
2131	003042	066030	;ERROR 127	RMBA NOT CLEARED BY CONTROLLER CLEAR
2132	003044	070526	EMT127	
2133	003046	070652	EHT1	
2134	003050	070742	EDT1	
2135			EFT1	
2136				
2137				
2138	003052	066042	;ERROR 130	RMCS2 NOT CLEARED BY CONTROLLER CLEAR
			EMT130	

2139	003054	070526		EHT1	
2140	003056	070652		EDT1	
2141	003060	070742		EFT1	
2142					
2143					
2144			;ERROR	131	RMER1 NOT CLEARED BY CONTROLLER CLEAR
2145	003062	066054		EMT131	
2146	003064	070526		EHT1	
2147	003066	070652		EDT1	
2148	003070	070742		EFT1	
2149					
2150					
2151			;ERROR	132	RMAS NOT CLEARED BY CONTROLLER CLEAR
2152	003072	066066		EMT132	
2153	003074	070526		EHT1	
2154	003076	070652		EDT1	
2155	003100	070742		EFT1	
2156					
2157					
2158			;ERROR	133	RMMR1 NOT CLEARED BY CONTROLLER CLEAR
2159	003102	066100		EMT133	
2160	003104	070526		EHT1	
2161	003106	070652		EDT1	
2162	003110	070742		EFT1	
2163					
2164					
2165			;ERROR	134	RMDS NOT CLEARED BY CONTROLLER CLEAR
2166	003112	066112		EMT134	
2167	003114	070526		EHT1	
2168	003116	070652		EDT1	
2169	003120	070742		EFT1	
2170					
2171					
2172			;ERROR	135	RMEC2 NOT CLEARED BY CONTROLLER CLEAR
2173	003122	066124		EMT135	
2174	003124	070526		EHT1	
2175	003126	070652		EDT1	
2176	003130	070742		EFT1	
2177					
2178					
2179			;ERROR	136	RMMR2 NOT CLEARED BY CONTROLLER CLEAR
2180	003132	066136		EMT136	
2181	003134	070526		EHT1	
2182	003136	070652		EDT1	
2183	003140	070742		EFT1	
2184					
2185					
2186			;ERROR	137	RMCS1 NOT CLEARED BY ERROR CLEAR
2187	003142	066150		EMT137	
2188	003144	070526		EHT1	
2189	003146	070652		EDT1	
2190	003150	070742		EFT1	
2191					
2192					
2193			;ERROR	140	RMCS2 NOT CLEARED BY ERROR CLEAR
2194	003152	066160		EMT140	

2195	003154	070526	EHT1	
2196	003156	070652	EDT1	
2197	003160	070742	EFT1	
2198				
2199				
2200			;ERROR	141 RMCS1 NOT CLEARED BY DRIVE CLEAR
2201	003162	066170	EMT141	
2202	003164	070526	EHT1	
2203	003166	070652	EDT1	
2204	003170	070742	EFT1	
2205				
2206				
2207			;ERROR	142 RMD5 NOT CLEARED BY DRIVE CLEAR
2208	003172	066200	EMT142	
2209	003174	070526	EHT1	
2210	003176	070652	EDT1	
2211	003200	070742	EFT1	
2212				
2213				
2214			;ERROR	143 RMER1 NOT CLEARED BY DRIVE CLEAR
2215	003202	066210	EMT143	
2216	003204	070526	EHT1	
2217	003206	070652	EDT1	
2218	003210	070742	EFT1	
2219				
2220				
2221			;ERROR	144 RMAS NOT CLEARED BY DRIVE CLEAR
2222	003212	066220	EMT144	
2223	003214	070526	EHT1	
2224	003216	070652	EDT1	
2225	003220	070742	EFT1	
2226				
2227				
2228			;ERROR	145 RMMR1 NOT CLEARED BY DRIVE CLEAR
2229	003222	066230	EMT145	
2230	003224	070526	EHT1	
2231	003226	070652	EDT1	
2232	003230	070742	EFT1	
2233				
2234				
2235			;ERROR	146 RMMR2 NOT CLEARED BY DRIVE CLEAR
2236	003232	066240	EMT146	
2237	003234	070526	EHT1	
2238	003236	070652	EDT1	
2239	003240	070742	EFT1	
2240				
2241				
2242			;ERROR	147 RMER2 NOT CLEARED BY DRIVE CLEAR
2243	003242	066250	EMT147	
2244	003244	070526	EHT1	
2245	003246	070652	EDT1	
2246	003250	070742	EFT1	
2247				
2248				
2249			;ERROR	150 RMEC2 NOT CLEARED BY DRIVE CLEAR
2250	003252	066260	EMT150	

2251	003254	070526	EHT1	
2252	003256	070652	EDT1	
2253	003260	070742	EFT1	
2254				
2255				
2256				;ERROR 151 MEDIUM NOT ON LINE
2257	003262	066270	EMT151	
2258	003264	070526	EHT1	
2259	003266	070652	EDT1	
2260	003270	070742	EFT1	
2261				
2262				
2263				;ERROR 152 DRIVE FAULT
2264	003272	066302	EMT152	
2265	003274	070526	EHT1	
2266	003276	070652	EDT1	
2267	003300	070742	EFT1	
2268				
2269				
2270				;ERROR 153 UNSAFE SHOULD BE SET BECAUSE DVC IS SET
2271	003302	066314	EMT153	
2272	003304	070526	EHT1	
2273	003306	070652	EDT1	
2274	003310	070742	EFT1	
2275				
2276				
2277				;ERROR 154 UNSAFE SHOULD NOT BE SET, AC IS LOW
2278	003312	066332	EMT154	
2279	003314	070526	EHT1	
2280	003316	070652	EDT1	
2281	003320	070742	EFT1	
2282				
2283				
2284				;ERROR 155 VOLUME VALID NOT SET BY PACK ACK
2285	003322	066350	EMT155	
2286	003324	070526	EHT1	
2287	003326	070652	EDT1	
2288	003330	070742	EFT1	
2289				
2290				
2291				;ERROR 156 OFFSET MODE NOT SET BY OFFSET COMMAND
2292	003332	066362	EMT156	
2293	003334	070526	EHT1	
2294	003336	070652	EDT1	
2295	003340	070742	EFT1	
2296				
2297				
2298				;ERROR 157 OFFSET MODE NOT RESET BY RTC COMMAND
2299	003342	066374	EMT157	
2300	003344	070526	EHT1	
2301	003346	070652	EDT1	
2302	003350	070742	EFT1	
2303				
2304				
2305				;ERROR 160 RMOF NOT RESET BY RIP COMMAND
2306	003352	066406	EMT160	

2307 003354 070526
 2308 003356 070652
 2309 003360 070742
 2310
 2311
 2312
 2313 003362 066416
 2314 003364 070526
 2315 003366 070652
 2316 003370 070742
 2317
 2318
 2319
 2320 003372 066430
 2321 003374 070526
 2322 003376 070652
 2323 003400 070742
 2324
 2325
 2326
 2327
 2328 003402 070322
 2329 003404 070610
 2330 003406 070706
 2331 003410 070776
 2332
 2333
 2334
 2335 003412 066452
 2336 003414 070526
 2337 003416 070652
 2338 003420 070742
 2339
 2340
 2341
 2342 003422 066460
 2343 003424 070526
 2344 003426 070652
 2345 003430 070742
 2346
 2347
 2348
 2349 003432 066466
 2350 003434 070526
 2351 003436 070652
 2352 003440 070742
 2353
 2354
 2355
 2356 003442 066474
 2357 003444 070526
 2358 003446 070652
 2359 003450 070742
 2360
 2361
 2362

EHT1
 EDT1
 EFT1

 ;ERROR 161 RMDA NOT RESET BY RIP COMMAND
 EMT161
 EHT1
 EDT1
 EFT1

 ;ERROR 162 RMDC NOT RESET BY RIP COMMAND
 EMT162
 EHT1
 EDT1
 EFT1

 ;ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
 ; WRITE BUFFER
 EMT336
 EHT336
 EDT336
 EFT336

 ;ERROR 164 OPI SHOULD NOT BE SET
 EMT164
 EHT1
 EDT1
 EFT1

 ;ERROR 165 IVC SHOULD NOT BE SET
 EMT165
 EHT1
 EDT1
 EFT1

 ;ERROR 166 IAE SHOULD NOT BE SET
 EMT166
 EHT1
 EDT1
 EFT1

 ;ERROR 167 NEM SHOULD NOT BE SET
 EMT167
 EHT1
 EDT1
 EFT1

 ;ERROR 170 UNUSED

2363	003452	000000		0	
2364	003454	000000		0	
2365	003456	000000		0	
2366	003460	000000		0	
2367					
2368					
2369			;ERROR	171	"ATA" NOT SET DURING RETURN TO CENTERLINE
2370	003462	066512		EMT171	
2371	003464	070526		EHT1	
2372	003466	070652		EDT1	
2373	003470	070742		EFT1	
2374					
2375					
2376			;ERROR	172	"ATA" NOT SET BY OFFSET COMMAND
2377	003472	066522		EMT172	
2378	003474	070526		EHT1	
2379	003476	070652		EDT1	
2380	003500	070742		EFT1	
2381					
2382					
2383			;ERROR	173	RMR2 NOT INITIALIZED BY UNIBUS INIT
2384	003502	066532		EMT173	
2385	003504	070526		EHT1	
2386	003506	070652		EDT1	
2387	003510	070742		EFT1	
2388					
2389					
2390			;ERROR	174	RMR2 NOT INITIALIZED BY CONTROLLER CLEAR
2391	003512	066542		EMT174	
2392	003514	070526		EHT1	
2393	003516	070652		EDT1	
2394	003520	070742		EFT1	
2395					
2396					
2397			;ERROR	175	SELECTED DEVICE IS IN WRITE PROTECT
2398	003522	066554		EMT175	
2399	003524	070526		EHT1	
2400	003526	070652		EDT1	
2401	003530	070742		EFT1	
2402					
2403					
2404			;ERROR	176	CANNOT SET DIAGNOSTIC MODE
2405	003532	066562		EMT176	
2406	003534	070526		EHT1	
2407	003536	070652		EDT1	
2408	003540	070742		EFT1	
2409					
2410					
2411			;ERROR	177	INCORRECT "MOL" STATUS DURING DIAGNOSTIC MODE
2412	003542	066570		EMT177	
2413	003544	070526		EHT1	
2414	003546	070652		EDT1	
2415	003550	070742		EFT1	
2416					
2417					
2418			;ERROR	200	INCORRECT "PIP" STATUS DURING DIAGNOSTIC MODE

2419	003552	066602	EMT200	
2420	003554	070526	EHT1	
2421	003556	070652	EDT1	
2422	003560	070742	EFT1	
2423				
2424				
2425				
2426	003562	066614	EMT201	;ERROR 201 INCORRECT "WRL" STATUS DURING DIAGNOSTIC MODE
2427	003564	070526	EHT1	
2428	003566	070652	EDT1	
2429	003570	070742	EFT1	
2430				
2431				
2432				
2433	003572	066626	EMT202	;ERROR 202 INCORRECT "SKI" STATUS DURING DIAGNOSTIC MODE
2434	003574	070526	EHT1	
2435	003576	070652	EDT1	
2436	003600	070742	EFT1	
2437				
2438				
2439				
2440	003602	066640	EMT203	;ERROR 203 INCORRECT "DVC" STATUS DURING DIAGNOSTIC MODE
2441	003604	070526	EHT1	
2442	003606	070652	EDT1	
2443	003610	070742	EFT1	
2444				
2445				
2446				
2447	003612	066652	EMT204	;ERROR 204 "VV" WAS NOT RESET BY MAINTENANCE UNIT READY
2448	003614	070526	EHT1	
2449	003616	070652	EDT1	
2450	003620	070742	EFT1	
2451				
2452				
2453				
2454	003622	066670	EMT205	;ERROR 205 SELECTED DEVICE HAS A PERSISTENT "SKI" ERROR
2455	003624	070526	EHT1	
2456	003626	070652	EDT1	
2457	003630	070742	EFT1	
2458				
2459				
2460				
2461	003632	066700	EMT206	;ERROR 206 "LBC" DID NOT SET DURING DIAGNOSTIC MODE
2462	003634	070526	EHT1	
2463	003636	070652	EDT1	
2464	003640	070742	EFT1	
2465				
2466				
2467				
2468	003642	066710	EMT207	;ERROR 207 UNEXPECTED LOSS OF "MOL" - MEDIUM IS OFF LINE
2469	003644	070526	EHT1	
2470	003646	070652	EDT1	
2471	003650	070742	EFT1	
2472				
2473				
2474				
				;ERROR 210 UNEXPECTED LOSS OF VOLUME VALID - "VV" = 0

2475	003652	066722		EMT210	
2476	003654	070526		EHT1	
2477	003656	070652		EDT1	
2478	003660	070742		EFT1	
2479					
2480					
2481			;ERROR	211	UNEXPECTED MECHANICAL MOTION - "PIP" = 1
2482	003662	066730		EMT211	
2483	003664	070526		EHT1	
2484	003666	070652		EDT1	
2485	003670	070742		EFT1	
2486					
2487					
2488			;ERROR	212	UNEXPECTED DEVICE FAULT - "DVC" = 1
2489	003672	066744		EMT212	
2490	003674	070526		EHT1	
2491	003676	070652		EDT1	
2492	003700	070742		EFT1	
2493					
2494					
2495			;ERROR	213	UNEXPECTED SEEK INCOMPLETE ERROR - "SKI" = 1
2496	003702	066760		EMT213	
2497	003704	070526		EHT1	
2498	003706	070652		EDT1	
2499	003710	070742		EFT1	
2500					
2501					
2502			;ERROR	214	DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
2503	003712	066770		EMT214	
2504	003714	070540		EHT2	
2505	003716	070662		EDT2	
2506	003720	070752		EFT2	
2507					
2508					
2509			;ERROR	215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
2510	003722	067010		EMT215	
2511	003724	070540		EHT2	
2512	003726	070662		EDT2	
2513	003730	070752		EFT2	
2514					
2515					
2516			;ERROR	216	INCORRECT "IVC" STATUS
2517	003732	067022		EMT216	
2518	003734	070526		EHT1	
2519	003736	070652		EDT1	
2520	003740	070742		EFT1	
2521					
2522					
2523			;ERROR	217	INCORRECT "IAE" STATUS
2524	003742	067032		EMT217	
2525	003744	070526		EHT1	
2526	003746	070652		EDT1	
2527	003750	070742		EFT1	
2528					
2529					
2530			;ERROR	220	INCORRECT "WLE" STATUS

031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086

003752 067042
003754 070526
003756 070652
003760 070742

003762 067052
003764 070526
003766 070652
003770 070742

003772 067062
003774 070526
003776 070652
004000 070742

004002 067072
004004 070564
004006 070676
004010 070766

004012 000000
004014 000000
004016 000000
004020 000000

004022 000000
004024 000000
004026 000000
004030 000000

004032 000000
004034 000000
004036 000000
004040 000000

004042 000000
004044 000000
004046 000000
004050 000000

EMT220
EHT1
EDT1
EFT1

;ERROR 221 INCORRECT "OPI" STATUS
EMT221
EHT1
EDT1
EFT1

;ERROR 222 RMO3 DID NOT DETECT RMR ERROR
EMT222
EHT1
EDT1
EFT1

;ERROR 223 RMO3 DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
EMT223
EHT223
EDT223
EFT223

;ERROR 224 UNUSED
0
0
0
0

;ERROR 225 UNUSED
0
0
0
0

;ERROR 226 UNUSED
0
0
0
0

;ERROR 227 UNUSED
0
0
0
0

;ERROR 230 UNUSED

2587	004052	000000			0
2588	004054	000000			0
2589	004056	000000			0
2590	004060	000000			0
2591					
2592					
2593					
2594					
2595					
2596					
2597					
2598					
2599					
2600					
2601					
2602					
2603					
2604					
2605					
2606					
2607					
2608	004062	000000			
2609	004064	000000			
2610	004066	000000			
2611	004070	000000			
2612					
2613					
2614					
2615					
2616					
2617					
2618					
2619					
2620					
2621					
2622					
2623					
2624					
2625					
2626					
2627					
2628					
2629					
2630					
2631					
2632					
2633					
2634					
2635					
2636					
2637					
2638					
2639					
2640					
2641					
2642					

2643	004152	000000		0	
2644	004154	000000		0	
2645	004156	000000		0	
2646	004160	000000		0	
2647					
2648					
2649			;ERROR	241	UNUSED
2650	004162	000000		0	
2651	004164	000000		0	
2652	004166	000000		0	
2653	004170	000000		0	
2654					
2655					
2656			;ERROR	242	UNUSED
2657	004172	000000		0	
2658	004174	000000		0	
2659	004176	000000		0	
2660	004200	000000		0	
2661					
2662					
2663			;ERROR	243	UNUSED
2664	004202	000000		0	
2665	004204	000000		0	
2666	004206	000000		0	
2667	004210	000000		0	
2668					
2669					
2670			;ERROR	244	UNUSED
2671	004212	000000		0	
2672	004214	000000		0	
2673	004216	000000		0	
2674	004220	000000		0	
2675					
2676					
2677			;ERROR	245	UNUSED
2678	004222	000000		0	
2679	004224	000000		0	
2680	004226	000000		0	
2681	004230	000000		0	
2682					
2683					
2684			;ERROR	246	"ATA" NOT RESET BY GO WHEN "ERR" = 0
2685	004232	067146		EMT246	
2686	004234	070526		EHT1	
2687	004236	070652		EDT1	
2688	004240	070742		EFT1	
2689					
2690					
2691			;ERROR	247	"ATA" NOT RESET BY WRITING RMAS
2692	004242	067156		EMT247	
2693	004244	070526		EHT1	
2694	004246	070652		EDT1	
2695	004250	070742		EFT1	
2696					
2697					
2698			;ERROR	250	"ATA" WAS RESET BY GO WHEN "ERR" = 1

2699	004252	067170		EMT250	
2700	004254	070526		EHT1	
2701	004256	070652		EDT1	
2702	004260	070742		EFT1	
2703					
2704					
2705			;ERROR	251	PROGRAM INTERRUPT WAS NOT GENERATED
2706	004262	067204		EMT251	
2707	004264	070540		EHT2	
2708	004266	070662		EDT2	
2709	004270	070752		EFT2	
2710					
2711					
2712			;ERROR	252	PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
2713	004272	067212		EMT252	
2714	004274	070540		EHT2	
2715	004276	070662		EDT2	
2716	004300	070752		EFT2	
2717					
2718					
2719			;ERROR	253	OFFSET MODE WAS NOT RESET BY WRITING RMDC
2720	004302	067220		EMT253	
2721	004304	070526		EHT1	
2722	004306	070652		EDT1	
2723	004310	070742		EFT1	
2724					
2725					
2726			;ERROR	254	INCORRECT "ILF" STATUS
2727	004312	067236		EMT254	
2728	004314	070526		EHT1	
2729	004316	070652		EDT1	
2730	004320	070742		EFT1	
2731					
2732					
2733			;ERROR	255	INCORRECT "ATA" STATUS
2734	004322	067246		EMT255	
2735	004324	070526		EHT1	
2736	004326	070652		EDT1	
2737	004330	070742		EFT1	
2738					
2739					
2740			;ERROR	256	INCORRECT "ILR" STATUS
2741	004332	067256		EMT256	
2742	004334	070576		EHT256	
2743	004336	070676		EDT223	
2744	004340	070766		EFT223	
2745					
2746					
2747			;ERROR	257	INVALID IAE STATUS DURING SEARCH COMMAND
2748	004342	067266		EMT257	
2749	004344	070526		EHT1	
2750	004346	070652		EDT1	
2751	004350	070742		EFT1	
2752					
2753					
2754			;ERROR	260	"IVC" WAS NOT DETECTED DURING SEARCH COMMAND

2755	004352	067300		EMT260	
2756	004354	070526		EHT1	
2757	004356	070652		EDT1	
2758	004360	070742		EFT1	
2759					
2760					
2761			;ERROR	261	DRIVE EXECUTED SEARCH WITH ERROR SET
2762	004362	067312		EMT261	
2763	004364	070526		EHT1	
2764	004366	070652		EDT1	
2765	004370	070742		EFT1	
2766					
2767					
2768			;ERROR	262	"LBC" ERROR NOT SET DURING DIAGNOSTIC MODE
2769	004372	067332		EMT262	
2770	004374	070526		EHT1	
2771	004376	070652		EDT1	
2772	004400	070742		EFT1	
2773					
2774					
2775			;ERROR	263	"SKI" ERROR DURING SEARCH COMMAND
2776	004402	067342		EMT263	
2777	004404	070526		EHT1	
2778	004406	070652		EDT1	
2779	004410	070742		EFT1	
2780					
2781					
2782			;ERROR	264	"IVC" ERROR DURING SEARCH - LOST VOLUME VALID
2783	004412	067352		EMT264	
2784	004414	070526		EHT1	
2785	004416	070652		EDT1	
2786	004420	070742		EFT1	
2787					
2788					
2789			;ERROR	265	ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
2790	004422	067372		EMT265	
2791	004424	070526		EHT1	
2792	004426	070652		EDT1	
2793	004430	070742		EFT1	
2794					
2795					
2796			;ERROR	266	DEVICE FAULT (DVC) DURING SEARCH
2797	004432	067412		EMT266	
2798	004434	070526		EHT1	
2799	004436	070652		EDT1	
2800	004440	070742		EFT1	
2801					
2802					
2803			;ERROR	267	SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER ADDRESS IS TOO LARGE
2804			:		
2805	004442	067424		EMT267	
2806	004444	070526		EHT1	
2807	004446	070652		EDT1	
2808	004450	070742		EFT1	
2809					
2810					

2811			;ERROR 270	OPI ERROR DURING SEARCH BECAUSE MOL = 0
2812	004452	067442	EMT270	
2813	004454	070526	EHT1	
2814	004456	070652	EDT1	
2815	004460	070742	EFT1	
2816				
2817				
2818			;ERROR 271	OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
2819			;	DIDN'T DROP
2820	004462	067456	EMT271	
2821	004464	070526	EHT1	
2822	004466	070652	EDT1	
2823	004470	070742	EFT1	
2824				
2825				
2826			;ERROR 272	LOST MOL DURING SEARCH, OPI IS NOT SET
2827	004472	067474	EMT272	
2828	004474	070526	EHT1	
2829	004476	070652	EDT1	
2830	004500	070742	EFT1	
2831				
2832				
2833			;ERROR 273	PIP STIL SET AFTER SEARCH
2834	004502	067512	EMT273	
2835	004504	070526	EHT1	
2836	004506	070652	EDT1	
2837	004510	070742	EFT1	
2838				
2839				
2840			;ERROR 274	PARITY ERROR OCCURRED WHILE WRITING REMOTE
2841			;	REGISTERS BUT MXF DID NOT SET
2842	004512	067530	EMT274	
2843	004514	070526	EHT1	
2844	004516	070652	EDT1	
2845	004520	070742	EFT1	
2846				
2847				
2848			;ERROR 275	MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
2849			;	COMMAND STARTED
2850	004522	067546	EMT275	
2851	004524	070526	EHT1	
2852	004526	070652	EDT1	
2853	004530	070742	EFT1	
2854				
2855				
2856			;ERROR 276	"OPI" ERROR DURING DATA TRANSFER BECAUSE "MOL" WAS
2857			;	ZERO
2858	004532	067560	EMT276	
2859	004534	070526	EHT1	
2860	004536	070652	EDT1	
2861	004540	070742	EFT1	
2862				
2863				
2864			;ERROR 277	"OPI" ERROR DURING DATA TRANSFER BECAUSE 1) ON
2865			;	CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
2866			;	3) RUN TIMED OUT

2867	004542	067574		EMT277
2868	004544	070526		EHT1
2869	004546	070652		EDT1
2870	004550	070742		EFT1
2871				
2872				
2873			;ERROR	300 "IVC" ERROR DURING DATA TRANSFER BECAUSE VOLUME
2874			;	WAS NOT VALID
2875	004552	067612		EMT300
2876	004554	070526		EHT1
2877	004556	070652		EDT1
2878	004560	070742		EFT1
2879				
2880				
2881			;ERROR	301 ERRONEOUS "IVC" ERROR DURING DATA TRANSFER - VOLUME
2882			;	IS VALID
2883	004562	067632		EMT301
2884	004564	070526		EHT1
2885	004566	070652		EDT1
2886	004570	070742		EFT1
2887				
2888				
2889			;ERROR	302 "ILR" ERROR DURING DATA TRANSFER
2890	004572	067654		EMT302
2891	004574	070526		EHT1
2892	004576	070652		EDT1
2893	004600	070742		EFT1
2894				
2895				
2896			;ERROR	303 "ILF" ERROR DURING DATA TRANSFER
2897	004602	067666		EMT303
2898	004604	070526		EHT1
2899	004606	070652		EDT1
2900	004610	070742		EFT1
2901				
2902				
2903			;ERROR	304 "RMR" ERROR DURING DATA TRANSFER
2904	004612	067700		EMT304
2905	004614	070526		EHT1
2906	004616	070652		EDT1
2907	004620	070742		EFT1
2908				
2909				
2910			;ERROR	305 INCORRECT "IAE" STATUS DURING DATA TRANSFER
2911	004622	067712		EMT305
2912	004624	070526		EHT1
2913	004626	070652		EDT1
2914	004630	070742		EFT1
2915				
2916				
2917			;ERROR	306 "SKI" ERROR DURING DATA TRANSFER
2918	004632	067724		EMT306
2919	004634	070526		EHT1
2920	004636	070652		EDT1
2921	004640	070742		EFT1
2922				

2923				
2924				
2925	004642	067734	;ERROR 307	DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
2926	004644	070526	EMT307	
2927	004646	070652	EHT1	
2928	004650	070742	EDT1	
2929			EFT1	
2930				
2931			;ERROR 310	DEVICE FAULT DURING DATA TRANSFER
2932	004652	067754	EMT310	
2933	004654	070526	EHT1	
2934	004656	070652	EDT1	
2935	004660	070742	EFT1	
2936				
2937				
2938			;ERROR 311	LOSS OF BIT CLOCK DURING DATA TRANSFER
2939	004662	067766	EMT311	
2940	004664	070526	EHT1	
2941	004666	070652	EDT1	
2942	004670	070742	EFT1	
2943				
2944				
2945			;ERROR 312	LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
2946	004672	070000	EMT312	
2947	004674	070526	EHT1	
2948	004676	070652	EDT1	
2949	004700	070742	EFT1	
2950				
2951				
2952			;ERROR 313	UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
2953	004702	070012	EMT313	
2954	004704	070526	EHT1	
2955	004706	070652	EDT1	
2956	004710	070742	EFT1	
2957				
2958				
2959			;ERROR 314	DRIVE TIMING ERROR DURING DATA TRANSFER
2960	004712	070032	EMT314	
2961	004714	070526	EHT1	
2962	004716	070652	EDT1	
2963	004720	070742	EFT1	
2964				
2965				
2966			;ERROR 315	WRITE LOCK ERROR
2967	004722	070044	EMT315	
2968	004724	070526	EHT1	
2969	004726	070652	EDT1	
2970	004730	070742	EFT1	
2971				
2972				
2973			;ERROR 316	ERRONEOUS WRITE LOCK ERROR
2974	004732	070056	EMT316	
2975	004734	070526	EHT1	
2976	004736	070652	EDT1	
2977	004740	070742	EFT1	
2978				

2979				
2980			;ERROR	317 HEADER CRC ERROR DURING DATA TRANSFER
2981	004742	070070		EMT317
2982	004744	070526		EHT1
2983	004746	070652		EDT1
2984	004750	070742		EFT1
2985				
2986				
2987			;ERROR	320 FORMAT ERROR DURING DATA TRANSFER
2988	004752	070100		EMT320
2989	004754	070526		EHT1
2990	004756	070652		EDT1
2991	004760	070742		EFT1
2992				
2993				
2994			;ERROR	321 HEADER COMPARE ERROR DURING DATA TRANSFER
2995	004762	070110		EMT321
2996	004764	070526		EHT1
2997	004766	070652		EDT1
2998	004770	070742		EFT1
2999				
3000				
3001			;ERROR	322 HEADER ERRORS SHOULD NOT BE SET
3002	004772	070120		EMT322
3003	004774	070526		EHT1
3004	004776	070652		EDT1
3005	005000	070742		EFT1
3006				
3007				
3008			;ERROR	323 DATA CHECK ERROR DURING DATA TRANSFER
3009	005002	070126		EMT323
3010	005004	070526		EHT1
3011	005006	070652		EDT1
3012	005010	070742		EFT1
3013				
3014				
3015			;ERROR	324 CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
3016	005012	070136		EMT324
3017	005014	070526		EHT1
3018	005016	070652		EDT1
3019	005020	070742		EFT1
3020				
3021				
3022			;ERROR	325 UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
3023	005022	070150		EMT325
3024	005024	070526		EHT1
3025	005026	070652		EDT1
3026	005030	070742		EFT1
3027				
3028				
3029			;ERROR	326 DATA PAPITY ERROR DURING READ COMMAND
3030	005032	070162		EMT326
3031	005034	070526		EHT1
3032	005036	070652		EDT1
3033	005040	070742		EFT1
3034				

3035				
3036			;ERROR	327 OFFSET MODE NOT RESET BY WRITE COMMAND
3037	005042	070200		EMT327
3038	005044	070526		EHT1
3039	005046	070652		EDT1
3040	005050	070742		EFT1
3041				
3042				
3043			;ERROR	330 DATA PARITY ERROR DURING WRITE COMMAND
3044	005052	070212		EMT330
3045	005054	070526		EHT1
3046	005056	070652		EDT1
3047	005060	070742		EFT1
3048				
3049				
3050			;ERROR	331 WRITE CLOCK FAILURE DURING WRITE COMMAND
3051	005062	070222		EMT331
3052	005064	070526		EHT1
3053	005066	070652		EDT1
3054	005070	070742		EFT1
3055				
3056				
3057			;ERROR	332 DATA LATE ERROR DURING DATA TRANSFER
3058	005072	070234		EMT332
3059	005074	070526		EHT1
3060	005076	070652		EDT1
3061	005100	070742		EFT1
3062				
3063				
3064			;ERROR	333 PIP STIL SET AFTER DATA TRANSFER - SKI = 0
3065	005102	070246		EMT333
3066	005104	070526		EHT1
3067	005106	070652		EDT1
3068	005110	070742		EFT1
3069				
3070				
3071			;ERROR	334 LOST MOL DURING DATA TRANSFER - OPI = 0
3072	005112	070264		EMT334
3073	005114	070526		EHT1
3074	005116	070652		EDT1
3075	005120	070742		EFT1
3076				
3077				
3078			;ERROR	335 LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
3079	005122	070302		EMT335
3080	005124	070526		EHT1
3081	005126	070652		EDT1
3082	005130	070742		EFT1
3083				
3084				
3085			;ERROR	336 DATA READ DOES NOT COMPARE WITH DATA WRITTEN
3086	005132	070322		EMT336
3087	005134	070610		EHT336
3088	005136	070706		EDT336
3089	005140	070776		EFT336
3090				

3091				
3092			;ERROR	337 WRITE CHECK ERROR NOT DETECTED
3093	005142	070332		EMT337
3094	005144	070622		EHT337
3095	005146	070716		EDT337
3096	005150	071006		EFT337
3097				
3098				
3099			;ERROR	340 WRITE CHECK ERROR AT UNEXPECTED ADDRESS
3100	005152	070342		EMT340
3101	005154	070610		EHT336
3102	005156	070706		EDT336
3103	005160	070776		EFT336
3104				
3105				
3106			;ERROR	341 INCORRECT DATA DURING WRITE CHECK ERROR
3107	005162	070354		EMT341
3108	005164	070610		EHT336
3109	005166	070706		EDT336
3110	005170	070776		EFT336
3111				
3112				
3113			;ERROR	342 "IVC" ERROR NOT DETECTED DURING DATA TRANSFER
3114	005172	070362		EMT342
3115	005174	070526		EHT1
3116	005176	070652		EDT1
3117	005200	070742		EFT1
3118				
3119				
3120			;ERROR	343 "FER" NOT DETECTED DURING DATA TRANSFER
3121	005202	070374		EMT343
3122	005204	070526		EHT1
3123	005206	070652		EDT1
3124	005210	070742		EFT1
3125				
3126				
3127			;ERROR	344 "HCE" NOT DETECTED DURING DATA TRANSFER
3128	005212	070406		EMT344
3129	005214	070634		EHT344
3130	005216	070726		EDT344
3131	005220	071016		EFT344
3132				
3133				
3134			;ERROR	345 "BSE" NOT DETECTED DURING DATA TRANSFER
3135	005222	070420		EMT345
3136	005224	070526		EHT1
3137	005226	070652		EDT1
3138	005230	070742		EFT1
3139				
3140				
3141			;ERROR	346 HEADER ERROR WAS DETECTED W/ HCI SET
3142	005232	070430		EMT346
3143	005234	070526		EHT1
3144	005236	070652		EDT1
3145	005240	070742		EFT1
3146				

```

3147
3148 ;ERROR 347 DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
3149 005242 070444 EMT347
3150 005244 070526 EHT1
3151 005246 070652 EDT1
3152 005250 070742 EFT1
3153
3154
3155 ;ERROR 350 LOST VOLUME VALID DURING SEARCH - "IVC" = 0
3156 005252 070456 EMT350
3157 005254 070526 EHT1
3158 005256 070652 EDT1
3159 005260 070742 EFT1
3160
3161
3162 ;ERROR 351 "ATA" DID NOT SET DURING SEARCH
3163 005262 070474 EMT351
3164 005264 070526 EHT1
3165 005266 070652 EDT1
3166 005270 070742 EFT1
3167
3168
3169 ;ERROR 352 PROGRAM TIMEOUT WHILE TESTING RMLA
3170 005272 070504 EMT352
3171 005274 000000 0
3172 005276 000000 0
3173 005300 000000 0
3174
3175
3176 ;ERROR 353 LOOK AHEAD TEST FAILS
3177 005302 070510 EMT353
3178 005304 070646 EHT353
3179 005306 070740 EDT353
3180 005310 071026 EFT353
3181
3182
3183 ;ERROR 354 BSE SHOULD NOT BE SET
3184 005312 070520 EMT354
3185 005314 070526 EHT1
3186 005316 070652 EDT1
3187 005320 070742 EFT1
3188
3189
3190 ;PUT ERROR TABLE HERE
3191 ;SBTTL ERROR TABLE USAGE
3192
3193 ;THE ERROR TABLE ABOVE CONSISTS OF FOUR WORD ENTRIES FOR EACH ERROR
3194 ;NUMBER, I.E.,
3195 ;
3196 ; EMT - ERROR MESSAGE TABLE ADDRESS
3197 ; EHT - ERROR HEADER TABLE ADDRESS
3198 ; EDT - ERROR DATA TABLE ADDRESS
3199 ; EFT - ERROR FORMAT TABLE ADDRESS
3200
3201 ;THE EMT ENTRY IS THE ADDRESS OF THE TABLE OF ERROR MESSAGE STRINGS
3202 ;FOR THE PARTICULAR ERROR. EACH ERROR MESSAGE TABLE LISTS THE ADDRESS

```

3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221

: OF ONE OR MORE ERROR MESSAGE STRINGS WHICH ARE TO BE FORMATTED AND
: TYPED BY THE ERROR TYPE SUBROUTINE. IF THE EMT ENTRY IS ZERO, THERE IS
: NO MESSAGE TO BE TYPED FOR THE ERROR.

: SIMILARLY, THE EHT, EDT, AND EFT ENTRIES ARE ADDRESSES OF TABLES
: OF HEADER, DATA AND FORMAT INFORMATION FOR A GIVEN ERROR. EACH ENTRY
: IN THE ERROR HEADER TABLE MAY OR MAY NOT HAVE AN ASSOCIATED LINE OF
: DATA, HOWEVER, EACH DATA LINE MUST HAVE AN ASSOCIATED FORMAT AND
: HEADER. THAT IS, A HEADER LINE MAY BE PRINTED WITHOUT ANY DATA,
: BUT A DATA LINE IS NOT PRINTED WITHOUT A HEADER, AND EACH DATA LINE
: MUST ALSO HAVE A FORMAT.

: IN SUMMARY,

: EACH NONZERO ENTRY IS THE ADDRESS OF A TABLE,
: EACH TABLE IS A LIST OF ADDRESSES WHICH DEFINES THE LOCATIONS
: OF MESSAGE STRINGS, HEADERS, DATA OR FORMAT.

```

3222 .SBTTL START OF PROGRAM
3223
3224
3225 005322 START:
3226
3227 ;CLEAR AND SETUP FOR TEST EXECUTION
3228 NOP
3229 005324 000005 RESET ;INITIALIZE THE SYSTEM
3230 005326 013746 000300 MOV PR6, -(SP) ;PUT NEW PS ON STACK
3231 005332 012746 005340 MOV #64$, -(SP) ;PUT NEW PC ON STACK
3232 005336 000002 RTI ;POP NEW PC AND PS
3233 005340
3234
3235 .SBTTL INITIALIZE THE COMMON TAGS
3236 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
3237 005340 012706 001114 MOV #CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
3238 005344 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
3239 005346 022706 001154 CMP #SMR, R6 ;;DONE?
3240 005352 001374 BNE .-6 ;;LOOP BACK IF NO
3241 005354 012706 001100 MOV #STACK, SP ;;SETUP THE STACK POINTER
3242
3243 ;;INITIALIZE A FEW VECTORS
3244 005360 012737 060032 000020 MOV #SSCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
3245 005366 012737 000340 000022 MOV #340, @IOTVEC+2 ;;LEVEL 7
3246 005374 012737 060442 000030 MOV #ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
3247 005402 012737 000340 000032 MOV #340, @EMTVEC+2 ;;LEVEL 7
3248 005410 012737 062202 000034 MOV #STRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
3249 005416 012737 000340 000036 MOV #340, @TRAPVEC+2 ;;LEVEL 7
3250 005424 012737 062310 000024 MOV #SPWRDN, @PMRVEC ;;POWER FAILURE VECTOR
3251 005432 012737 000340 000026 MOV #340, @PMRVEC+2 ;;LEVEL 7
3252 005440 013737 032316 032310 MOV SENDCT, SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
3253 005446 005037 001206 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
3254 005452 005037 001210 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
3255 005458 112737 000001 001131 MOVB #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
3256 005464 012737 005464 001122 MOV #., $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
3257 005472 012737 005472 001124 MOV #., $LPERR ;;SETUP THE ERROR LOOP ADDRESS
3258 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3259 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
3260 005500 013746 000004 MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
3261 005504 012737 005540 000004 MOV #65$, @ERRVEC ;;SET UP ERROR VECTOR
3262 005512 012737 177570 001154 MOV #DSMR, SMR ;;SETUP FOR A HARDWARE SWICH REGISTER
3263 005520 012737 177570 001156 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
3264 005526 022777 177777 173420 CMP #-1, @SMR ;;TRY TO REFERENCE HARDWARE SMR
3265 005534 001012 BNE 67$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3266 ;;AND THE HARDWARE SMR IS NOT = -1
3267 005536 000403 BR 66$ ;;BRANCH IF NO TIMEOUT
3268 005540 012716 005546 65$: MOV #66$, (SP) ;;SET UP FOR TRAP RETURN
3269 005544 000002 RTI
3270 005546 012737 000176 001154 66$: MOV #SWREG, SMR ;;POINT TO SOFTWARE SMR
3271 005554 012737 000174 001156 66$: MOV #DISPREG, DISPLAY
3272 005562 012637 000004 67$: MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR
3273 005566 005037 001230 CLR $PASS ;;CLEAR PASS COUNT
3274 005572 132737 000200 001243 BITB #APTSIZE, $ENVM ;;TEST USER SIZE UNDER APT
3275 005600 001403 BEQ 68$ ;;YES, USE NON-APT SWITCH
3276 005602 012737 001244 001154 68$: MOV #SSWREG, SMR ;;NO, USE APT SWITCH REGISTER
3277 005610
    
```



```

3278 .SBTTL TYPE PROGRAM NAME
3279 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
3280 005610 005227 177777 INC 8-1 ;;FIRST TIME?
3281 005614 001060 BNE 69$ ;;BRANCH IF NO
3282 005616 022737 032452 000042 CMP #SENDAD,2#42 ;;ACT-11?
3283 005624 001454 BEQ 69$ ;;BRANCH IF YES
3284 005626 104401 005674 TYPE 70$ ;;TYPE ASCIZ STRING
3285 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
3286 005632 005737 000042 TST 2#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
3287 005636 001012 BNE 71$ ;;BRANCH IF YES
3288 005640 123727 001242 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
3289 005646 001406 BEQ 71$ ;;BRANCH IF YES
3290 005650 023727 001154 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
3291 005656 001005 BNE 72$ ;;BRANCH IF NO
3292 005660 104407 GTSWR ;;GET SOFT-SWR SETTINGS
3293 005662 000403 BR 72$
3294 005664 112737 000001 001150 71$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
3295 005672 72$:
3296 005672 000431 BR 69$ ;;GET OVER THE ASCIZ
3297 ;;70$: .ASCIZ <CRLF>DZRMDA - RMD3 SUBSYSTEM FUNCTIONAL TEST, PART 2<CRLF>
3298 69$:
3299
3300
3301 ;FIND OUT IF PROGRAM IS RUNNING IN STANDALONE MODE
3302 005756 005737 000042 TST 42 ;;IS LOC 42 ZERO ??
3303 005762 001003 BNE 10$ ;;NO - NOT IN STANDALONE
3304 005764 105737 001242 TSTB $ENV ;;IS APT ENVIRONMENT ZERO ??
3305 005770 001411 BEQ STANDALONE ;;YES - PROGRAM IN STANDALONE
3306 005772 10$:
3307
3308 ;PROGRAM NOT RUNNING IN STANDALONE - SEE IF SIZING IS ALLOWED
3309 005772 132737 000200 001243 BITB #BIT7,$ENVM ;;SIZING ALLOWED ??
3310 006000 001003 BNE 20$ ;;NO
3311 006002 012737 000377 001300 MOV #377,$DEVN ;;YES - SET DEVICE MAP FOR ALL DEVICES
3312 006010 20$:
3313
3314 ;GO TO COMMON START CODE
3315 006010 000137 006632 JMP CMNSTART
  
```

```

3316 006014
3317 006014 004737 060652
3318
3319
3320 006020 005327 000001
3321 006024 100024
3322
3323
3324
3325 006026 104401 063204
3326 006032 104411
3327 006034 012637 001176
3328 006040 104401 001176
3329 006044 123727 001176 000131
3330 006052 001002
3331 006054 000137 006764
3332 006060 123727 001176 000116
3333 006066 001420
3334 006070 104401 063053
3335 006074 000754
3336 006076
3337
3338
3339 006076 104401 063055
3340 006102 104411
3341 006104 012637 001176
3342 006110 104401 001176
3343 006114 123727 001176 000131
3344 006122 001002
3345 006124 104401 102574
3346 006130
3347
3348
3349 006130 104401 063111
3350 006134 104411
3351 006136 012637 001176
3352 006142 104401 001176
3353 006146 123727 001176 000131
3354 006154 001137
3355 006156
3356
3357
3358 006156 104401 063243
3359 006162 013746 001276
3360 006166 104402
3361 006170 104401 063044
3362 006174 104413
3363 006176 012637 001176
3364 006202 001412
3365 006204 022737 160000 001176
3366 006212 101403
3367 006214 104401 063270
3368 006220 000756
3369 006222 013737 001176 001276
3370 006230 113737 001272 001176
3371 006236 105037 001177

STANDALONE:
JSR PC,STKINT ;INITIALIZE CONSOLE

;SEE IF THIS IS THE FIRST START AFTER PROGRAM WAS LOADED
DEC #1 ;FIRST START ??
BPL 10$ ;YES !!

;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
5$: TYPE ,CNSLOO ;MAINTAIN PREVIOUS PARAMETERS??
RDCHR ;GET RESPONSE
MOV (SP)+,STMP1 ;ECHO RESPONSE
TYPE ,STMP1
CMPB $STMP1,#'Y ;YES RESPONSE??
BNE 6$ ;NO!!
JMP READY ;KEEP PREVIOUS PARAMETERS
6$: CMPB $STMP1,#'N ;NO RESPONSE??
BEQ 20$ ;GET NEW PARAMETERS
TYPE ,QSTMRK ;NOT YES OR NO, TYPE "??
BR 5$ ;RETRY
10$:

;SEE IF OPERATOR WANTS HELP FILE
TYPE ,HELPOST ;WANT HELP ??
RDCHR ;GET RESPONSE
MOV (SP)+,STMP1 ;SAVE AND ECHO RESPONSE
TYPE ,STMP1
CMPB $STMP1,#'Y ;WAS IT A YES RESPONSE ??
BNE 20$ ;NO - DONT TYPE HELP
TYPE ,HELP ;YES - TYPE HELP TEXT
20$:

;SEE IF USER WANTS TO CHANGE RMO3 UNIBUS ADDRESS
TYPE ,UBUSQST ;WANT TO CHANGE ADDRESS ??
RDCHR ;GET RESPONSE
MOV (SP)+,STMP1 ;SAVE AND ECHO RESPONSE
TYPE ,STMP1
CMPB $STMP1,#'Y ;WAS IT A YES RESPONSE ??
BNE 30$ ;NO !!
30$:

;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, AND INTERRUPT VECTOR
TYPE ,CNSLO1 ;TYPE CURRENT BUS ADDRESS
MOV $BASE,-(SP) ;SAVE $BASE FOR TYPEOUT
TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE ,CLSPRN
RDOCT ;GET NEW BUS ADDRESS
MOV (SP)+,STMP1 ;CARRIAGE RETURN??
BEQ 50$ ;YES-SKIP TO NEXT ENTRY
CMP #160000,STMP1 ;BASE ADDRESS IN I/O PAGE??
BLOS 40$ ;YES
TYPE ,CNSLO2 ;TYPE WARNING MESSAGE
BR 30$ ;RETRY
40$: MOV $STMP1,$BASE ;STORE NEW BUS ADDRESS
50$: MOV $VECT1,$STMP1 ;TYPE CURRENT VECTOR ADDRESS
CLR $STMP1+1

```

3372	006242	104401	063351			TYPE	CNSLO3		
3373	006246	013746	001176			MOV	\$TMP1,-(SP)	::	SAVE \$TMP1 FOR TYPEOUT
3374	006252	104403				TYPOS		::	GO TYPE--OCTAL ASCII
3375	006254	003				.BYTE	3	::	TYPE 3 DIGIT(S)
3376	006255	000				.BYTE	0	::	SUPPRESS LEADING ZEROS
3377	006256	104401	063044			TYPE	,CLSPRN		
3378	006262	104413				RDOCT		::	GET NEW VECTOR ADDRESS
3379	006264	012637	001176			MOV	(SP)+,\$TMP1	::	CARRIAGE RETURN??
3380	006270	001412				BEQ	70\$::	YES-SKIP TO NEXT ENTRY
3381	006272	022737	001000	001176		CMP	#1000,\$TMP1	::	VECTOR ADDRESS < 1000??
3382	006300	101003				BHI	60\$::	YES!!
3383	006302	104401	063401			TYPE	,CNSLO4	::	TYPE WARNING MESSAGE
3384	006306	000750				BR	50\$::	RETRY
3385	006310	113737	001176	001272	60\$:	MOVB	\$TMP1,\$VECT1	::	STORE NEW VECTOR ADDRESS
3386	006316	113737	001273	001176	70\$:	MOVB	\$VECT1+1,\$TMP1	::	TYPE CURRENT PRIORITY
3387	006324	006237	001176			ASR	\$TMP1		
3388	006330	006237	001176			ASR	\$TMP1		
3389	006334	006237	001176			ASR	\$TMP1		
3390	006340	006237	001176			ASR	\$TMP1		
3391	006344	006237	001176			ASR	\$TMP1		
3392	006350	105037	001177			CLRB	\$TMP1+1		
3393	006354	104401	063455			TYPE	,CNSLO5		
3394	006360	013746	001176			MOV	\$TMP1,-(SP)	::	SAVE \$TMP1 FOR TYPEOUT
3395	006364	104403				TYPOS		::	GO TYPE--OCTAL ASCII
3396	006366	001				.BYTE	1	::	TYPE 1 DIGIT(S)
3397	006367	000				.BYTE	0	::	SUPPRESS LEADING ZEROS
3398	006370	104401	063044			TYPE	,CLSPRN		
3399	006374	104413				RDOCT		::	GET NEW PRIORITY
3400	006376	012637	001176			MOV	(SP)+,\$TMP1	::	CARRIAGE RETURN??
3401	006402	001424				BEQ	90\$::	YES-SKIP TO NEXT ENTRY
3402	006404	023727	001176	000007		CMP	\$TMP1,#7	::	LEGAL PRIORITY??
3403	006412	002403				BLT	80\$::	YES!!
3404	006414	104401	063511			TYPE	,CNSLO6	::	TYPE WARNING MESSAGE
3405	006420	000736				BR	70\$::	RETRY
3406	006422				80\$:			::	STORE NEW PRIORITY
3407	006422	006337	001176			ASL	\$TMP1		
3408	006426	006337	001176			ASL	\$TMP1		
3409	006432	006337	001176			ASL	\$TMP1		
3410	006436	006337	001176			ASL	\$TMP1		
3411	006442	006337	001176			ASL	\$TMP1		
3412	006446	113737	001176	001273		MOVB	\$TMP1,\$VECT1+1		
3413	006454								
3414									
3415									
3416	006454	005037	001300			CLR	\$DEVN	::	CLEAR DEVICE MAP
3417	006460	104401	063536			TYPE	,CNSLO7	::	TYPE INPUT INSTRUCTIONS
3418	006464	104401	063047			TYPE	,PROMPT	::	TYPE PROMPTING CHARACTER
3419	006470	104411				RDCHR		::	GET RESPONSE
3420	006472	012637	001176			MOV	(SP)+,\$TMP1	::	ECHO RESPONSE
3421	006476	104401	001176			TYPE	\$TMP1		
3422	006502	023727	001176	000101		CMP	\$TMP1,#'A	::	TEST ALL DRIVES??
3423	006510	001017				BNE	110\$::	NO
3424	006512	012737	000377	001300		MOV	#377,\$DEVN	::	TEST ALL DEVICES
3425	006520	000444				BR	140\$::	SKIP TO NEXT ENTRY
3426	006522	104401	063047		100\$:	TYPE	,PROMPT	::	TYPE PROMPTING CHARACTER
3427	006526	104411				RDCHR		::	GET RESPONSE

; DIALOGUE TO INPUT DEVICE NUMBERS

J06

DZRM DA - RMO3 FUNCTIONAL TEST, PART 2
 DZRM DA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 74
 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0076

3428	006530	012637	001176		MOV	(SP)+,STMP1	;ECHO RESPONSE
3429	006534	104401	001176		TYPE	STMP1	
3430	006540	023727	001176	000015	CMP	\$STMP1,#CR	;CARRIAGE RETURN??
3431	006546	001431			BEQ	140\$	
3432	006550	023727	001176	000060	110\$: CMP	STMP1,#'0	;NUMBER < 0??
3433	006556	002404			BLT	120\$;YES!!
3434	006560	023727	001176	000067	CMP	STMP1,#'7	;NUMBER > 7??
3435	006566	003403			BLE	130\$;NO!!
3436	006570	104401	063053		120\$: TYPE	QSTMRK	;TYPE "?"
3437	006574	000752			BR	100\$;RETRY
3438	006576	013701	001176		130\$: MOV	STMP1,R1	;R1=DRIVE NUMBER
3439	006602	042701	177770		BIC	#1C7,R1	
3440	006606	116102	063776		MOVB	ATNTBL(R1),R2	;DECODE DEVICE NUMBER
3441	006612	042702	177400		BIC	#1C377,R2	;CLEAR UNUSED BITS
3442	006616	050237	001300		BIS	R2,\$DEV#	;SET DEVICE # IN MAP
3443	006622	122737	000377	001300	CMPB	#377,\$DEV#	;DONE ??
3444	006630	101334			BHI	100\$;NO
3445	006632				140\$:		
3446							

```

3447 006632
3448
3449
3450 006632 013700 001300
3451 006636 012701 001450
3452 006642 010137 001446
3453 006646 012702 000001
3454 006652 005003
3455 006654 030200
3456 006656 001406
3457 006660 010311
3458 006662 116361 063776 000001
3459 006670 062701 000002
3460 006674 006302
3461 006676 105702
3462 006700 001402
3463 006702 005203
3464 006704 000763
3465 006706 005011
3466
3467
3468 006710 004737 037500
3469 006714 000403
3470 006716 104000
3471 006720 000000
3472 006722 000775
3473 006724
3474
3475 006724 005737 000042
3476 006730 001012
3477 006732 123727 001242 000001
3478 006740 001406
3479 006742 023727 001154 000176
3480 006750 001005
3481 006752 104407
3482 006754 000403
3483 006756 112737 000001 001150
3484 006764
3485 006764 000240
3486 006766 004737 060652
3487 006772 013746 000300
3488 006776 012746 007004
3489 007002 000002
3490 007004
3491 007004 117737 172436 001234
3492 007012 005037 001472

```

```

CMNSTART:
;ASSEMBLE TEST QUE FROM DEVICE MAP
MOV SDEVN,R0 ;R0 = DEVICE MAP
MOV #TSTQUE+2,R1 ;R1 = ADDRESS OF FIRST ENTRY IN QUE
MOV R1,TSTQUE ;INITIALIZE ENTRY POINTER
MOV #1,R2 ;R2 = DEVICE POINTER
CLR R3 ;R3 = DEVICE NUMBER
10$: BIT R2,R0 ;IS THIS DEVICE IN MAP ??
BEQ 20$ ;NO !!
MOV R3,(R1) ;YES - ENTER DEVICE NUMBER IN QUE
MOVB ATNTBL(R3),1(R1) ;ENTER ATTENTION BIT IN QUE
ADD #2,R1 ;ADVANCE ENTRY POINTER
20$: ASL R2 ;ADVANCE DEVICE POINTER
TSTB R2 ;DONE ALL DEVICES ??
BEQ 25$ ;YES
INC R3 ;ADVANCE DEVICE NUMBER
BR 10$ ;ENTER NEXT DEVICE
25$: CLR (R1) ;TERMINATE TEST QUE

;SIZE FOR CLOCK
JSR PC,SIZCLK ;SEE IF CLOCK PRESENT
BR 40$ ;YES - CLOCK IS PRESENT
30$: ERROR ;NO CLOCK
HALT
BR 30$

40$:
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
TST #42 ;ARE WE RUNNING UNDER XXDP/ACT?
BNE 64$ ;BRANCH IF YES
CMPB $ENV,#1 ;ARE WE RUNNING UNDER APT?
BEQ 64$ ;BRANCH IF YES
CMP SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
BNE 65$ ;BRANCH IF NO
GTSWR ;GET SOFT-SWR SETTINGS
BR 65$
64$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
65$:
READY: NOP ;READY TO START TEST
JSR PC,$TKINT ;INITIALIZE TTY
MOV PR6,-(SP) ;PUT NEW PS ON STACK
MOV #64$,-(SP) ;PUT NEW PC ON STACK
RTI ;POP NEW PC AND PS

64$: MOVB #TSTQUE,$UNIT ;LOAD UNIT NUMBER
CLR MEDENB ;CLEAR MEDIA ENABLE

```

```

3493
3494
3495
3496 007016
3497 007016 000240
3498 007020 012737 007034 001122
3499 007026 012737 007034 001124
3500 007034
3501 007034 012706 001100
3502 007040 013700 001276
3503 007044 013701 001446
3504 007050 012737 000001 001226
3505 007056 005001
3506 007060 013746 000004
3507 007064 013746 000006
3508 007070 012737 007162 000004
3509 007076 012737 000300 000006
3510
3511 007104 110160 000001
3512 007110 010160 000002
3513 007114 016002 000002
3514 007120 010160 000004
3515 007124 016002 000004
3516 007130 010160 000010
3517 007134 016002 000010
3518 007140 010160 000022
3519 007144 016002 000022
3520 007150 012637 000006
3521 007154 012637 000004
3522 007160 000415
3523
3524 007162 022626
3525 007164 012637 000006
3526 007170 012637 000004
3527 007174 104110
3528 007176 005737 000042
3529 007202 001002
3530 007204 000137 005322
3531 007210 000137 032262
3532 007214
3533
3534
3535
3536
3537
3538 007214
3539 007214 000004
3540 007216 000240
3541 007220 012706 001100
3542 007224 013700 001276
3543 007230 013701 001446
3544 007234 012737 000002 001226
3545
3546 007242 004737 046154
3547 007246 000404
3548 007250 000240

;*****
;#TEST 1 CONTROLLER ACCESS TEST
;*****
TST1:
NOP ;START OF TEST
MOV #1$,SLPADR
MOV #1$,SLPERR
1$:
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #1,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
CLR R1
MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
MOV #3$,ERRVEC
MOV #PR6,ERRVEC+2
MOVB R1,RMCS1+1(R0) ;MOVE HI BYTE TO RMCS1
MOV R1,RMWC(R0) ;MOVE WORD COUNT REGISTER
MOV RMWC(R0),R2
MOV R1,RMBA(R0) ;MOVE BUS ADDRESS REGISTER
MOV RMBA(R0),R2
MOV R1,RMCS2(R0) ;MOVE CONTROL STATUS REGISTER
MOV RMCS2(R0),R2
MOV R1,RMDB(R0) ;MOVE DATA BUFFER
MOV RMDB(R0),R2
MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
BR 7$ ;NO BUS TIMEOUT OCCURRED
3$:
CMP (SP)+,(SP)+ ;ADJUST STACK
MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
ERROR 110 ;CANNOT ACCESS MASSBUS CONTROLLER
TST 42 ;STAND ALONE MODE??
BNE 5$ ;NO!!
JMP START ;YES-GO GET $BASE
5$:
JMP SEOP ;GO TO END OF PASS HANDLER
7$:

;*****
;#TEST 2 DEVICE AVAILABLE TEST
;*****
TST2:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
JSR PC,CNTCLR
BR 2$ ;GO TO 2$ IF NO ERROR
NOP ;RETURN HERE IF ERROR

```

```

3549 007252 104000          ERROR
3550 007254 000137 007374 2S:  JMP 7$ ;GO TO 7$ IF ERROR WAS FOUND
3551 007260          ;
3552 007260 013746 000004      MOV ERRVEC -(SP) ;;PUSH ERRVEC ON STACK
3553 007264 013746 000006      MOV ERRVEC+2 -(SP) ;;PUSH ERRVEC+2 ON STACK
3554 007270 012737 007360 000004  MOV #5$,ERRVEC
3555 007276 013737 000300 000006  MOV PR6,ERRVEC+2
3556
3557 007304 016037 000000 001176  MOV RMCS1(RO),STMP1 ;GET DVA STATUS
3558 007312 016037 000010 001174  MOV RMCS2(RO),STMP0 ;GET NED STATUS
3559 007320 012637 000006      MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
3560 007324 012637 000004      MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
3561 007330 032737 010000 001174  BIT #NED,STMP0 ;NONEXISTENT DEVICE??
3562 007336 001402          BEQ 3$ ;NO!!
3563 007340 104111          ERROR 111 ;NONEXISTENT DEVICE
3564 007342 000414          BR 7$
3565 007344 032737 004000 001176 3$: BIT #DVA,STMP1 ;DEVICE AVAILABLE??
3566 007352 001012          BNE 9$ ;YES!!
3567 007354 104112          ERROR 112 ;DEVICE NOT AVAILABLE
3568 007356 000406          BR 7$
3569
3570 007360 022626          5$: CMP (SP)+,(SP)+ ;ADJUST STACK
3571 007362 012637 000006      MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
3572 007366 012637 000004      MOV (SP)+,ERRVEC ;;POP STACK INTO ERRVEC
3573 007372 104113          ERROR 113 ;BUS TIMEOUT (04 TRAP)
3574 007374 000137 032226 7$:  JMP SEOSP
3575
3576 007400          9$:
3577
3578 ;*****
3579 ;*TEST 3 DRIVE TYPE TEST
3580 ;*****
3581
3582 007400          †T3:
3583 007400 000004          SCOPE ;SCOPE CALL
3584 007402 000240          NOP ;START OF TEST
3585 007404 012706 001100      MOV #STACK,SP ;INITIALIZE STACK POINTER
3586 007410 013700 001276      MOV $BASE,RO ;RO=UNIBUS ADDRESS
3587 007414 013701 001446      MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
3588 007420 012737 000003 001226  MOV #3,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
3589
3590 007426 004737 046154      JSR PC,CNTCLR
3591 007432 000404          BR 2$ ;GO TO 2$ IF NO ERROR
3592 007434 000240          NOP ;RETURN HERE IF ERROR
3593 007436 104000          ERROR ;ERROR NUMBER DEFINED BY SUB
3594 007440 000137 007536      JMP 4$ ;GO TO 4$ IF ERROR WAS FOUND
3595
3596 007444          2$:
3597 007444 112737 000026 001504  MOVB #RMDT,GETINX ;SETUP GET INDEX TABLE
3598 007452 112737 000200 001505  MOVB #200,GETINX+1
3599 007460 012737 007542 001354  MOV #5$,RMDTI ;RMDT INPUT BUFFER = 5$
3600 007466 004737 037012      JSR PC,GET ;GO GET DRIVE TYPE
3601 007472 000402          BR 3$ ;GO TO 3$ IF NO ERROR
3602 007474 000240          NOP ;RETURN HERE IF ERROR
3603 007476 104000          ERROR ;ERROR NUMBER SPECIFIED BY GET
3604 007500 022737 020024 001354 3$:  CMP #SNGPRT,RMDTI ;SINGLE PORT RMD3??
3605 007506 001415          BEQ 5$ ;YES!!

```



```

3661 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
3662 007750 004737 036726 JSR PC,GETSTS ;SETUP FOR STATUS
3663 007754 004737 037622 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
3664 007760
3665
3666 ;GO READ SEEK STATUS
3667 007760 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
3668 007764 000404 BR 60$ ;GO TO 60$ IF NO ERROR
3669 007766 000240 NOP ;RETURN HERE IF ERROR
3670 007770 104000 ERROR ;ERROR DEFINED BY GET SUB
3671 007772 000137 010342 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3672 007776
3673
3674 ;VERIFY THE RESULTS OF THE SEEK COMMAND
3675 007776 004737 044714 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
3676 010002 000405 BR 70$ ;GO TO 70$ IF NO ERROR
3677 010004 000240 NOP ;RETURN HERE IF ERROR
3678 010006 104000 ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
3679 010010 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3680 010012 000137 010342 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3681 010016
3682
3683 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
3684 010016 012737 000063 001376 MOV #MH!GO, RMCS10 ;LOAD WRITE HEADER AND DATA
3685 010024 012702 001536 MOV #PUTINX+3, R2 ;EXTEND REGISTER INDEX TABLE
3686 010030 112722 000002 MOVB #RMC, (R2)+
3687 010034 112722 000004 MOVB #RMB, (R2)+
3688 010040 112722 000000 MOVB #RMS1, (R2)+
3689 010044 112722 000200 MOVB #200, (R2)+
3690 010050 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
3691 010054 000404 BR 80$ ;GO TO 80$ IF NO ERROR
3692 010056 000240 NOP ;RETURN HERE IF ERROR
3693 010060 104000 ERROR ;ERROR DEFINED BY PUT SUB
3694 010062 000137 010342 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3695 010066
3696
3697 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
3698 010066 004737 037622 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
3699 010072 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
3700 010076 000404 BR 90$ ;GO TO 90$ IF NO ERROR
3701 010100 000240 NOP ;RETURN HERE IF ERROR
3702 010102 104000 ERROR ;ERROR DEFINED BY GET SUB
3703 010104 000137 010342 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3704 010110
3705
3706 ;VERIFY RESULTS OF WRITE COMMAND
3707 010110 004737 040006 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
3708 010114 000405 BR 100$ ;GO TO 100$ IF NO ERROR
3709 010116 000240 NOP ;RETURN HERE IF ERROR
3710 010120 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
3711 010122 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3712 010124 000137 010342 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3713 010130
3714 010130 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3715 010134 000405 BR 110$ ;GO TO 110$ IF NO ERROR
3716 010136 000240 NOP ;RETURN HERE IF ERROR

```

```

3717 010140 104000          ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
3718 010142 004736          JSR          PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3719 010144 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3720 010150          110$:
3721 010150 004737 040640          JSR          PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
3722 010154 000405          BR          120$      ;GO TO 120$ IF NO ERROR
3723 010156 000240          NOP          ;RETURN HERE IF ERROR
3724 010160 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
3725 010162 004736          JSR          PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3726 010164 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3727 010170          120$:
3728
3729
3730          ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
3731 010170 012737 000073 001376   MOV          #RH!GO, RMC510 ;READ HEADER & DATA COMMAND
3732 010176 012737 103600 001402   MOV          #BUFTWO, RMBAD ;CHANGE BUS ADDRESS
3733 010204 004737 037262          JSR          PC, PUT    ;GO WRITE REGISTERS VIA PUT SUB
3734 010210 000404          BR          130$      ;GO TO 130$ IF NO ERROR
3735 010212 000240          NOP          ;RETURN HERE IF ERROR
3736 010214 104000          ERROR          ;ERROR DEFINED BY PUT SUB
3737 010216 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3738 010222          130$:
3739
3740          ;WAIT FOR READ TO COMPLETE AND GET STATUS
3741 010222 004737 037622          JSR          PC, TIMEOUT ;WAIT FOR READ TO COMPLETE
3742 010226 004737 037012          JSR          PC, GET    ;GO READ REGISTERS VIA GET SUB
3743 010232 000404          BR          140$      ;GO TO 140$ IF NO ERROR
3744 010234 000240          NOP          ;RETURN HERE IF ERROR
3745 010236 104000          ERROR          ;ERROR DEFINED BY GET SUB
3746 010240 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3747 010244          140$:
3748
3749          ;VERIFY THE RESULTS OF READ OPERATION
3750 010244 004737 040006          JSR          PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
3751 010250 000405          BR          150$      ;GO TO 150$ IF NO ERROR
3752 010252 000240          NOP          ;RETURN HERE IF ERROR
3753 010254 104000          ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
3754 010256 004736          JSR          PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3755 010260 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3756 010264          150$:
3757 010264 004737 052312          JSR          PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3758 010270 000405          BR          160$      ;GO TO 160$ IF NO ERROR
3759 010272 000240          NOP          ;RETURN HERE IF ERROR
3760 010274 104000          ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
3761 010276 004736          JSR          PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3762 010300 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3763 010304          160$:
3764 010304 004737 040640          JSR          PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
3765 010310 000405          BR          170$      ;GO TO 170$ IF NO ERROR
3766 010312 000240          NOP          ;RETURN HERE IF ERROR
3767 010314 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
3768 010316 004736          JSR          PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3769 010320 000137 010342   JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3770 010324          170$:
3771
3772          ;VERIFY DATA

```

3773	010324	004737	036360	JSR	PC,CMPBUF	;GO COMPARE WRITE, READ DATA BUFFERS
3774	010330	102574		.WORD	BUFONE	;STARTING ADDRESS OF WRITE BUFFER
3775	010332	103600		.WORD	BUFTWO	;STARTING ADDRESS OF READ BUFFER
3776	010334	000402		BR	180\$;GO TO 180\$ IF NO ERROR
3777	010336	000240		NOP		;RETURN HERE IF ERROR
3778	010340	104000		ERROR		;ERROR # DEFINED BY CMPBUF SUBROUTINE

180\$:

```

;*****
;TEST 5      FORMAT ZEROS - 16
;*****

```

3784	010342			TST5:	SCOPE	;SCOPE CALL
3785	010342	000004			NOP	;START OF TEST
3786	010344	000240			MOV	#STACK,SP
3787	010346	012706	001100		MOV	\$BASE,R0
3788	010352	013700	001276		MOV	TSTQUE,R1
3789	010356	013701	001446		MOV	#5,\$STESTN
3790	010362	012737	000005	001226	MOV	

10\$:

;SETUP PARAMETERS FOR GENERATING DATA BUFFER

3794	010370	012737	000000	001432	MOV	#0,RMDCO	;CYLINDER = 0
3795	010376	012737	000000	001404	MOV	#0,RMDAO	;TRACK = SECTOR = 0
3796	010404	012737	010000	001430	MOV	#FMT16,RMOFO	;16 BIT FORMAT
3797	010412	012737	177376	001400	MOV	#(2+256)+1,RMWC0	;2 + 256 WORDS
3798	010420	012737	102574	001402	MOV	#BUFONE,RMBA0	;DATA BUFFER ADDRESS
3799	010426	012737	000062	001376	MOV	#MH,RMCS10	;WRITE HEADER AND DATA
3800	010434	012737	064110	001174	MOV	#ZEROS,\$TMP0	;USE ALL ZEROS DATA PATTERN
3801	010442	012737	000001	001176	MOV	#1,\$TMP1	
3802	010450	004737	036114		JSR	PC,GENBUF	;GO GENERATE DATA BUFFER

20\$:

;PREPARE DEVICE FOR DATA TRANSFER

3806	010454	004737	033274		JSR	PC,TSTPRP	;PREPARE DEVICE FOR TEST
3807	010460	154130			.WORD	154130	;TASK DESCRIPTOR
3808	010462	000404			BR	30\$;GO TO 30\$ IF NO ERROR
3809	010464	000240			NOP		;RETURN HERE IF ERROR
3810	010466	104000			ERROR		;ERROR # DEFINED BY TSTPRP SUBROUTINE
3811	010470	000137	011142		JMP	180\$;GO TO 180\$ IF ERROR WAS FOUND

30\$:

;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER

3815	010474	012737	000005	001376	MOV	#SEEK!GO,RMCS10	;CHANGE COMMAND TO SEEK
3816	010502	012702	001533		MOV	#PUTINX,R2	;WRITE REGISTER INDEX TABLE
3817	010506	112722	000006		MOVB	#RMDA,(R2)+	
3818	010512	112722	000034		MOVB	#RMDC,(R2)+	
3819	010516	112722	000032		MOVB	#RMOF,(R2)+	
3820	010522	112722	000000		MOVB	#RMCS1,(R2)+	
3821	010526	112722	000200		MOVB	#200,(R2)+	
3822	010532	004737	037262		JSR	PC,PUT	;GO WRITE REGISTERS VIA PUT SUB
3823	010536	000404			BR	40\$;GO TO 40\$ IF NO ERROR
3824	010540	000240			NOP		;RETURN HERE IF ERROR
3825	010542	104000			ERROR		;ERROR DEFINED BY PUT SUB
3826	010544	000137	011142		JMP	180\$;GO TO 180\$ IF ERROR WAS FOUND

40\$:

3828

```

3829 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
3830 010550 004737 036726 JSR PC,GETSTS ;SETUP FOR STATUS
3831 010554 004737 037622 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
3832 010560
3833
3834 ;GO READ SEEK STATUS
3835 010560 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
3836 010564 000404 BR 60$ ;GO TO 60$ IF NO ERROR
3837 010566 000240 NOP ;RETURN HERE IF ERROR
3838 010570 104000 ERROR ;ERROR DEFINED BY GET SUB
3839 010572 000137 011142 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3840 010576
3841
3842 ;VERIFY THE RESULTS OF THE SEEK COMMAND
3843 010576 004737 044714 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
3844 010602 000405 BR 70$ ;GO TO 70$ IF NO ERROR
3845 010604 000240 NOP ;RETURN HERE IF ERROR
3846 010606 104000 ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
3847 010610 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3848 010612 000137 011142 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3849 010616
3850
3851 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
3852 010616 012737 000063 001376 MOV #WH!GO, RMCS10 ;LOAD WRITE HEADER AND DATA
3853 010624 012702 001536 MOV #PUTINX+3, R2 ;EXTEND REGISTER INDEX TABLE
3854 010630 112722 000002 MOVB #RMC, (R2)+
3855 010634 112722 000004 MOVB #RMB, (R2)+
3856 010640 112722 000000 MOVB #RMC1, (R2)+
3857 010644 112722 000200 MOVB #200, (R2)+
3858 010650 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
3859 010654 000404 BR 80$ ;GO TO 80$ IF NO ERROR
3860 010656 000240 NOP ;RETURN HERE IF ERROR
3861 010660 104000 ERROR ;ERROR DEFINED BY PUT SUB
3862 010662 000137 011142 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3863 010666
3864
3865 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
3866 010666 004737 037622 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
3867 010672 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
3868 010676 000404 BR 90$ ;GO TO 90$ IF NO ERROR
3869 010700 000240 NOP ;RETURN HERE IF ERROR
3870 010702 104000 ERROR ;ERROR DEFINED BY GET SUB
3871 010704 000137 011142 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3872 010710
3873
3874 ;VERIFY RESULTS OF WRITE COMMAND
3875 010710 004737 040006 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
3876 010714 000405 BR 100$ ;GO TO 100$ IF NO ERROR
3877 010716 000240 NOP ;RETURN HERE IF ERROR
3878 010720 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
3879 010722 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3880 010724 000137 011142 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
3881 010730
3882 010730 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3883 010734 000405 BR 110$ ;GO TO 110$ IF NO ERROR
3884 010736 000240 NOP ;RETURN HERE IF ERROR

```

```

3885 010740 104000          ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
3886 010742 004736          JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3887 010744 000137 011142    JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3888 010750          110$:
3889 010750 004737 040640    JSR          PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
3890 010754 000405          BR          120$      ;GO TO 120$ IF NO ERROR
3891 010756 000240          NOP          ;RETURN HERE IF ERROR
3892 010760 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
3893 010762 004736          JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3894 010764 000137 011142    JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3895 010770          120$:
3896
3897
3898
3899 010770 012737 000073 001376 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
3900 010776 012737 103600 001402    MOV          #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
3901 011004 004737 037262    MOV          #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
3902 011010 000404          JSR          PC,PUT    ;GO WRITE REGISTERS VIA PUT SUB
3903 011012 000240          BR          130$      ;GO TO 130$ IF NO ERROR
3904 011014 104000          NOP          ;RETURN HERE IF ERROR
3905 011016 000137 011142    ERROR        ;ERROR DEFINED BY PUT SUB
3906 011022          JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3907
3908
3909 011022 004737 037622    ;WAIT FOR READ TO COMPLETE AND GET STATUS
3910 011026 004737 037012    JSR          PC,TIMOUT ;WAIT FOR READ TO COMPLETE
3911 011032 000404          JSR          PC,GET    ;GO READ REGISTERS VIA GET SUB
3912 011034 000240          BR          140$      ;GO TO 140$ IF NO ERROR
3913 011036 104000          NOP          ;RETURN HERE IF ERROR
3914 011040 000137 011142    ERROR        ;ERROR DEFINED BY GET SUB
3915 011044          JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3916
3917
3918 011044 004737 040006    ;VERIFY THE RESULTS OF READ OPERATION
3919 011050 000405          JSR          PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
3920 011052 000240          BR          150$      ;GO TO 150$ IF NO ERROR
3921 011054 104000          NOP          ;RETURN HERE IF ERROR
3922 011056 004736          ERROR        ;ERROR # DEFINED BY PRIERR SUBROUTINE
3923 011060 000137 011142    JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3924 011064          JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3925 011064 004737 052312    150$:
3926 011070 000405          JSR          PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3927 011072 000240          BR          160$      ;GO TO 160$ IF NO ERROR
3928 011074 104000          NOP          ;RETURN HERE IF ERROR
3929 011076 004736          ERROR        ;ERROR # DEFINED BY DTASTS SUBROUTINE
3930 011100 000137 011142    JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3931 011104          JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3932 011104 004737 040640    160$:
3933 011110 000405          JSR          PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
3934 011112 000240          BR          170$      ;GO TO 170$ IF NO ERROR
3935 011114 104000          NOP          ;RETURN HERE IF ERROR
3936 011116 004736          ERROR        ;ERROR # DEFINED BY SECERR SUBROUTINE
3937 011120 000137 011142    JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3938 011124          JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
3939
3940          ;VERIFY DATA

```

3941 011124 004737 036360
3942 011130 102574
3943 011132 103600
3944 011134 000402
3945 011136 000240
3946 011140 104000
3947 011142

JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
.WORD BUFONE ;STARTING ADDRESS OF WRITE BUFFER
.WORD BUFTWO ;STARTING ADDRESS OF READ BUFFER
BR 180\$;GO TO 180\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE

180\$:

;TEST 6 ZERO FILL TEST

TST6:

3952 011142
3953 011142 000004
3954 011144 000240
3955 011146 012706 001100
3956 011152 013700 001276
3957 011156 013701 001446
3958 011162 012737 000006 001226
3959 011170

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV \$BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #6,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10\$:

;SETUP PARAMETERS FOR GENERATING DATA BUFFER

3962 011170 012737 000000 001432
3963 011176 012737 000000 001404
3964 011204 012737 010000 001430
3965 011212 012737 177376 001400
3966 011220 012737 102574 001402
3967 011226 012737 000062 001376
3968 011234 012737 064110 001174
3969 011242 012737 000001 001176
3970 011250 004737 036114
3971 011254

MOV #0,RMDC0 ;CYLINDER = 0
MOV #0,RMDA0 ;TRACK = SECTOR = 0
MOV #FAT16,RMOFO ;16 BIT FORMAT
MOV #(<C<2+256.>+1),RMWCO ;2 + 256 WORDS
MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
MOV #WH,RMCS10 ;WRITE HEADER AND DATA
MOV #ZEROS,\$TMPD ;USE ALL ZEROS DATA PATTERN
MOV #1,\$TMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20\$:

;PREPARE DEVICE FOR DATA TRANSFER

3973
3974 011254 004737 033274
3975 011260 154130
3976 011262 000404
3977 011264 000240
3978 011266 104000
3979 011270 000137 011756
3980 011274

JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
.WORD 154130 ;TASK DESCRIPTOR
BR 30\$;GO TO 30\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 180\$;GO TO 180\$ IF ERROR WAS FOUND

30\$:

;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER

3982
3983 011274 012737 000005 001376
3984 011302 012702 001533
3985 011306 112722 000006
3986 011312 112722 000034
3987 011316 112722 000032
3988 011322 112722 000000
3989 011326 112722 000200
3990 011332 004737 037262
3991 011336 000404
3992 011340 000240
3993 011342 104000
3994 011344 000137 011756
3995 011350
3996

MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 40\$;GO TO 40\$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 180\$;GO TO 180\$ IF ERROR WAS FOUND

40\$:

```

3997
3998 011350 004737 036726
3999 011354 004737 037622
4000 011360
4001
4002
4003 011360 004737 037012
4004 011364 000404
4005 011366 000240
4006 011370 104000
4007 011372 000137 011756
4008 011376
4009
4010
4011 011376 004737 044714
4012 011402 000405
4013 011404 000240
4014 011406 104000
4015 011410 004736
4016 011412 000137 011756
4017 011416
4018
4019
4020 011416 012737 177776 001400
4021 011424 012737 000063 001376
4022 011432 012702 001536
4023 011436 112722 000002
4024 011442 112722 000004
4025 011446 112722 000000
4026 011452 112722 000200
4027 011456 004737 037262
4028 011462 000404
4029 011464 000240
4030 011466 104000
4031 011470 000137 011756
4032 011474
4033
4034
4035 011474 004737 037622
4036 011500 004737 037012
4037 011504 000404
4038 011506 000240
4039 011510 104000
4040 011512 000137 011756
4041 011516
4042
4043
4044 011516 004737 040006
4045 011522 000405
4046 011524 000240
4047 011526 104000
4048 011530 004736
4049 011532 000137 011756
4050 011536
4051 011536 004737 052312
4052 011542 000405

```

```

;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
      JSR   PC,GETSTS           ;SETUP FOR STATUS
      JSR   PC,TIMOUT          ;WAIT FOR SEEK TO COMPLETE
50$:
;GO READ SEEK STATUS
      JSR   PC,GET             ;GO READ REGISTERS VIA GET SUB
      BR    60$                ;GO TO 60$ IF NO ERROR
      NOP                                ;RETURN HERE IF ERROR
      ERROR                                ;ERROR DEFINED BY GET SUB
      JMP   180$               ;GO TO 180$ IF ERROR WAS FOUND
60$:
;VERIFY THE RESULTS OF THE SEEK COMMAND
      JSR   PC,SEKSTS          ;GO VERIFY RESULTS OF SEEK OPERATION
      BR    70$                ;GO TO 70$ IF NO ERROR
      NOP                                ;RETURN HERE IF ERROR
      ERROR                                ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
      JMP   180$               ;GO TO 180$ IF ERROR WAS FOUND
70$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
      MOV   #(<C2+1),RMWC      ;FORMAT PARTIAL SECTOR
      MOV   #WH!GO,RMCS10     ;LOAD WRITE HEADER AND DATA
      MOV   #PUTINX+3,R2      ;EXTEND REGISTER INDEX TABLE
      MOVB  #RMWC,(R2)+
      MOVB  #RMBR,(R2)+
      MOVB  #RMCS1,(R2)+
      MOVB  #200,(R2)+
      JSR   PC,PUT             ;GO WRITE REGISTERS VIA PUT SUB
      BR    80$                ;GO TO 80$ IF NO ERROR
      NOP                                ;RETURN HERE IF ERROR
      ERROR                                ;ERROR DEFINED BY PUT SUB
      JMP   180$               ;GO TO 180$ IF ERROR WAS FOUND
80$:
;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
      JSR   PC,TIMOUT          ;WAIT FOR COMMAND TO COMPLETE
      JSR   PC,GET             ;GO READ REGISTERS VIA GET SUB
      BR    90$                ;GO TO 90$ IF NO ERROR
      NOP                                ;RETURN HERE IF ERROR
      ERROR                                ;ERROR DEFINED BY GET SUB
      JMP   180$               ;GO TO 180$ IF ERROR WAS FOUND
90$:
;VERIFY RESULTS OF WRITE COMMAND
      JSR   PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
      BR    100$              ;GO TO 100$ IF NO ERROR
      NOP                                ;RETURN HERE IF ERROR
      ERROR                                ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
      JMP   180$               ;GO TO 180$ IF ERROR WAS FOUND
100$:
      JSR   PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
      BR    110$              ;GO TO 110$ IF NO ERROR

```

4053	011544	000240			NOP				;RETURN HERE IF ERROR
4054	011546	104000			ERROR				;ERROR # DEFINED BY DTASTS SUBROUTINE
4055	011550	004736			JSR	PC,2(SP)+			;GO BACK FOR MORE ERROR CHECKS
4056	011552	000137	011756		JMP	180\$;GO TO 180\$ IF ERROR WAS FOUND
4057	011556								
4058	011556	004737	040640		JSR	PC,SECERR		110\$:	;GO CHECK FOR SECONDARY ERRORS
4059	011562	000405			BR	120\$;GO TO 120\$ IF NO ERROR
4060	011564	000240			NOP				;RETURN HERE IF ERROR
4061	011566	104000			ERROR				;ERROR # DEFINED BY SECERR SUBROUTINE
4062	011570	004736			JSR	PC,2(SP)+			;GO BACK FOR MORE ERROR CHECKS
4063	011572	000137	011756		JMP	180\$		120\$:	;GO TO 180\$ IF ERROR WAS FOUND
4064	011576								
4065									
4066									
4067									
4068	011576	012737	177376	001400	;READ HEADER AND DATA FOR SECTOR JUST WRITTEN				
4069	011604	012737	000073	001376	MOV	#(C<2+256.>+1),RMC0			;READ A FULL SECTOR
4070	011612	012737	103600	001402	MOV	#RH!GO,RMCS10			;READ HEADER & DATA COMMAND
4071	011620	004737	037262		MOV	#BUFTW0,RMBA0			;CHANGE BUS ADDRESS
4072	011624	000404			JSR	PC,PUT			;GO WRITE REGISTERS VIA PUT SUB
4073	011626	000240			BR	130\$;GO TO 130\$ IF NO ERROR
4074	011630	104000			NOP				;RETURN HERE IF ERROR
4075	011632	000137	011756		ERROR				;ERROR DEFINED BY PUT SUB
4076	011636				JMP	180\$		130\$:	;GO TO 180\$ IF ERROR WAS FOUND
4077									
4078									
4079	011636	004737	037622		;WAIT FOR READ TO COMPLETE AND GET STATUS				
4080	011642	004737	037012		JSR	PC,TIMOUT			;WAIT FOR READ TO COMPLETE
4081	011646	000404			JSR	PC,GET			;GO READ REGISTERS VIA GET SUB
4082	011650	000240			BR	140\$;GO TO 140\$ IF NO ERROR
4083	011652	104000			NOP				;RETURN HERE IF ERROR
4084	011654	000137	011756		ERROR				;ERROR DEFINED BY GET SUB
4085	011660				JMP	180\$		140\$:	;GO TO 180\$ IF ERROR WAS FOUND
4086									
4087									
4088	011660	004737	040006		;VERIFY THE RESULTS OF READ OPERATION				
4089	011664	000405			JSR	PC,PRIERR			;GO CHECK FOR PRIMARY ERRORS
4090	011666	000240			BR	150\$;GO TO 150\$ IF NO ERROR
4091	011670	104000			NOP				;RETURN HERE IF ERROR
4092	011672	004736			ERROR				;ERROR # DEFINED BY PRIERR SUBROUTINE
4093	011674	000137	011756		JSR	PC,2(SP)+			;GO BACK FOR MORE ERROR CHECKS
4094	011700				JMP	180\$		150\$:	;GO TO 180\$ IF ERROR WAS FOUND
4095	011700	004737	052312		JSR	PC,DTASTS			;GO VERIFY RESULTS OF DATA TRANSFER
4096	011704	000405			BR	160\$;GO TO 160\$ IF NO ERROR
4097	011706	000240			NOP				;RETURN HERE IF ERROR
4098	011710	104000			ERROR				;ERROR # DEFINED BY DTASTS SUBROUTINE
4099	011712	004736			JSR	PC,2(SP)+			;GO BACK FOR MORE ERROR CHECKS
4100	011714	000137	011756		JMP	180\$		160\$:	;GO TO 180\$ IF ERROR WAS FOUND
4101	011720								
4102	011720	004737	040640		JSR	PC,SECERR			;GO CHECK FOR SECONDARY ERRORS
4103	011724	000405			BR	170\$;GO TO 170\$ IF NO ERROR
4104	011726	000240			NOP				;RETURN HERE IF ERROR
4105	011730	104000			ERROR				;ERROR # DEFINED BY SECERR SUBROUTINE
4106	011732	004736			JSR	PC,2(SP)+			;GO BACK FOR MORE ERROR CHECKS
4107	011734	000137	011756		JMP	180\$		170\$:	;GO TO 180\$ IF ERROR WAS FOUND
4108	011740								


```

4109
4110
4111 011740 004737 036360
4112 011744 102574
4113 011746 103600
4114 011750 000402
4115 011752 000240
4116 011754 104000
4117 011756
4118
4119
4120
4121
4122 011756
4123 011756 000004
4124 011760 000240
4125 011762 012706 001100
4126 011766 013700 001276
4127 011772 013701 001446
4128 011776 012737 000007 001226
4129 012004
4130
4131
4132 012004 012737 000000 001432
4133 012012 012737 000000 001404
4134 012020 012737 010000 001430
4135 012026 012737 177376 001400
4136 012034 012737 102574 001402
4137 012042 012737 000062 001376
4138 012050 012737 064110 001174
4139 012056 012737 000001 001176
4140 012064 004737 036114
4141 012070
4142
4143
4144 012070 004737 033274
4145 012074 154130
4146 012076 000404
4147 012100 000240
4148 012102 104000
4149 012104 000137 012532
4150 012110
4151
4152
4153 012110 012737 000005 001376
4154 012116 012702 001533
4155 012122 112722 000006
4156 012126 112722 000034
4157 012132 112722 000032
4158 012136 112722 000000
4159 012142 112722 000200
4160 012146 004737 037262
4161 012152 000404
4162 012154 000240
4163 012156 104000
4164 012160 000137 012532

;VERIFY DATA
JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
;STARTING ADDRESS OF WRITE BUFFER
;STARTING ADDRESS OF READ BUFFER
;GO TO 180$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY CMPBUF SUBROUTINE
180$:

;*****
;#TEST 7 FORMAT CHECK ZEROS - 16
;*****
TST7:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
10$:

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = SECTOR = 0
MOV #FAT16,RMOFO ;16 BIT FORMAT
MOV #(<↑C(2+256.)+1>),RMWCO ;2 + 256 WORDS
MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
MOV #MH,RMCS10 ;WRITE HEADER AND DATA
MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
MOV #1,$TMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER
20$:

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
;TASK DESCRIPTOR
;GO TO 30$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 260$ IF ERROR WAS FOUND
30$:

;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

```

```

4165 012164
4166
4167
4168 012164 004737 036726
4169 012170 004737 037622
4170 012174
4171
4172
4173 012174 004737 037012
4174 012200 000404
4175 012202 000240
4176 012204 104000
4177 012206 000137 012532
4178 012212
4179
4180
4181 012212 004737 044714
4182 012216 000405
4183 012220 000240
4184 012222 104000
4185 012224 004736
4186 012226 000137 012532
4187 012232
4188
4189
4190 012232 012737 000063 001376
4191 012240 012702 001536
4192 012244 112722 000002
4193 012250 112722 000004
4194 012254 112722 000000
4195 012260 112722 000200
4196 012264 004737 037262
4197 012270 000404
4198 012272 000240
4199 012274 104000
4200 012276 000137 012532
4201 012302
4202
4203
4204 012302 004737 037622
4205 012306 004737 037012
4206 012312 000404
4207 012314 000240
4208 012316 104000
4209 012320 000137 012532
4210 012324
4211
4212
4213 012324 004737 040006
4214 012330 000405
4215 012332 000240
4216 012334 104000
4217 012336 004736
4218 012340 000137 012532
4219 012344
4220 012344 004737 052312
    
```

```

40$:
;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
    JSR    PC,GETSTS      ;SETUP FOR STATUS
    JSR    PC,TIMOUT     ;WAIT FOR SEEK TO COMPLETE
50$:
;GO READ SEEK STATUS
    JSR    PC,GET        ;GO READ REGISTERS VIA GET SUB
    BR     60$           ;GO TO 60$ IF NO ERROR
    NOP
    ERROR  ;RETURN HERE IF ERROR
    ;ERROR DEFINED BY GET SUB
    JMP    260$         ;GO TO 260$ IF ERROR WAS FOUND
60$:
;VERIFY THE RESULTS OF THE SEEK COMMAND
    JSR    PC,SEKSTS     ;GO VERIFY RESULTS OF SEEK OPERATION
    BR     70$           ;GO TO 70$ IF NO ERROR
    NOP
    ERROR  ;RETURN HERE IF ERROR
    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
    JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
    JMP    260$         ;GO TO 260$ IF ERROR WAS FOUND
70$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
    MOV    #WH!GO, RMCS10 ;LOAD WRITE HEADER AND DATA
    MOV    #PUTINX+3, R2  ;EXTEND REGISTER INDEX TABLE
    MOVB   #RMC, (R2)+
    MOVB   #RMA, (R2)+
    MOVB   #RMS1, (R2)+
    MOVB   #200, (R2)+
    JSR    PC,PUT        ;GO WRITE REGISTERS VIA PUT SUB
    BR     80$           ;GO TO 80$ IF NO ERROR
    NOP
    ERROR  ;RETURN HERE IF ERROR
    ;ERROR DEFINED BY PUT SUB
    JMP    260$         ;GO TO 260$ IF ERROR WAS FOUND
80$:
;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
    JSR    PC,TIMOUT     ;WAIT FOR COMMAND TO COMPLETE
    JSR    PC,GET        ;GO READ REGISTERS VIA GET SUB
    BR     90$           ;GO TO 90$ IF NO ERROR
    NOP
    ERROR  ;RETURN HERE IF ERROR
    ;ERROR DEFINED BY GET SUB
    JMP    260$         ;GO TO 260$ IF ERROR WAS FOUND
90$:
;VERIFY RESULTS OF WRITE COMMAND
    JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
    BR     100$          ;GO TO 100$ IF NO ERROR
    NOP
    ERROR  ;RETURN HERE IF ERROR
    ;ERROR # DEFINED BY PRIERR SUBROUTINE
    JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
    JMP    260$         ;GO TO 260$ IF ERROR WAS FOUND
100$:
    JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
    
```

```

4221 012350 000405 BR 110$ ;GO TO 110$ IF NO ERROR
4222 012352 000240 NOP ;RETURN HERE IF ERROR
4223 012354 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4224 012356 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4225 012360 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4226 012364 110$:
4227 012364 004737 040640 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4228 012370 000405 BR 120$ ;GO TO 120$ IF NO ERROR
4229 012372 000240 NOP ;RETURN HERE IF ERROR
4230 012374 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
4231 012376 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4232 012400 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4233 012404 120$:
4234
4235
4236 ;WRITE CHECK HEADER AND DATA FOR SECTOR JUST WRITTEN
4237 012404 012737 000053 001376 MOV #WCH!GO, RMCS10 ;WRITE CHECK COMMAND
4238 012412 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
4239 012416 000404 BR 130$ ;GO TO 130$ IF NO ERROR
4240 012420 000240 NOP ;RETURN HERE IF ERROR
4241 012422 104000 ERROR ;ERROR DEFINED BY PUT SUB
4242 012424 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4243 012430 130$:
4244
4245 ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
4246 012430 004737 037622 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
4247 012434 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
4248 012440 000404 BR 140$ ;GO TO 140$ IF NO ERROR
4249 012442 000240 NOP ;RETURN HERE IF ERROR
4250 012444 104000 ERROR ;ERROR DEFINED BY GET SUB
4251 012446 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4252 012452 140$:
4253
4254 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
4255 012452 004737 040006 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4256 012456 000405 BR 150$ ;GO TO 150$ IF NO ERROR
4257 012460 000240 NOP ;RETURN HERE IF ERROR
4258 012462 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
4259 012464 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4260 012466 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4261 012472 150$:
4262 012472 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4263 012476 000405 BR 160$ ;GO TO 160$ IF NO ERROR
4264 012500 000240 NOP ;RETURN HERE IF ERROR
4265 012502 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4266 012504 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4267 012506 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4268 012512 160$:
4269 012512 004737 040640 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4270 012516 000405 BR 170$ ;GO TO 170$ IF NO ERROR
4271 012520 000240 NOP ;RETURN HERE IF ERROR
4272 012522 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
4273 012524 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4274 012526 000137 012532 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4275 012532 170$:
4276

```

```

4277 012532
4278
4279
4280
4281
4282 012532
4283 012532 000004
4284 012534 000240
4285 012536 012706 001100
4286 012542 013700 001276
4287 012546 013701 001446
4288 012552 012737 000010 001226
4289 012560
4290
4291
4292 012560 012737 000000 001432
4293 012566 012737 000000 001404
4294 012574 012737 010000 001430
4295 012602 012737 177376 001400
4296 012610 012737 102574 001402
4297 012616 012737 000062 001376
4298 012624 012737 064110 001174
4299 012632 012737 000001 001176
4300 012640 004737 036114
4301 012644
4302
4303
4304 012644 004737 033274
4305 012650 154130
4306 012652 000404
4307 012654 000240
4308 012656 104000
4309 012660 000137 013452
4310 012664
4311
4312
4313 012664 012737 000005 001376
4314 012672 012702 001533
4315 012676 112722 000006
4316 012702 112722 000034
4317 012706 112722 000032
4318 012712 112722 000000
4319 012716 112722 000200
4320 012722 004737 037262
4321 012726 000404
4322 012730 000240
4323 012732 104000
4324 012734 000137 013452
4325 012740
4326
4327
4328 012740 004737 036726
4329 012744 004737 037622
4330 012750
4331
4332

```

260\$:

```

;*****
;TEST 10 FORMAT CHECK ZEROS W/ WCE ERROR
;*****
TST10:

```

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK, SP ;INITIALIZE STACK POINTER
MOV $BASE, R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
MOV #10, $TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

10\$:

```

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0, RMDCO ;CYLINDER = 0
MOV #0, RMDAO ;TRACK = SECTOR = 0
MOV #FAT16, RMOFO ;16 BIT FORMAT
MOV #(<↑C(2+256.)+1>), RMWCO ;2 + 256 WORDS
MOV #BUFONE, RMBAO ;DATA BUFFER ADDRESS
MOV #WH, RMC$10 ;WRITE HEADER AND DATA
MOV #ZEROS, $TMP0 ;USE ALL ONES DATA PATTERN
MOV #1, $TMP1
JSR PC, GENBUF ;GO GENERATE DATA BUFFER

```

20\$:

```

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC, TSTPRP ;PREPARE DEVICE FOR TEST
;WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

```

30\$:

```

;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
MOV #SEEK!GO, RMC$10 ;CHANGE COMMAND TO SEEK
MOV #PUTINX, R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA, (R2)+
MOVB #RMDC, (R2)+
MOVB #RMOF, (R2)+
MOVB #RMC$1, (R2)+
MOVB #200, (R2)+
JSR PC, PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR DEFINED BY PUT SUB
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

```

40\$:

```

;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
JSR PC, GETSTS ;SETUP FOR STATUS
JSR PC, TIMEOUT ;WAIT FOR SEEK TO COMPLETE

```

50\$:

;GO READ SEEK STATUS

4333	012750	004737	037012		JSR	PC,GET		;GO READ REGISTERS VIA GET SUB
4334	012754	000404			BR	60\$;GO TO 60\$ IF NO ERROR
4335	012756	000240			NOP			;RETURN HERE IF ERROR
4336	012760	104000			ERROR			;ERROR DEFINED BY GET SUB
4337	012762	000137	013452		JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4338	012766			60\$:				
4339								
4340								;VERIFY THE RESULTS OF THE SEEK COMMAND
4341	012766	004737	044714		JSR	PC,SEKSTS		;GO VERIFY RESULTS OF SEEK OPERATION
4342	012772	000405			BR	70\$;GO TO 70\$ IF NO ERROR
4343	012774	000240			NOP			;RETURN HERE IF ERROR
4344	012776	104000			ERROR			;ERROR # DEFINED BY SEKSTS SUBROUTINE
4345	013000	004736			JSR	PC,2(SP)+		;GO BACK FOR MORE ERROR CHECKS
4346	013002	000137	013452		JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4347	013006			70\$:				
4348								
4349								;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
4350	013006	012737	000063	001376	MOV	#WH!GO,RMCS10		;LOAD WRITE HEADER AND DATA

4351	013014	012702	001536	MOV	#PUTINX+3,R2	;EXTEND REGISTER INDEX TABLE
4352	013020	112722	000002	MOVW	#RMC,(R2)+	
4353	013024	112722	000004	MOVW	#RMA,(R2)+	
4354	013030	112722	000000	MOVW	#RMC1,(R2)+	
4355	013034	112722	000200	MOVW	#200,(R2)+	
4356	013040	004737	037262	JSR	PC,PUT	;GO WRITE REGISTERS VIA PUT SUB
4357	013044	000404		BR	80\$;GO TO 80\$ IF NO ERROR
4358	013046	000240		NOP		;RETURN HERE IF ERROR
4359	013050	104000		ERROR		;ERROR DEFINED BY PUT SUB
4360	013052	000137	013452	JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4361	013056					
4362						
4363						
4364	013056	004737	037622	JSR	PC,TIMOUT	;WAIT FOR COMMAND TO COMPLETE
4365	013062	004737	037012	JSR	PC,GET	;GO READ REGISTERS VIA GET SUB
4366	013066	000404		BR	90\$;GO TO 90\$ IF NO ERROR
4367	013070	000240		NOP		;RETURN HERE IF ERROR
4368	013072	104000		ERROR		;ERROR DEFINED BY GET SUB
4369	013074	000137	013452	JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4370	013100					
4371						
4372						
4373	013100	004737	040006	JSR	PC,PRIERR	;GO CHECK FOR PRIMARY ERRORS
4374	013104	000405		BR	100\$;GO TO 100\$ IF NO ERROR
4375	013106	000240		NOP		;RETURN HERE IF ERROR
4376	013110	104000		ERROR		;ERROR # DEFINED BY PRIERR SUBROUTINE
4377	013112	004736		JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
4378	013114	000137	013452	JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4379	013120					
4380	013120	004737	052312	JSR	PC,DTASTS	;GO VERIFY RESULTS OF DATA TRANSFER
4381	013124	000405		BR	110\$;GO TO 110\$ IF NO ERROR
4382	013126	000240		NOP		;RETURN HERE IF ERROR
4383	013130	104000		ERROR		;ERROR # DEFINED BY DTASTS SUBROUTINE
4384	013132	004736		JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
4385	013134	000137	013452	JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4386	013140					
4387	013140	004737	040640	JSR	PC,SECERR	;GO CHECK FOR SECONDARY ERRORS
4388	013144	000405		BR	120\$;GO TO 120\$ IF NO ERROR
4389	013146	000240		NOP		;RETURN HERE IF ERROR
4390	013150	104000		ERROR		;ERROR # DEFINED BY SECERR SUBROUTINE
4391	013152	004736		JSR	PC,@(SP)+	;GO BACK FOR MORE ERROR CHECKS
4392	013154	000137	013452	JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4393	013160					
4394						
4395						
4396	013160	005137	103576	COM	BUFTWO-2	;COMPLEMENT LAST DATA WORD
4397						
4398						
4399						
4400	013164	012737	000053	MOV	#WCH!GO,RMC10	;LOAD COMMAND
4401	013172	004737	037262	JSR	PC,PUT	;GO WRITE REGISTERS VIA PUT SUB
4402	013176	000404		BR	180\$;GO TO 180\$ IF NO ERROR
4403	013200	000240		NOP		;RETURN HERE IF ERROR
4404	013202	104000		ERROR		;ERROR DEFINED BY PUT SUB
4405	013204	000137	013452	JMP	260\$;GO TO 260\$ IF ERROR WAS FOUND
4406	013210					

4407						
4408						
4409	013210	004737	037622			
4410	013214	004737	037012			
4411	013220	000404				
4412	013222	000240				
4413	013224	104000				
4414	013226	000137	013452			
4415	013232					
4416						
4417						
4418	013232	004737	040006			
4419	013236	000405				
4420	013240	000240				
4421	013242	104000				
4422	013244	004736				
4423	013246	000137	013452			
4424	013252					
4425						


```

;WAIT FOR WRITE CHECK COMMAND TO COMPLETE AND READ STATUS
      JSR      PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
      JSR      PC,GET        ;GO READ REGISTERS VIA GET SUB
      BR       190$          ;GO TO 190$ IF NO ERROR
      NOP
      ERROR
      JMP      260$          ;GO TO 260$ IF ERROR WAS FOUND
190$:
;CHECK FOR PRIMARY ERRORS
      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      BR       200$          ;GO TO 200$ IF NO ERROR
      NOP
      ERROR
      JSR      PC,@(SP)+
      JMP      260$          ;GO BACK FOR MORE ERROR CHECKS
200$:
;GO TO 260$ IF ERROR WAS FOUND

```

```

4426 ;MAKE SURE THE WRITE CHECK ERROR WAS DETECTED
4427 013252 032737 040000 001336 BIT #WCE,RMCS2I ;IS WRITE CHECK ERROR SET??
4428 013260 001023 BNE 210$ ;YES!!
4429 013262 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4430 013266 000405 BR 205$ ;GO TO 205$ IF NO ERROR
4431 013270 000240 NOP ;RETURN HERE IF ERROR
4432 013272 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4433 013274 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4434 013276 000137 013452 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4435 013302 013737 001336 001140 205$: MOV RMCS2I,$GDDAT ;LOAD EXPECTED STATUS
4436 013310 052737 040000 001140 BIS #WCE,$GDDAT
4437 013316 013737 001336 001142 MOV RMCS2I,$BDDAT ;LOAD RECEIVED STATUS
4438 013324 104337 ERROR 337 ;WRITE CHECK ERROR NOT DETECTED
4439 013326 000451 BR 260$
4440 013330 210$:
4441 ;VERIFY THE ADDRESS OF THE WRITE CHECK ERROR
4442 MOV #BUFTWO-2,$GDADR ;LOAD EXPECTED ADDRESS
4443 013330 012737 103576 001134 MOV RMBAI,$BDDADR ;LOAD RECEIVED ADDRESS
4444 013336 013737 001332 001136 SUB #2,$BDDADR ;DECREMENT RECEIVED ADDRESS
4445 013344 162737 000002 001136
4446
4447 ;GET WCE DATA AND VERIFY IT IS OK ;SETUP FOR READING RMDB
4448 013352 112737 000022 001504 MOVB #RMDB,GETINX ;SETUP FOR READING RMDB
4449 013360 112737 000200 001505 MOVB #200,GETINX+1
4450 013366 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
4451 013372 000404 BR 230$ ;GO TO 230$ IF NO ERROR
4452 013374 000240 NOP ;RETURN HERE IF ERROR
4453 013376 104000 ERROR ;ERROR DEFINED BY GET SUB
4454 013400 000137 013452 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4455 013404 013737 001350 001142 230$: MOV RMDBI,$BDDAT ;LOAD RECEIVED DATA WORD
4456 013412 013737 103576 001140 MOV BUFTWO-2,$GDDAT ;LOAD EXPECTED DATA WORD
4457 013420 005137 001140 COM $GDDAT
4458 013424 023737 001134 001136 CMP $GDADR,$BDDADR ;IS ADDRESS OK??
4459 013432 001402 BEQ 220$ ;YES!!
4460 013434 104340 ERROR 340 ;ADDRESS OF WCE INCORRECT
4461 013436 000405 BR 260$
4462 013440 023737 001140 001142 220$: CMP $GDDAT,$BDDAT ;IS DATA WORD OK??
4463 013446 001401 BEQ 260$ ;YES!!
4464 013450 104341 ERROR 341 ;UNEXPECTED WCE DATA
4465 013452 260$:
4466
4467 ;*****
4468 ;*TEST 11 FORMAT ONES - 16
4469 ;*****
4470 TST11:
4471 013452 000004 SCOPE ;SCOPE CALL
4472 013454 000240 NOP ;START OF TEST
4473 013456 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
4474 013462 013700 001276 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
4475 013466 013701 001446 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
4476 013472 012737 000011 001226 MOV #11,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4477 013500 10$:
4478 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
4480 013500 012737 000000 001432 MOV #0,RMDCO ;CYLINDER = 0
4481 013506 012737 000000 001404 MOV #0,RMDAO ;TRACK = SECTOR = 0

```



```

4482 013514 012737 010000 001430      MOV      #FMT16,RMOFO      ;16 BIT FORMAT
4483 013522 012737 177376 001400      MOV      #(<1C<2+256.>+1),RMWCO ;2 + 256 WORDS
4484 013530 012737 102574 001402      MOV      #BUFONE,RMBAO     ;DATA BUFFER ADDRESS
4485 013536 012737 000062 001376      MOV      #WH,RMCS10       ;WRITE HEADER AND DATA
4486 013544 012737 064046 001174      MOV      #ONES,$TMP0      ;USE ALL ONES DATA PATTERN
4487 013552 012737 000001 001176      MOV      #1,$TMP1
4488 013560 004737 036114          JSR      PC,GENBUF        ;GO GENERATE DATA BUFFER
4489 013564
4490
4491
4492 013564 004737 033274          ;PREPARE DEVICE FOR DATA TRANSFER
4493 013570 154130          JSR      PC,TSTPRP       ;PREPARE DEVICE FOR TEST
4494 013572 000404          .WORD   154130 ;TASK DESCRIPTOR
4495 013574 000240          BR      30$             ;GO TO 30$ IF NO ERROR
4496 013576 104000          NOP
4497 013600 000137 014252          ERROR   ;RETURN HERE IF ERROR
4498 013604          JMP      180$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4499
4500          ;GO TO 180$ IF ERROR WAS FOUND
4501 013604 012737 000005 001376      ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
4502 013612 012702 001533          MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
4503 013616 112722 000006          MOV      #PUTINX,R2       ;WRITE REGISTER INDEX TABLE
4504 013622 112722 000034          MOV      #RMDA,(R2)+
4505 013626 112722 000032          MOV      #RMDC,(R2)+
4506 013632 112722 000000          MOV      #RMOF,(R2)+
4507 013636 112722 000200          MOV      #RMCS1,(R2)+
4508 013642 004737 037262          MOV      #200,(R2)+
4509 013646 000404          JSR      PC,PUT         ;GO WRITE REGISTERS VIA PUT SUB
4510 013650 000240          BR      40$             ;GO TO 40$ IF NO ERROR
4511 013652 104000          NOP
4512 013654 000137 014252          ERROR   ;RETURN HERE IF ERROR
4513 013660          JMP      180$          ;ERROR DEFINED BY PUT SUB
4514
4515          ;GO TO 180$ IF ERROR WAS FOUND
4516 013660 004737 036726          ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
4517 013664 004737 037622          JSR      PC,GETSTS      ;SETUP FOR STATUS
4518 013670          JSR      PC,TIMOUT     ;WAIT FOR SEEK TO COMPLETE
4519
4520          ;GO READ SEEK STATUS
4521 013670 004737 037012          JSR      PC,GET         ;GO READ REGISTERS VIA GET SUB
4522 013674 000404          BR      60$             ;GO TO 60$ IF NO ERROR
4523 013676 000240          NOP
4524 013700 104000          ERROR   ;RETURN HERE IF ERROR
4525 013702 000137 014252          JMP      180$          ;ERROR DEFINED BY GET SUB
4526 013706
4527
4528          ;GO TO 180$ IF ERROR WAS FOUND
4529 013706 004737 044714          ;VERIFY THE RESULTS OF THE SEEK COMMAND
4530 013712 000405          JSR      PC,SEKSTS     ;GO VERIFY RESULTS OF SEEK OPERATION
4531 013714 000240          BR      70$             ;GO TO 70$ IF NO ERROR
4532 013716 104000          NOP
4533 013720 004736          ERROR   ;RETURN HERE IF ERROR
4534 013722 000137 014252          JSR      PC,$(SP)+     ;ERROR # DEFINED BY SEKSTS SUBROUTINE
4535 013726          JMP      180$          ;GO BACK FOR MORE ERROR CHECKS
4536
4537          ;GO TO 180$ IF ERROR WAS FOUND
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND

```



```

4594
4595 014132 004737 037622
4596 014136 004737 037012
4597 014142 000404
4598 014144 000240
4599 014146 104000
4600 014150 000137 014252
4601 014154
4602
4603
4604 014154 004737 040006
4605 014160 000405
4606 014162 000240
4607 014164 104000
4608 014166 004736
4609 014170 000137 014252
4610 014174
4611 014174 004737 052312
4612 014200 000405
4613 014202 000240
4614 014204 104000
4615 014206 004736
4616 014210 000137 014252
4617 014214
4618 014214 004737 040640
4619 014220 000405
4620 014222 000240
4621 014224 104000
4622 014226 004736
4623 014230 000137 014252
4624 014234
4625
4626
4627 014234 004737 036360
4628 014240 102574
4629 014242 103600
4630 014244 000402
4631 014246 000240
4632 014250 104000
4633 014252
4634
4635
4636
4637
4638 014252
4639 014252 000004
4640 014254 000240
4641 014256 012706 001100
4642 014262 013700 001276
4643 014266 013701 001446
4644 014272 012737 000012 001226
4645 014300
4646
4647
4648 014300 012737 000000 001432
4649 014306 012737 000000 001404

```

```

;WAIT FOR READ TO COMPLETE AND GET STATUS
JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY GET SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
140$:
;VERIFY THE RESULTS OF READ OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
150$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160$ ;GO TO 160$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
160$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 170$ ;GO TO 170$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
170$:
;VERIFY DATA
JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
;WORD BUFOE ;STARTING ADDRESS OF WRITE BUFFER
;WORD BUFTWO ;STARTING ADDRESS OF READ BUFFER
BR 180$ ;GO TO 180$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
180$:
;*****
; *TEST 12 FORMAT CHECK ONES - 16
;*****
TST12:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #12,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = SECTOR = 0

```

```

4650 014314 012737 010000 001430 MOV #FMT16,RMOFO ;16 BIT FORMAT
4651 014322 012737 177376 001400 MOV #<↑C(2+256.)+1>,RMMCO ;2 + 256 WORDS
4652 014330 012737 102574 001402 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
4653 014336 012737 000062 001376 MOV #WH,RMCS10 ;WRITE HEADER AND DATA
4654 014344 012737 064046 001174 MOV #ONES,$TMP0 ;USE ALL ONES DATA PATTERN
4655 014352 012737 000001 001176 MOV #1,$TMP1
4656 014360 004737 036114 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
4657 014364
4658
4659
4660 014364 004737 033274 ;PREPARE DEVICE FOR DATA TRANSFER
4661 014370 154130 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
4662 014372 000404 .WORD 154130 ;TASK DESCRIPTOR
4663 014374 000240 BR 30$ ;GO TO 30$ IF NO ERROR
4664 014376 104000 NOP ;RETURN HERE IF ERROR
4665 014400 000137 015026 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4666 014404 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4667
4668
4669 014404 012737 000005 001376 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
4670 014412 012702 001533 MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
4671 014416 112722 000006 MOVB #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
4672 014422 112722 000034 MOVB #RMDA,(R2)+
4673 014426 112722 000032 MOVB #RMDC,(R2)+
4674 014432 112722 000000 MOVB #RMOF,(R2)+
4675 014436 112722 000200 MOVB #RMCS1,(R2)+
4676 014442 004737 037262 MOVB #200,(R2)+
4677 014446 000404 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
4678 014450 000240 BR 40$ ;GO TO 40$ IF NO ERROR
4679 014452 104000 NOP ;RETURN HERE IF ERROR
4680 014454 000137 015026 ERROR ;ERROR DEFINED BY PUT SUB
4681 014460 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4682
4683
4684 014460 004737 036726 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
4685 014464 004737 037622 JSR PC,GETSTS ;SETUP FOR STATUS
4686 014470 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
4687
4688
4689 014470 004737 037012 ;GO READ SEEK STATUS
4690 014474 000404 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
4691 014476 000240 BR 60$ ;GO TO 60$ IF NO ERROR
4692 014500 104000 NOP ;RETURN HERE IF ERROR
4693 014502 000137 015026 ERROR ;ERROR DEFINED BY GET SUB
4694 014506 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4695
4696
4697 014506 004737 044714 ;VERIFY THE RESULTS OF THE SEEK COMMAND
4698 014512 000405 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
4699 014514 000240 BR 70$ ;GO TO 70$ IF NO ERROR
4700 014516 104000 NOP ;RETURN HERE IF ERROR
4701 014520 004736 ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
4702 014522 000137 015026 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4703 014526 JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND
4704
4705 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND

```

```

4706 014526 012737 000063 001376      MOV      #WH!GO, RMCS10      ;LOAD WRITE HEADER AND DATA
4707 014534 012702 001536              MOV      #PUTINX+3, R2      ;EXTEND REGISTER INDEX TABLE
4708 014540 112722 000002              MOVB     #RMHC, (R2)+
4709 014544 112722 000004              MOVB     #RMBA, (R2)+
4710 014550 112722 000000              MOVB     #RMCS1, (R2)+
4711 014554 112722 000200              MOVB     #200, (R2)+
4712 014560 004737 037262              JSR      PC, PUT          ;GO WRITE REGISTERS VIA PUT SUB
4713 014564 000404              BR       80$             ;GO TO 80$ IF NO ERROR
4714 014566 000240              NOP
4715 014570 104000              ERROR   ;RETURN HERE IF ERROR
4716 014572 000137 015026              JMP      260$           ;ERROR DEFINED BY PUT SUB
4717 014576              ;GO TO 260$ IF ERROR WAS FOUND
4718
4719              ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
4720 014576 004737 037622              JSR      PC, TIMEOUT      ;WAIT FOR COMMAND TO COMPLETE
4721 014602 004737 037012              JSR      PC, GET          ;GO READ REGISTERS VIA GET SUB
4722 014606 000404              BR       90$             ;GO TO 90$ IF NO ERROR
4723 014610 000240              NOP
4724 014612 104000              ERROR   ;RETURN HERE IF ERROR
4725 014614 000137 015026              JMP      260$           ;ERROR DEFINED BY GET SUB
4726 014620              ;GO TO 260$ IF ERROR WAS FOUND
4727
4728              ;VERIFY RESULTS OF WRITE COMMAND
4729 014620 004737 040006              JSR      PC, PRIERR      ;GO CHECK FOR PRIMARY ERRORS
4730 014624 000405              BR       100$            ;GO TO 100$ IF NO ERROR
4731 014626 000240              NOP
4732 014630 104000              ERROR   ;RETURN HERE IF ERROR
4733 014632 004736              JSR      PC, J(SP)+      ;ERROR # DEFINED BY PRIERR SUBROUTINE
4734 014634 000137 015026              JMP      260$           ;GO BACK FOR MORE ERROR CHECKS
4735 014640              ;GO TO 260$ IF ERROR WAS FOUND
4736 014640 004737 052312              JSR      PC, DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
4737 014644 000405              BR       110$            ;GO TO 110$ IF NO ERROR
4738 014646 000240              NOP
4739 014650 104000              ERROR   ;RETURN HERE IF ERROR
4740 014652 004736              JSR      PC, J(SP)+      ;ERROR # DEFINED BY DTASTS SUBROUTINE
4741 014654 000137 015026              JMP      260$           ;GO BACK FOR MORE ERROR CHECKS
4742 014660              ;GO TO 260$ IF ERROR WAS FOUND
4743 014660 004737 040640              JSR      PC, SECERR      ;GO CHECK FOR SECONDARY ERRORS
4744 014664 000405              BR       120$            ;GO TO 120$ IF NO ERROR
4745 014666 000240              NOP
4746 014670 104000              ERROR   ;RETURN HERE IF ERROR
4747 014672 004736              JSR      PC, J(SP)+      ;ERROR # DEFINED BY SECERR SUBROUTINE
4748 014674 000137 015026              JMP      260$           ;GO BACK FOR MORE ERROR CHECKS
4749 014700              ;GO TO 260$ IF ERROR WAS FOUND
4750
4751              ;WRITE CHECK HEADER AND DATA FOR SECTOR JUST WRITTEN
4752              ;WRITE CHECK COMMAND
4753 014700 012737 000053 001376      MOV      #WCH!GO, RMCS10
4754 014706 004737 037262              JSR      PC, PUT          ;GO WRITE REGISTERS VIA PUT SUB
4755 014712 000404              BR       130$            ;GO TO 130$ IF NO ERROR
4756 014714 000240              NOP
4757 014716 104000              ERROR   ;RETURN HERE IF ERROR
4758 014720 000137 015026              JMP      260$           ;ERROR DEFINED BY PUT SUB
4759 014724              ;GO TO 260$ IF ERROR WAS FOUND
4760
4761              ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS

```

```

4762 014724 004737 037622      JSR    PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
4763 014730 004737 037012      JSR    PC,GET         ;GO READ REGISTERS VIA GET SUB
4764 014734 000404              BR     140$          ;GO TO 140$ IF NO ERROR
4765 014736 000240              NOP                    ;RETURN HERE IF ERROR
4766 014740 104000              ERROR                ;ERROR DEFINED BY GET SUB
4767 014742 000137 015026      JMP    260$          ;GO TO 260$ IF ERROR WAS FOUND
4768 014746
4769
4770
4771 014746 004737 040006      ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
4772 014752 000405              JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
4773 014754 000240              BR     150$          ;GO TO 150$ IF NO ERROR
4774 014756 104000              NOP                    ;RETURN HERE IF ERROR
4775 014760 004736              ERROR                ;ERROR # DEFINED BY PRIERR SUBROUTINE
4776 014762 000137 015026      JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
4777 014766              JMP    260$          ;GO TO 260$ IF ERROR WAS FOUND
4778 014766 004737 052312      150$: JSR    PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
4779 014772 000405              BR     160$          ;GO TO 160$ IF NO ERROR
4780 014774 000240              NOP                    ;RETURN HERE IF ERROR
4781 014776 104000              ERROR                ;ERROR # DEFINED BY DTASTS SUBROUTINE
4782 015000 004736              JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
4783 015002 000137 015026      JMP    260$          ;GO TO 260$ IF ERROR WAS FOUND
4784 015006
4785 015006 004737 040640      160$: JSR    PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
4786 015012 000405              BR     170$          ;GO TO 170$ IF NO ERROR
4787 015014 000240              NOP                    ;RETURN HERE IF ERROR
4788 015016 104000              ERROR                ;ERROR # DEFINED BY SECERR SUBROUTINE
4789 015020 004736              JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
4790 015022 000137 015026      JMP    260$          ;GO TO 260$ IF ERROR WAS FOUND
4791 015026
4792
4793 015026
4794
4795
4796
4797
4798 015026
4799 015026 000004
4800 015030 000240
4801 015032 012706 001100
4802 015036 013700 001276
4803 015042 013701 001446
4804 015046 012737 000013 001226
4805 015054
4806
4807
4808 015054 012737 000000 001432
4809 015062 012737 000000 001404
4810 015070 012737 010000 001430
4811 015076 012737 177376 001400
4812 015104 012737 102574 001402
4813 015112 012737 000062 001376
4814 015120 012737 064046 001174
4815 015126 012737 000001 001176
4816 015134 004737 036114
4817 015140

;*****
;#TEST 13      FORMAT CHECK ONES W/ WCE ERRORS
;*****
†ST13:
      SCOPE                ;SCOPE CALL
      NOP                  ;START OF TEST
      MOV    #STACK,SP     ;INITIALIZE STACK POINTER
      MOV    $BASE,R0      ;R0=UNIBUS ADDRESS
      MOV    TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
      MOV    #13,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
      MOV    #0,RMDCO      ;CYLINDER = 0
      MOV    #0,RMDAO      ;TRACK = SECTOR = 0
      MOV    #FMT16,RMOFO  ;16 BIT FORMAT
      MOV    #<C(2+256.)+1>,RMWCO ;2 + 256 WORDS
      MOV    #BUFONE,RMBAO ;DATA BUFFER ADDRESS
      MOV    #WH,RMCS10    ;WRITE HEADER AND DATA
      MOV    #ONES,$TMPO   ;USE ALL ONES DATA PATTERN
      MOV    #1,$TMP1
      JSR    PC,GENBUF     ;GO GENERATE DATA BUFFER

20$:

```

4818					
4819					
4820	015140	004737	033274		
4821	015144	154130			
4822	015146	000404			
4823	015150	000240			
4824	015152	104000			
4825	015154	000137	015744		
4826	015160				
4827					
4828					
4829	015160	012737	000005	001376	
4830	015166	012702	001533		
4831	015172	112722	000006		
4832	015176	112722	000034		
4833	015202	112722	000032		
4834	015206	112722	000000		
4835	015212	112722	000200		
4836	015216	004737	037262		
4837	015222	000404			
4838	015224	000240			
4839	015226	104000			
4840	015230	000137	015744		
4841	015234				
4842					
4843					
4844	015234	004737	036726		
4845	015240	004737	037622		
4846	015244				
4847					
4848					
4849	015244	004737	037012		
4850	015250	000404			
4851	015252	000240			
4852	015254	104000			
4853	015256	000137	015744		
4854	015262				
4855					
4856					
4857	015262	004737	044714		
4858	015266	000405			
4859	015270	000240			
4860	015272	104000			
4861	015274	004736			
4862	015276	000137	015744		
4863	015302				
4864					
4865					
4866	015302	012737	000063	001376	
4867	015310	012702	001536		
4868	015314	112722	000002		
4869	015320	112722	000004		
4870	015324	112722	000000		
4871	015330	112722	000200		
4872	015334	004737	037262		
4873	015340	000404			

```

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

30$:
;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

40$:
;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
JSR PC,GETSTS ;SETUP FOR STATUS
JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE

50$:
;GO READ SEEK STATUS
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 60$ ;GO TO 60$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

60$:
;VERIFY THE RESULTS OF THE SEEK COMMAND
JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
BR 70$ ;GO TO 70$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 260$ ;GO TO 260$ IF ERROR WAS FOUND

70$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
MOV #WH!GO,RMCS10 ;LOAD WRITE HEADER AND DATA
MOV #PUTINX+3,R2 ;EXTEND REGISTER INDEX TABLE
MOVB #RMWC,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 80$ ;GO TO 80$ IF NO ERROR
    
```

```

4874 015342 000240      NOP      ;RETURN HERE IF ERROR
4875 015344 104000      ERROR    ;ERROR DEFINED BY PUT SUB
4876 015346 000137 015744  JMP      260$ ;GO TO 260$ IF ERROR WAS FOUND
4877 015352
4878
4879
4880 015352 004737 037622  ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
4881 015356 004737 037012  JSR      PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
4882 015362 000404      BR      90$ ;GO READ REGISTERS VIA GET SUB
4883 015364 000240      NOP      ;GO TO 90$ IF NO ERROR
4884 015366 104000      ERROR    ;RETURN HERE IF ERROR
4885 015370 000137 015744  JMP      260$ ;ERROR DEFINED BY GET SUB
4886 015374
4887
4888
4889 015374 004737 040006  ;VERIFY RESULTS OF WRITE COMMAND
4890 015400 000405      BR      100$ ;GO CHECK FOR PRIMARY ERRORS
4891 015402 000240      NOP      ;GO TO 100$ IF NO ERROR
4892 015404 104000      ERROR    ;RETURN HERE IF ERROR
4893 015406 004736      JSR      PC,@(SP)+ ;ERROR # DEFINED BY PRIERR SUBROUTINE
4894 015410 000137 015744  JMP      260$ ;GO BACK FOR MORE ERROR CHECKS
4895 015414
4896 015414 004737 052312  ;GO VERIFY RESULTS OF DATA TRANSFER
4897 015420 000405      BR      110$ ;GO TO 110$ IF NO ERROR
4898 015422 000240      NOP      ;RETURN HERE IF ERROR
4899 015424 104000      ERROR    ;ERROR # DEFINED BY DTASTS SUBROUTINE
4900 015426 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4901 015430 000137 015744  JMP      260$ ;GO TO 260$ IF ERROR WAS FOUND
4902 015434
4903 015434 004737 040640  ;GO CHECK FOR SECONDARY ERRORS
4904 015440 000405      BR      120$ ;GO TO 120$ IF NO ERROR
4905 015442 000240      NOP      ;RETURN HERE IF ERROR
4906 015444 104000      ERROR    ;ERROR # DEFINED BY SECERR SUBROUTINE
4907 015446 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4908 015450 000137 015744  JMP      260$ ;GO TO 260$ IF ERROR WAS FOUND
4909 015454
4910
4911
4912 015454 005137 103576  ;ALTER DATA BUFFER
4913
4914
4915
4916 015460 012737 000053 001376  ;SETUP AND WRITE CHECK HEADER AND DATA COMMAND
4917 015466 004737 037262  MOV      #WCH!GO,RMCS10 ;LOAD COMMAND
4918 015472 000404      JSR      PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
4919 015474 000240      BR      180$ ;GO TO 180$ IF NO ERROR
4920 015476 104000      NOP      ;RETURN HERE IF ERROR
4921 015500 000137 015744  ERROR    ;ERROR DEFINED BY PUT SUB
4922 015504      JMP      260$ ;GO TO 260$ IF ERROR WAS FOUND
4923
4924
4925 015504 004737 037622  ;WAIT FOR WRITE CHECK COMMAND TO COMPLETE AND READ STATUS
4926 015510 004737 037012  JSR      PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
4927 015514 000404      BR      190$ ;GO READ REGISTERS VIA GET SUB
4928 015516 000240      NOP      ;GO TO 190$ IF NO ERROR
4929 015520 104000      ERROR    ;RETURN HERE IF ERROR
                     ;ERROR DEFINED BY GET SUB

```



```

4930 015522 000137 015744          JMP      260$      ;GO TO 260$ IF ERROR WAS FOUND
4931 015526
4932
4933
4934 015526 004737 040006          ;CHECK FOR PRIMARY ERRORS
4935 015532 000405          JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
4936 015534 000240          BR       200$          ;GO TO 200$ IF NO ERROR
4937 015536 104000          NOP
4938 015540 004736          ERROR   ;RETURN HERE IF ERROR
4939 015542 000137 015744          JSR      PC,(SP)+      ;ERROR # DEFINED BY PRIERR SUBROUTINE
4940 015546          JMP      260$          ;GO BACK FOR MORE ERROR CHECKS
4941
4942          ;MAKE SURE THE WRITE CHECK ERROR WAS DETECTED
4943 015546 032737 040000 001336          BIT      #WCE,RMCS2I      ;IS WRITE CHECK ERROR SET??
4944 015554 001022          BNE     210$          ;YES!!
4945 015556 004737 052312          JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
4946 015562 000405          BR       205$          ;GO TO 205$ IF NO ERROR
4947 015564 000240          NOP
4948 015566 104000          ERROR   ;RETURN HERE IF ERROR
4949 015570 004736          JSR      PC,(SP)+      ;ERROR # DEFINED BY DTASTS SUBROUTINE
4950 015572 000137 015744          JMP      260$          ;GO BACK FOR MORE ERROR CHECKS
4951 015576 013737 001336 001140          205$:  MOV     RMCS2I,$GDDAT      ;GO TO 260$ IF ERROR WAS FOUND
4952 015604 052737 040000 001140          BIS     #WCE,$GDDAT      ;LOAD EXPECTED STATUS
4953 015612 013737 001336 001142          MOV     RMCS2I,$BDDAT
4954 015620 104337          ERROR   337          ;LOAD RECEIVED STATUS
4955 015622          ;WRITE CHECK ERROR NOT DETECTED
4956
4957          ;VERIFY THE ADDRESS OF THE WRITE CHECK ERROR
4958 015622 012737 103576 001134          MOV     #BUFTWO-2,$GDADR      ;LOAD EXPECTED ADDRESS
4959 015630 013737 001332 001136          MOV     RMBAI,$BDDADR      ;LOAD RECEIVED ADDRESS
4960 015636 162737 000002 001136          SUB     #2,$BDDADR      ;DECREMENT RECEIVED ADDRESS
4961
4962          ;GET WCE DATA AND VERIFY IT IS OK
4963 015644 112737 000022 001504          MOV     #RMDB,GETINX      ;SETUP FOR READING RMDB
4964 015652 112737 000200 001505          MOV     #200,GETINX+1
4965 015660 004737 037012          JSR      PC,GET      ;GO READ REGISTERS VIA GET SUB
4966 015664 000404          BR       230$          ;GO TO 230$ IF NO ERROR
4967 015666 000240          NOP
4968 015670 104000          ERROR   ;RETURN HERE IF ERROR
4969 015672 000137 015744          JMP      260$          ;ERROR DEFINED BY GET SUB
4970 015676 013737 001350 001142          230$:  MOV     RMDBI,$BDDAT      ;GO TO 260$ IF ERROR WAS FOUND
4971 015704 013737 103576 001140          MOV     BUFTWO-2,$GDDAT      ;LOAD RECEIVED DATA WORD
4972 015712 005137 001140          COM     $GDDAT      ;LOAD EXPECTED DATA WORD
4973 015716 023737 001134 001136          CMP     $GDADR,$BDDADR      ;IS ADDRESS OK??
4974 015724 001402          BEQ     220$          ;YES!!
4975 015726 104340          ERROR   340          ;ADDRESS OF WCE INCORRECT
4976 015730 000405          BR       260$
4977 015732
4978 015732 023737 001140 001142          220$:  CMP     $GDDAT,$BDDAT      ;IS DATA WORD OK??
4979 015740 001401          BEQ     260$          ;YES!!
4980 015742 104341          ERROR   341          ;UNEXPECTED WCE DATA
4981 015744
4982
4983          ;*****
4984          ;*TEST 14          FORMAT MULTIPLE SECTORS
4985          ;*****

```

```

4986 015744
4987 015744 000004
4988 015746 000240
4989 015750 012706 001100
4990 015754 013700 001276
4991 015760 013701 001446
4992 015764 012737 000014 001226
4993 015772
4994
4995
4996 015772 012737 000000 001432
4997 016000 012737 000000 001404
4998 016006 012737 010000 001430
4999 016014 012737 176774 001400
5000 016022 012737 102574 001402
5001 016030 012737 000062 001376
5002 016036 012737 064110 001174
5003 016044 012737 000001 001176
5004 016052 004737 036114
5005 016056
5006
5007
5008 016056 004737 033274
5009 016062 154130
5010 016064 000404
5011 016066 000240
5012 016070 104000
5013 016072 000137 016520
5014 016076
5015
5016
5017 016076 012737 000005 001376
5018 016104 012702 001533
5019 016110 112722 000006
5020 016114 112722 000034
5021 016120 112722 000032
5022 016124 112722 000000
5023 016130 112722 000200
5024 016134 004737 037262
5025 016140 000404
5026 016142 000240
5027 016144 104000
5028 016146 000137 016520
5029 016152
5030
5031
5032 016152 004737 036726
5033 016156 004737 037622
5034 016162
5035
5036
5037 016162 004737 037012
5038 016166 000404
5039 016170 000240
5040 016172 104000
5041 016174 000137 016520

TST14:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK_SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #14,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDC0 ;CYLINDER = 0
MOV #0,RMDA0 ;TRACK = SECTOR = 0
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #(<C<<2+256.>*2)+1,RMWC0 ;WORD COUNT FOR 2 SECTORS
MOV #BUFONE,RMBA0 ;DATA BUFFER ADDRESS
MOV #WH,RMCS10 ;WRITE HEADER AND DATA
MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
MOV #1,$TMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20$:
;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
;WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

30$:
;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

40$:
;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
JSR PC,GETSTS ;SETUP FOR STATUS
JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE

50$:
;GO READ SEEK STATUS
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 60$ ;GO TO 60$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

```

5042	016200			60\$:	
5043					
5044					
5045	016200	004737	044714		
5046	016204	000405			
5047	016206	000240			
5048	016210	104000			
5049	016212	004736			
5050	016214	000137	016520		
5051	016220			70\$:	
5052					
5053					
5054	016220	012737	000063	001376	
5055	016226	012702	001536		
5056	016232	112722	000002		
5057	016236	112722	000004		
5058	016242	112722	000000		
5059	016246	112722	000200		
5060	016252	004737	037262		
5061	016256	000404			
5062	016260	000240			
5063	016262	104000			
5064	016264	000137	016520		
5065	016270			80\$:	
5066					
5067					
5068	016270	004737	037622		
5069	016274	004737	037012		
5070	016300	000404			
5071	016302	000240			
5072	016304	104000			
5073	016306	000137	016520		
5074	016312			90\$:	
5075					
5076					
5077	016312	004737	040006		
5078	016316	000405			
5079	016320	000240			
5080	016322	104000			
5081	016324	004736			
5082	016326	000137	016520		
5083	016332			100\$:	
5084	016332	004737	052312		
5085	016336	000405			
5086	016340	000240			
5087	016342	104000			
5088	016344	004736			
5089	016346	000137	016520		
5090	016352			110\$:	
5091	016352	004737	040640		
5092	016356	000405			
5093	016360	000240			
5094	016362	104000			
5095	016364	004736			
5096	016366	000137	016520		
5097	016372			120\$:	

```

;VERIFY THE RESULTS OF THE SEEK COMMAND
JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
BR 70$ ;GO TO 70$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

70$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
MOV #WH!GO, RMCS10 ;LOAD WRITE HEADER AND DATA
MOV #PUTINX+3, R2 ;EXTEND REGISTER INDEX TABLE
MOVB #RMC, (R2)+
MOVB #RMB, (R2)+
MOVB #RMCS1, (R2)+
MOVB #200, (R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

80$:
;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 90$ ;GO TO 90$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

90$:
;VERIFY RESULTS OF WRITE COMMAND
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 100$ ;GO TO 100$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

100$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 110$ ;GO TO 110$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

110$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 120$ ;GO TO 120$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
    
```

```

5098
5099
5100 ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
5101 016372 012737 000053 001376 MOV #WCH!GO, RMCS10 ;READ HEADER & DATA COMMAND
5102 016400 004737 037262 JSR PC, PUT ;GO WRITE REGISTERS VIA PUT SUB
5103 016404 000404 BR 130$ ;GO TO 130$ IF NO ERROR
5104 016406 000240 NOP ;RETURN HERE IF ERROR
5105 016410 104000 ERROR ;ERROR DEFINED BY PUT SUB
5106 016412 000137 016520 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5107 016416
5108
5109 ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
5110 016416 004737 037622 JSR PC, TIMEOUT ;WAIT FOR WRITE CHECK TO FINISH
5111 016422 004737 037012 JSR PC, GET ;GO READ REGISTERS VIA GET SUB
5112 016426 000404 BR 140$ ;GO TO 140$ IF NO ERROR
5113 016430 000240 NOP ;RETURN HERE IF ERROR
5114 016432 104000 ERROR ;ERROR DEFINED BY GET SUB
5115 016434 000137 016520 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5116 016440
5117
5118 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
5119 016440 004737 040006 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
5120 016444 000405 BR 150$ ;GO TO 150$ IF NO ERROR
5121 016446 000240 NOP ;RETURN HERE IF ERROR
5122 016450 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
5123 016452 004736 JSR PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5124 016454 000137 016520 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5125 016460
5126 016460 004737 052312 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
5127 016464 000405 BR 160$ ;GO TO 160$ IF NO ERROR
5128 016466 000240 NOP ;RETURN HERE IF ERROR
5129 016470 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
5130 016472 004736 JSR PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5131 016474 000137 016520 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5132 016500
5133 016500 004737 040640 JSR PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
5134 016504 000405 BR 170$ ;GO TO 170$ IF NO ERROR
5135 016506 000240 NOP ;RETURN HERE IF ERROR
5136 016510 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
5137 016512 004736 JSR PC, J(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5138 016514 000137 016520 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5139 016520
5140 016520
5141
5142 ;*****
5143 ;*TEST 15 FORMAT W/ HEAD SWITCHING
5144 ;*****
5145 TST15:
5146 016520 000004 SCOPE ;SCOPE CALL
5147 016522 000240 NOP ;START OF TEST
5148 016524 012706 001100 MOV #STACK, SP ;INITIALIZE STACK POINTER
5149 016530 013700 001276 MOV $BASE, R0 ;R0=UNIBUS ADDRESS
5150 016534 013701 001446 MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
5151 016540 012737 000015 001226 MOV #15, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
5152 016546
5153

```

```

5154 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
5155 016546 012737 000000 001432 MOV #0,RMDC0 ;CYLINDER = 0
5156 016554 012737 000037 001404 MOV #31,RMDAO ;START AT LAST SECTOR
5157 016562 012737 010000 001430 MOV #FMT16,RMOFO ;16 BIT FORMAT
5158 016570 012737 176774 001400 MOV #(<↑C<<2+256.>#2)+1,RMWC0 ;WORD COUNT FOR 2 SECTORS
5159 016576 012737 102574 001402 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
5160 016604 012737 000062 001376 MOV #MH,RMCS10 ;WRITE HEADER AND DATA
5161 016612 012737 064110 001174 MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
5162 016620 012737 000001 001176 MOV #1,$TMP1
5163 016626 004737 036114 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
5164 016632
5165
5166 ;PREPARE DEVICE FOR DATA TRANSFER
5167 016632 004737 033274 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
5168 016636 154130 .WORD 154130 ;TASK DESCRIPTOR
5169 016640 000404 BR 30$ ;GO TO 30$ IF NO ERROR
5170 016642 000240 NOP ;RETURN HERE IF ERROR
5171 016644 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5172 016646 000137 017274 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5173 016652
5174
5175 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
5176 016652 012737 000005 001376 MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
5177 016660 012702 001533 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
5178 016664 112722 000006 MOV #RMDA,(R2)+
5179 016670 112722 000034 MOV #RMDC,(R2)+
5180 016674 112722 000032 MOV #RMOF,(R2)+
5181 016700 112722 000000 MOV #RMCS1,(R2)+
5182 016704 112722 000200 MOV #200,(R2)+
5183 016710 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
5184 016714 000404 BR 40$ ;GO TO 40$ IF NO ERROR
5185 016716 000240 NOP ;RETURN HERE IF ERROR
5186 016720 104000 ERROR ;ERROR DEFINED BY PUT SUB
5187 016722 000137 017274 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5188 016726
5189
5190 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
5191 016726 004737 036726 JSR PC,GETSTS ;SETUP FOR STATUS
5192 016732 004737 037622 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
5193 016736
5194
5195 ;GO READ SEEK STATUS
5196 016736 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
5197 016742 000404 BR 60$ ;GO TO 60$ IF NO ERROR
5198 016744 000240 NOP ;RETURN HERE IF ERROR
5199 016746 104000 ERROR ;ERROR DEFINED BY GET SUB
5200 016750 000137 017274 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
5201 016754
5202
5203 ;VERIFY THE RESULTS OF THE SEEK COMMAND
5204 016754 004737 044714 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
5205 016760 000405 BR 70$ ;GO TO 70$ IF NO ERROR
5206 016762 000240 NOP ;RETURN HERE IF ERROR
5207 016764 104000 ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
5208 016766 004736 JSR PC,$(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5209 016770 000137 017274 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

```

```

5210 016774      70$:
5211
5212                ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
5213 016774 012737 000063 001376      MOV      #WH!GO, RMCS10      ;LOAD WRITE HEADER AND DATA
5214 017002 012702 001536                MOV      #PUTINX+3, R2      ;EXTEND REGISTER INDEX TABLE
5215 017006 112722 000002                MOVB    #RMC, (R2)+
5216 017012 112722 000004                MOVB    #RMA, (R2)+
5217 017016 112722 000000                MOVB    #RMS1, (R2)+
5218 017022 112722 000200                MOVB    #200, (R2)+
5219 017026 004737 037262                JSR     PC, PUT      ;GO WRITE REGISTERS VIA PUT SUB
5220 017032 000404                BR      80$          ;GO TO 80$ IF NO ERROR
5221 017034 000240                NOP
5222 017036 104000                ERROR   ;RETURN HERE IF ERROR
5223 017040 000137 017274                JMP     180$        ;ERROR DEFINED BY PUT SUB
5224 017044                ;GO TO 180$ IF ERROR WAS FOUND
5225
5226      80$:
5227                ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
5228 017044 004737 037622                JSR     PC, TIMEOUT      ;WAIT FOR COMMAND TO COMPLETE
5229 017050 004737 037012                JSR     PC, GET          ;GO READ REGISTERS VIA GET SUB
5230 017054 000404                BR      90$          ;GO TO 90$ IF NO ERROR
5231 017056 000240                NOP
5232 017060 104000                ERROR   ;RETURN HERE IF ERROR
5233 017062 000137 017274                JMP     180$        ;ERROR DEFINED BY GET SUB
5234 017066                ;GO TO 180$ IF ERROR WAS FOUND
5235
5236      90$:
5237                ;VERIFY RESULTS OF WRITE COMMAND
5238 017070 004737 040006                JSR     PC, PRIERR      ;GO CHECK FOR PRIMARY ERRORS
5239 017072 000405                BR      100$        ;GO TO 100$ IF NO ERROR
5240 017074 000240                NOP
5241 017076 104000                ERROR   ;RETURN HERE IF ERROR
5242 017100 004736                JSR     PC, DTASTS     ;ERROR # DEFINED BY PRIERR SUBROUTINE
5243 017102 000137 017274                JMP     180$        ;GO BACK FOR MORE ERROR CHECKS
5244 017106                ;GO TO 180$ IF ERROR WAS FOUND
5245      100$:
5246                ;GO VERIFY RESULTS OF DATA TRANSFER
5247 017106 004737 052312                JSR     PC, DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
5248 017112 000405                BR      110$        ;GO TO 110$ IF NO ERROR
5249 017114 000240                NOP
5250 017116 104000                ERROR   ;RETURN HERE IF ERROR
5251 017120 004736                JSR     PC, DTASTS     ;ERROR # DEFINED BY DTASTS SUBROUTINE
5252 017122 000137 017274                JMP     180$        ;GO BACK FOR MORE ERROR CHECKS
5253 017126                ;GO TO 180$ IF ERROR WAS FOUND
5254      110$:
5255                ;GO CHECK FOR SECONDARY ERRORS
5256 017126 004737 040640                JSR     PC, SECERR     ;GO CHECK FOR SECONDARY ERRORS
5257 017132 000405                BR      120$        ;GO TO 120$ IF NO ERROR
5258 017134 000240                NOP
5259 017136 104000                ERROR   ;RETURN HERE IF ERROR
5260 017140 004736                JSR     PC, SECERR     ;ERROR # DEFINED BY SECERR SUBROUTINE
5261 017142 000137 017274                JMP     180$        ;GO BACK FOR MORE ERROR CHECKS
5262 017146                ;GO TO 180$ IF ERROR WAS FOUND
5263
5264      120$:
5265                ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
5266 017146 012737 000053 001376      MOV      #WCH!GO, RMCS10      ;READ HEADER & DATA COMMAND
5267 017154 004737 037262                JSR     PC, PUT      ;GO WRITE REGISTERS VIA PUT SUB
5268 017160 000404                BR      130$        ;GO TO 130$ IF NO ERROR
5269 017162 000240                NOP
5270 017164 104000                ERROR   ;RETURN HERE IF ERROR
5271 017166 000137 017274                JMP     180$        ;ERROR DEFINED BY PUT SUB
5272                ;GO TO 180$ IF ERROR WAS FOUND

```

5266 017172
5267
5268
5269 017172 004737 037622
5270 017176 004737 037012
5271 017202 000404
5272 017204 000240
5273 017206 104000
5274 017210 000137 017274
5275 017214
5276
5277
5278 017214 004737 040006
5279 017220 000405
5280 017222 000240
5281 017224 104000
5282 017226 004736
5283 017230 000137 017274
5284 017234
5285 017234 004737 052312
5286 017240 000405
5287 017242 000240
5288 017244 104000
5289 017246 004736
5290 017250 000137 017274
5291 017254
5292 017254 004737 040640
5293 017260 000405
5294 017262 000240
5295 017264 104000
5296 017266 004736
5297 017270 000137 017274
5298 017274
5299 017274
5300
5301
5302
5303
5304 017274
5305 017274 000004
5306 017276 000240
5307 017300 012706 001100
5308 017304 013700 001276
5309 017310 013701 001446
5310 017314 012737 000016 001226
5311 017322
5312
5313
5314 017322 012737 000000 001432
5315 017330 012737 002037 001404
5316 017336 012737 010000 001430
5317 017344 012737 176774 001400
5318 017352 012737 102574 001402
5319 017360 012737 000062 001376
5320 017366 012737 064110 001174
5321 017374 012737 000001 001176

130S:
;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
JSR PC,TIMOUT ;WAIT FOR WRITE CHECK TO FINISH
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 140S ;GO TO 140S IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 180S ;GO TO 180S IF ERROR WAS FOUND
140S:
;VERIFY THE RESULTS OF WRITE CHECK OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150S ;GO TO 150S IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180S ;GO TO 180S IF ERROR WAS FOUND
150S:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160S ;GO TO 160S IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180S ;GO TO 180S IF ERROR WAS FOUND
160S:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 170S ;GO TO 170S IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180S ;GO TO 180S IF ERROR WAS FOUND
170S:
180S:

;TEST 16 FORMAT W/ MID TRANSFER SEEK

TST16:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV #BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #16,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10S:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDC0 ;CYLINDER = 0
MOV #002037,RMDA0 ;START AT LAST TRACK & SECTOR
MOV #FMT16,RMFO ;16 BIT FORMAT
MOV #(<C<<2+256.>#2)+1,RMWC0 ;WORD COUNT FOR 2 SECTORS
MOV #BUFONE,RMBA0 ;DATA BUFFER ADDRESS
MOV #MH,RMC\$10 ;WRITE HEADER AND DATA
MOV #ZEROS,\$TMP0 ;USE ALL ZEROS DATA PATTERN
MOV #1,\$TMP1

```

5322 017402 004737 036114          JSR      PC,GENBUF          ;GO GENERATE DATA BUFFER
5323 017406                                20$:
5324
5325                                ;PREPARE DEVICE FOR DATA TRANSFER
5326 017406 004737 033274          JSR      PC,TSTPRP          ;PREPARE DEVICE FOR TEST
5327 017412 154130                    .WORD   154130 ;TASK DESCRIPTOR
5328 017414 000404                    BR       30$                ;GO TO 30$ IF NO ERROR
5329 017416 000240                    NOP
5330 017420 104000                    ERROR
5331 017422 000137 020050          JMP      180$              ;RETURN HERE IF ERROR
5332 017426                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5333                                ;GO TO 180$ IF ERROR WAS FOUND
5334                                30$:
5335                                ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
5336 017426 012737 000005 001376  MOV      #SEEK!GO, RMCS10    ;CHANGE COMMAND TO SEEK
5337 017434 012702 001533          MOV      #PUTINX,R2        ;WRITE REGISTER INDEX TABLE
5338 017440 112722 000006          MOVVB   #RMDA,(R2)+
5339 017444 112722 000034          MOVVB   #RMDC,(R2)+
5340 017450 112722 000032          MOVVB   #RMOF,(R2)+
5341 017454 112722 000000          MOVVB   #RMCS1,(R2)+
5342 017460 112722 000200          MOVVB   #200,(R2)+
5343 017464 004737 037262          JSR      PC,PUT            ;GO WRITE REGISTERS VIA PUT SUB
5344 017470 000404                    BR       40$                ;GO TO 40$ IF NO ERROR
5345 017472 000240                    NOP
5346 017474 104000                    ERROR
5347 017476 000137 020050          JMP      180$              ;RETURN HERE IF ERROR
5348                                ;ERROR DEFINED BY PUT SUB
5349                                ;GO TO 180$ IF ERROR WAS FOUND
5350 017502 004737 036726          JSR      PC,GETSTS         ;SETUP FOR STATUS
5351 017506 004737 037622          JSR      PC,TIMOUT        ;WAIT FOR SEEK TO COMPLETE
5352 017512                                50$:
5353
5354                                ;GO READ SEEK STATUS
5355 017512 004737 037012          JSR      PC,GET            ;GO READ REGISTERS VIA GET SUB
5356 017516 000404                    BR       60$                ;GO TO 60$ IF NO ERROR
5357 017520 000240                    NOP
5358 017522 104000                    ERROR
5359 017524 000137 020050          JMP      180$              ;RETURN HERE IF ERROR
5360 017530                                ;ERROR DEFINED BY GET SUB
5361                                ;GO TO 180$ IF ERROR WAS FOUND
5362                                60$:
5363                                ;VERIFY THE RESULTS OF THE SEEK COMMAND
5364 017530 004737 044714          JSR      PC,SEKSTS        ;GO VERIFY RESULTS OF SEEK OPERATION
5365 017534 000405                    BR       70$                ;GO TO 70$ IF NO ERROR
5366 017536 000240                    NOP
5367 017540 104000                    ERROR
5368 017542 004736                    JSR      PC,@(SP)+         ;RETURN HERE IF ERROR
5369 017544 000137 020050          JMP      180$              ;ERROR # DEFINED BY SEKSTS SUBROUTINE
5370                                ;GO BACK FOR MORE ERROR CHECKS
5371                                ;GO TO 180$ IF ERROR WAS FOUND
5372                                70$:
5373                                ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
5374 017550 012737 000063 001376  MOV      #WH!GO, RMCS10    ;LOAD WRITE HEADER AND DATA
5375 017556 012702 001536          MOV      #PUTINX+3,R2     ;EXTEND REGISTER INDEX TABLE
5376 017562 112722 000002          MOVVB   #RMHC,(R2)+
5377 017566 112722 000004          MOVVB   #RMBB,(R2)+
5378 017572 112722 000000          MOVVB   #RMCS1,(R2)+
5379 017576 112722 000200          MOVVB   #200,(R2)+
    
```



```

5378 017602 004737 037262      JSR    PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
5379 017606 000404              BR     80$        ;GO TO 80$ IF NO ERROR
5380 017610 000240              NOP                    ;RETURN HERE IF ERROR
5381 017612 104000              ERROR          ;ERROR DEFINED BY PUT SUB
5382 017614 000137 020050      JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
5383 017620
5384
5385
5386 017620 004737 037622      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
5387 017624 004737 037012      JSR    PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
5388 017630 000404              BR     90$        ;GO READ REGISTERS VIA GET SUB
5389 017632 000240              NOP                    ;RETURN HERE IF ERROR
5390 017634 104000              ERROR          ;ERROR DEFINED BY GET SUB
5391 017636 000137 020050      JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
5392 017642
5393
5394
5395 017642 004737 040006      ;VERIFY RESULTS OF WRITE COMMAND
5396 017646 000405              JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
5397 017650 000240              BR     100$       ;GO TO 100$ IF NO ERROR
5398 017652 104000              NOP                    ;RETURN HERE IF ERROR
5399 017654 004736              ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
5400 017656 000137 020050      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
5401 017662              JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
5402 017662 004737 052312      100$: JSR    PC,DTASTS  ;GO VERIFY RESULTS OF DATA TRANSFER
5403 017666 000405              BR     110$       ;GO TO 110$ IF NO ERROR
5404 017670 000240              NOP                    ;RETURN HERE IF ERROR
5405 017672 104000              ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
5406 017674 004736              JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
5407 017676 000137 020050      JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
5408 017702
5409 017702 004737 040640      110$: JSR    PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
5410 017706 000405              BR     120$       ;GO TO 120$ IF NO ERROR
5411 017710 000240              NOP                    ;RETURN HERE IF ERROR
5412 017712 104000              ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
5413 017714 004736              JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
5414 017716 000137 020050      JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
5415 017722
5416
5417
5418
5419 017722 012737 000053 001376 ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
5420 017730 004737 037262      MOV    #WCH:GO, RMCS10 ;READ HEADER & DATA COMMAND
5421 017734 000404              JSR    PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
5422 017736 000240              BR     130$       ;GO TO 130$ IF NO ERROR
5423 017740 104000              NOP                    ;RETURN HERE IF ERROR
5424 017742 000137 020050      ERROR          ;ERROR DEFINED BY PUT SUB
5425 017746              JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
5426
5427
5428 017746 004737 037622      ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
5429 017752 004737 037012      JSR    PC,TIMOUT   ;WAIT FOR WRITE CHECK TO FINISH
5430 017756 000404              BR     140$       ;GO READ REGISTERS VIA GET SUB
5431 017760 000240              NOP                    ;RETURN HERE IF ERROR
5432 017762 104000              ERROR          ;ERROR DEFINED BY GET SUB
5433 017764 000137 020050      JMP     180$      ;GO TO 180$ IF ERROR WAS FOUND
    
```

```

5434 017770
5435
5436
5437 017770 004737 040006
5438 017774 000405
5439 017776 000240
5440 020000 104000
5441 020002 004736
5442 020004 000137 020050
5443 020010
5444 020010 004737 052312
5445 020014 000405
5446 020016 000240
5447 020020 104000
5448 020022 004736
5449 020024 000137 020050
5450 020030
5451 020030 004737 040640
5452 020034 000405
5453 020036 000240
5454 020040 104000
5455 020042 004736
5456 020044 000137 020050
5457 020050
5458 020050
5459
5460
5461
5462
5463 020050
5464 020050 000004
5465 020052 000240
5466 020054 012706 001100
5467 020060 013700 001276
5468 020064 013701 001446
5469 020070 012737 000017 001226
5470 020076
5471
5472
5473 020076 012737 000000 001432
5474 020104 012737 000000 001404
5475 020112 012737 010000 001430
5476 020120 012737 176774 001400
5477 020126 012737 102574 001402
5478 020134 012737 000062 001376
5479 020142 012737 064110 001174
5480 020150 012737 000001 001176
5481 020156 004737 036114
5482 020162
5483
5484
5485 020162 004737 033274
5486 020166 154130
5487 020170 000404
5488 020172 000240
5489 020174 104000

```

```

140$:
;VERIFY THE RESULTS OF WRITE CHECK OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

150$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160$ ;GO TO 160$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

160$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 170$ ;GO TO 170$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

170$:
180$:

*****
;TEST 17 FORMAT W/ IMPLIED SEEK
*****
TST17:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #17,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = SECTOR = 0
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #(<C<<2+256.>#2)+1,RMWC0 ;WORD COUNT FOR 2 SECTORS
MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
MOV #WH,RMCS10 ;WRITE HEADER AND DATA
MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
MOV #1,$TMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20$:
;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
;WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE

```

```

5490 020176 000137 020646          JMP      180$          ;GO TO 180$ IF ERROR WAS FOUND
5491 020202
5492
5493
5494 020202 013737 001432 020650    ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE OFF CYLINDER
5495 020210 012737 001466 001432    MOV      RMDCO,190$      ;SAVE CYLINDER ADDRESS
5496 020216 012737 000005 001376    MOV      #822,RMDCO     ;SEEK TO LAST CYLINDER
5497 020224 012702 001533          MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
5498 020230 112722 000006          MOV      #PUTINX,R2     ;WRITE REGISTER INDEX TABLE
5499 020234 112722 000034          MOVVB   #RMDA,(R2)+
5500 020240 112722 000032          MOVVB   #RMDC,(R2)+
5501 020244 112722 000000          MOVVB   #RMOF,(R2)+
5502 020250 112722 000200          MOVVB   #RMCS1,(R2)+
5503 020254 004737 037262          JSR     PC,PUT          ;GO WRITE REGISTERS VIA PUT SUB
5504 020260 000404          BR      40$           ;GO TO 40$ IF NO ERROR
5505 020262 000240          NOP
5506 020264 104000          ERROR   ;RETURN HERE IF ERROR
5507 020266 000137 020646          JMP     180$         ;ERROR DEFINED BY PUT SUB
5508 020272
5509
5510
5511 020272 004737 036726          ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
5512 020276 004737 037622          JSR     PC,GETSTS      ;SETUP FOR STATUS
5513 020302
5514
5515
5516 020302 004737 037012          ;GO READ SEEK STATUS
5517 020306 000404          JSR     PC,GET        ;GO READ REGISTERS VIA GET SUB
5518 020310 000240          BR      60$           ;GO TO 60$ IF NO ERROR
5519 020312 104000          NOP
5520 020314 000137 020646          ERROR   ;RETURN HERE IF ERROR
5521 020320
5522
5523
5524 020320 004737 044714          ;GO READ SEEK STATUS
5525 020324 000405          JSR     PC,SEKSTS     ;GO READ REGISTERS VIA GET SUB
5526 020326 000240          BR      70$           ;GO TO 70$ IF NO ERROR
5527 020330 104000          NOP
5528 020332 004736          ERROR   ;RETURN HERE IF ERROR
5529 020334 000137 020646          JMP     180$         ;ERROR # DEFINED BY SEKSTS SUBROUTINE
5530 020340
5531
5532
5533 020340 013737 020650 001432    ;GO BACK FOR MORE ERROR CHECKS
5534 020346 012737 000063 001376    JSR     PC,2(SP)+     ;GO TO 180$ IF ERROR WAS FOUND
5535 020354 012702 001536
5536 020360 112722 000002
5537 020364 112722 000004
5538 020370 112722 000000
5539 020374 112722 000200
5540 020400 004737 037262          ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
5541 020404 000404          MOV     190$,RMDCO    ;RESTORE DISK ADDRESS
5542 020406 000240          MOV     #WH!GO,RMCS10 ;LOAD WRITE HEADER AND DATA
5543 020410 104000          MOV     #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
5544 020412 000137 020646          MOVVB   #RMMC,(R2)+
5545 020416          MOVVB   #RMB A,(R2)+
          MOVVB   #RMCS1,(R2)+
          MOVVB   #200,(R2)+
          JSR     PC,PUT          ;GO WRITE REGISTERS VIA PUT SUB
          BR      80$           ;GO TO 80$ IF NO ERROR
          NOP
          ERROR   ;RETURN HERE IF ERROR
          JMP     180$         ;ERROR DEFINED BY PUT SUB
30$:
40$:
50$:
60$:
70$:
80$:

```

```

5546
5547
5548 020416 004737 037622
5549 020422 004737 037012
5550 020426 000404
5551 020430 000240
5552 020432 104000
5553 020434 000137 020646
5554 020440
5555
5556
5557 020440 004737 040006
5558 020444 000405
5559 020446 000240
5560 020450 104000
5561 020452 004736
5562 020454 000137 020646
5563 020460
5564 020460 004737 052312
5565 020464 000405
5566 020466 000240
5567 020470 104000
5568 020472 004736
5569 020474 000137 020646
5570 020500
5571 020500 004737 040640
5572 020504 000405
5573 020506 000240
5574 020510 104000
5575 020512 004736
5576 020514 000137 020646
5577 020520
5578
5579
5580
5581 020520 012737 000053 001376
5582 020526 004737 037262
5583 020532 000404
5584 020534 000240
5585 020536 104000
5586 020540 000137 020646
5587 020544
5588
5589
5590 020544 004737 037622
5591 020550 004737 037012
5592 020554 000404
5593 020556 000240
5594 020560 104000
5595 020562 000137 020646
5596 020566
5597
5598
5599 020566 004737 040006
5600 020572 000405
5601 020574 000240

;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 90$ ;GO TO 90$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
90$:

;VERIFY RESULTS OF WRITE COMMAND
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 100$ ;GO TO 100$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JMP 180$ ;GO BACK FOR MORE ERROR CHECKS
;GO TO 180$ IF ERROR WAS FOUND
100$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 110$ ;GO TO 110$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JMP 180$ ;GO BACK FOR MORE ERROR CHECKS
;GO TO 180$ IF ERROR WAS FOUND
110$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 120$ ;GO TO 120$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JMP 180$ ;GO BACK FOR MORE ERROR CHECKS
;GO TO 180$ IF ERROR WAS FOUND
120$:

;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
MOV #WCH!GO, RMCS10 ;READ HEADER & DATA COMMAND
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 130$ ;GO TO 130$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
130$:

;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
JSR PC,TIMOUT ;WAIT FOR WRITE CHECK TO FINISH
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
140$:

;VERIFY THE RESULTS OF WRITE CHECK OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR

```

```

5602 020576 104000          ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
5603 020600 004736          JSR          PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
5604 020602 000137 020646          JMP          180$        ;GO TO 180$ IF ERROR WAS FOUND
5605 020606                150$:
5606 020606 004737 052312          JSR          PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
5607 020612 000405          BR          160$        ;GO TO 160$ IF NO ERROR
5608 020614 000240          NOP          ;RETURN HERE IF ERROR
5609 020616 104000          ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
5610 020620 004736          JSR          PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
5611 020622 000137 020646          JMP          180$        ;GO TO 180$ IF ERROR WAS FOUND
5612 020626                160$:
5613 020626 004737 040640          JSR          PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
5614 020632 000405          BR          170$        ;GO TO 170$ IF NO ERROR
5615 020634 000240          NOP          ;RETURN HERE IF ERROR
5616 020636 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
5617 020640 004736          JSR          PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
5618 020642 000137 020646          JMP          180$        ;GO TO 180$ IF ERROR WAS FOUND
5619 020646                170$:
5620 020646                180$:
5621 020646 000401          BR          200$        ;TEMPORARY STORAGE
5622 020650 000000          .WORD
5623 020652                200$:
5624
5625
5626
5627
5628 020652
5629 020652 000004
5630 020654 000240
5631 020656 012706 001100
5632 020662 013700 001276
5633 020666 013701 001446
5634 020672 012737 000020 001226
5635 020700
5636
5637
5638 020700 012737 000000 001432
5639 020706 012737 000000 001404
5640 020714 012737 010000 001430
5641 020722 012737 177376 001400
5642 020730 012737 102574 001402
5643 020736 012737 000062 001376
5644 020744 012737 001404 001174
5645 020752 012737 000001 001176
5646 020760 004737 036114
5647 020764
5648
5649
5650 020764 004737 033274
5651 020770 154130
5652 020772 000404
5653 020774 000240
5654 020776 104000
5655 021000 000137 021376
5656 021004
5657

;*****
; *TEST 20          FORMAT EACH SECTOR ADDRESS
;*****
TST20:
        SCOPE          ;SCOPE CALL
        NOP          ;START OF TEST
        MOV          #STACK, SP ;INITIALIZE STACK POINTER
        MOV          $BASE, R0 ;R0=UNIBUS ADDRESS
        MOV          TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
        MOV          #20, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
        MOV          #0, RMDCO ;CYLINDER = 0
        MOV          #0, RMDAO ;TRACK = SECTOR = 0
15$:
        MOV          #FMT16, RMOFO ;16 BIT FORMAT
        MOV          #(<C(2+256.)+1>), RMWCO ;2 + 256 WORDS
        MOV          #BUFONE, RMBAO ;DATA BUFFER ADDRESS
        MOV          #WH, RMC510 ;WRITE HEADER AND DATA
        MOV          #RMDAO, $TMP0 ;USE SECTOR FOR DATA PATTERN
        MOV          #1, $TMP1
        JSR          PC, GENBUF ;GO GENERATE DATA BUFFER
20$:
;PREPARE DEVICE FOR DATA TRANSFER
        JSR          PC, TSTPRP ;PREPARE DEVICE FOR TEST
        .WORD          154130 ;TASK DESCRIPTOR
        BR          30$ ;GO TO 30$ IF NO ERROR
        NOP          ;RETURN HERE IF ERROR
        ERROR          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
        JMP          190$ ;GO TO 190$ IF ERROR WAS FOUND
30$:

```

```

5658 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
5659 021004 012737 000063 001376 MOV #RH!GO,RMCS10 ;WRITE HEADER AND DATA
5660 021012 012702 001533 MOV #PUT,INX,R2 ;WRITE REGISTER INDEX TABLE
5661 021016 112722 000006 MOVB #RMDA,(R2)+
5662 021022 112722 000034 MOVB #RMDC,(R2)+
5663 021026 112722 000032 MOVB #RMOF,(R2)+
5664 021032 112722 000002 MOVB #RMWC,(R2)+
5665 021036 112722 000004 MOVB #RMBA,(R2)+
5666 021042 112722 000000 MOVB #RMCS1,(R2)+
5667 021046 112722 000200 MOVB #200,(R2)+
5668 021052 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
5669 021056 000404 BR 80$ ;GO TO 80$ IF NO ERROR
5670 021060 000240 NOP ;RETURN HERE IF ERROR
5671 021062 104000 ERROR ;ERROR DEFINED BY PUT SUB
5672 021064 000137 021376 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
5673 021070
5674
5675 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
5676 021070 004737 036726 JSR PC,GETSTS
5677
5678 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
5679 021074 004737 037622 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
5680 021100 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
5681 021104 000404 BR 90$ ;GO TO 90$ IF NO ERROR
5682 021106 000240 NOP ;RETURN HERE IF ERROR
5683 021110 104000 ERROR ;ERROR DEFINED BY GET SUB
5684 021112 000137 021376 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
5685 021116
5686
5687 ;VERIFY RESULTS OF WRITE COMMAND
5688 021116 004737 040006 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
5689 021122 000405 BR 100$ ;GO TO 100$ IF NO ERROR
5690 021124 000240 NOP ;RETURN HERE IF ERROR
5691 021126 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
5692 021130 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5693 021132 000137 021376 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
5694 021136
5695 100$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
5696 021142 000405 BR 110$ ;GO TO 110$ IF NO ERROR
5697 021144 000240 NOP ;RETURN HERE IF ERROR
5698 021146 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
5699 021150 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5700 021152 000137 021376 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
5701 021156
5702 110$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
5703 021162 000405 BR 120$ ;GO TO 120$ IF NO ERROR
5704 021164 000240 NOP ;RETURN HERE IF ERROR
5705 021166 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
5706 021170 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5707 021172 000137 021376 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
5708 021176
5709
5710
5711 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
5712 021176 012737 000073 001376 MOV #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
5713 021204 012737 103600 001402 MOV #BUFTW0,RMBA0 ;CHANGE BUS ADDRESS
    
```

```

5714 021212 004737 037262      JSR    PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
5715 021216 000404              BR     130$      ;GO TO 130$ IF NO ERROR
5716 021220 000240              NOP                    ;RETURN HERE IF ERROR
5717 021222 104000              ERROR                ;ERROR DEFINED BY PUT SUB
5718 021224 000137 021376      JMP    190$      ;GO TO 190$ IF ERROR WAS FOUND
5719 021230
5720
5721
5722 021230 004737 037622      ;WAIT FOR READ TO COMPLETE AND GET STATUS
5723 021234 004737 037012      JSR    PC,TIMOUT    ;WAIT FOR READ TO COMPLETE
5724 021240 000404              JSR    PC,GET      ;GO READ REGISTERS VIA GET SUB
5725 021242 000240              BR     140$      ;GO TO 140$ IF NO ERROR
5726 021244 104000              NOP                    ;RETURN HERE IF ERROR
5727 021246 000137 021376      ERROR                ;ERROR DEFINED BY GET SUB
5728 021252              JMP    190$      ;GO TO 190$ IF ERROR WAS FOUND
5729
5730
5731 021252 004737 040006      ;VERIFY THE RESULTS OF READ OPERATION
5732 021256 000405              JSR    PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
5733 021260 000240              BR     150$      ;GO TO 150$ IF NO ERROR
5734 021262 104000              NOP                    ;RETURN HERE IF ERROR
5735 021264 004736              ERROR                ;ERROR # DEFINED BY PRIERR SUBROUTINE
5736 021266 000137 021376      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
5737 021272              JMP    190$      ;GO TO 190$ IF ERROR WAS FOUND
5738 021272 004737 052312      150$: JSR    PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
5739 021276 000405              BR     160$      ;GO TO 160$ IF NO ERROR
5740 021300 000240              NOP                    ;RETURN HERE IF ERROR
5741 021302 104000              ERROR                ;ERROR # DEFINED BY DTASTS SUBROUTINE
5742 021304 004736              JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
5743 021306 000137 021376      JMP    190$      ;GO TO 190$ IF ERROR WAS FOUND
5744 021312
5745 021312 004737 040640      160$: JSR    PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
5746 021316 000405              BR     170$      ;GO TO 170$ IF NO ERROR
5747 021320 000240              NOP                    ;RETURN HERE IF ERROR
5748 021322 104000              ERROR                ;ERROR # DEFINED BY SECERR SUBROUTINE
5749 021324 004736              JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
5750 021326 000137 021376      JMP    190$      ;GO TO 190$ IF ERROR WAS FOUND
5751 021332
5752
5753
5754 021332 004737 036360      ;VERIFY DATA
5755 021336 102574              JSR    PC,CMPBUF    ;GO COMPARE WRITE, READ DATA BUFFERS
5756 021340 103600              .WORD BUFOne        ;STARTING ADDRESS OF WRITE BUFFER
5757 021342 000404              .WORD BUFTwo        ;STARTING ADDRESS OF READ BUFFER
5758 021344 000240              BR     180$      ;GO TO 180$ IF NO ERROR
5759 021346 104000              NOP                    ;RETURN HERE IF ERROR
5760 021350 000137 021376      ERROR                ;ERROR # DEFINED BY CMPBUF SUBROUTINE
5761 021354              JMP    190$      ;GO TO 190$ IF ERROR WAS FOUND
5762
5763
5764 021354 062737 000001 001404      ;INCREMENT ADDRESS AND FORMAT NEXT SECTOR
5765 021362 122737 000037 001404      ADD    #1,RMDAO      ;ADVANCE SECTOR COUNT
5766 021370 103402              CMPB   #31.,RMDAO    ;DONE ALL SECTORS??
5767 021372 000137 020714      BLO   190$          ;YES!!
5768 021376              JMP    15$          ;GO DO NEXT SECTOR
5769

```

```

5770
5771
5772
5773 021376
5774 021376 000004
5775 021400 000240
5776 021402 012706 001100
5777 021406 013700 001276
5778 021412 013701 001446
5779 021416 012737 000021 001226
5780 021424
5781
5782
5783 021424 012737 000000 001432
5784 021432 012737 000000 001404
5785 021440 012737 010000 001430
5786 021446 012737 177376 001400
5787 021454 012737 102574 001402
5788 021462 012737 000062 001376
5789 021470 012737 001404 001174
5790 021476 012737 000001 001176
5791 021504 004737 036114
5792 021510
5793
5794
5795 021510 004737 033274
5796 021514 154130
5797 021516 000404
5798 021520 000240
5799 021522 104000
5800 021524 000137 022122
5801 021530
5802
5803
5804 021530 012737 000063 001376
5805 021536 012702 001533
5806 021542 112722 000006
5807 021546 112722 000034
5808 021552 112722 000032
5809 021556 112722 000002
5810 021562 112722 000004
5811 021566 112722 000000
5812 021572 112722 000200
5813 021576 004737 037262
5814 021602 000404
5815 021604 000240
5816 021606 104000
5817 021610 000137 022122
5818 021614
5819
5820
5821 021614 004737 036726
5822
5823
5824 021620 004737 037622
5825 021624 004737 037012

```

```

*****
;TEST 21 FORMAT EACH TRACK ADDRESS
*****
TST21:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #21,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = SECTOR = 0
15$: MOV #FAT16,RMOFO ;16 BIT FORMAT
MOV #(<C(2+256.)+1),RMWCO ;2 + 256 WORDS
MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
MOV #WH,RMCS10 ;WRITE HEADER AND DATA
MOV #RMDAO,$TMP0 ;USE TRACK FOR DATA PATTERN
MOV #1,$TMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20$:
;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
;WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

30$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

80$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC,GETSTS

;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB

```



```

5826 021630 000404          BR      90$      ;GO TO 90$ IF NO ERROR
5827 021632 000240          NOP
5828 021634 104000          ERROR      ;RETURN HERE IF ERROR
5829 021636 000137 022122  JMP      190$      ;ERROR DEFINED BY GET SUB
5830 021642                ;GO TO 190$ IF ERROR WAS FOUND
5831
5832
5833 021642 004737 040006    90$:
5834 021646 000405          ;VERIFY RESULTS OF WRITE COMMAND
5835 021650 000240          JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
5836 021652 104000          BR      100$      ;GO TO 100$ IF NO ERROR
5837 021654 004736          NOP
5838 021656 000137 022122  ERROR      ;RETURN HERE IF ERROR
5839 021662                ;ERROR # DEFINED BY PRIERR SUBROUTINE
5840 021662 004737 052312  JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5841 021666 000405          JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
5842 021670 000240          100$:
5843 021672 104000          JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
5844 021674 004736          BR      110$      ;GO TO 110$ IF NO ERROR
5845 021676 000137 022122  NOP
5846 021702                ;RETURN HERE IF ERROR
5847 021702 004737 040640  ERROR      ;ERROR # DEFINED BY DTASTS SUBROUTINE
5848 021706 000405          JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5849 021710 000240          JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
5850 021712 104000          110$:
5851 021714 004736          JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
5852 021716 000137 022122  BR      120$      ;GO TO 120$ IF NO ERROR
5853 021722                ;RETURN HERE IF ERROR
5854
5855
5856
5857 021722 012737 000073 001376 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
5858 021730 012737 103600 001402  MOV      #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
5859 021736 004737 037262  MOV      #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
5860 021742 000404          JSR      PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
5861 021744 000240          BR      130$      ;GO TO 130$ IF NO ERROR
5862 021746 104000          NOP
5863 021750 000137 022122  ERROR      ;RETURN HERE IF ERROR
5864 021754                ;ERROR DEFINED BY PUT SUB
5865
5866
5867 021754 004737 037622  JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
5868 021760 004737 037012  130$:
5869 021764 000404          ;WAIT FOR READ TO COMPLETE AND GET STATUS
5870 021766 000240          JSR      PC,TIMOUT ;WAIT FOR READ TO COMPLETE
5871 021770 104000          JSR      PC,GET      ;GO READ REGISTERS VIA GET SUB
5872 021772 000137 022122  BR      140$      ;GO TO 140$ IF NO ERROR
5873 021776                ;RETURN HERE IF ERROR
5874
5875
5876 021776 004737 040006  ERROR      ;ERROR DEFINED BY GET SUB
5877 022002 000405          JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
5878 022004 000240          140$:
5879 022006 104000          ;VERIFY THE RESULTS OF READ OPERATION
5880 022010 004736          JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
5881 022012 000137 022122  BR      150$      ;GO TO 150$ IF NO ERROR
                    NOP
                    ;RETURN HERE IF ERROR
                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
                    JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
                    JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND

```

```

5882 022016
5883 022016 004737 052312
5884 022022 000405
5885 022024 000240
5886 022026 104000
5887 022030 004736
5888 022032 000137 022122
5889 022036
5890 022036 004737 040640
5891 022042 000405
5892 022044 000240
5893 022046 104000
5894 022050 004736
5895 022052 000137 022122
5896 022056
5897
5898
5899 022056 004737 036360
5900 022062 102574
5901 022064 103600
5902 022066 000404
5903 022070 000240
5904 022072 104000
5905 022074 000137 022122
5906 022100
5907
5908
5909 022100 062737 000400 001404
5910 022106 022737 002000 001404
5911 022114 103402
5912 022116 000137 021440
5913 022122
5914
5915
5916
5917
5918 022122
5919 022122 000004
5920 022124 000240
5921 022126 012706 001100
5922 022132 013700 001276
5923 022136 013701 001446
5924 022142 012737 000022 001226
5925 022150
5926
5927
5928 022150 012737 000001 001432
5929 022156 012737 000000 001404
5930 022164 012737 010000 001430
5931 022172 012737 177376 001400
5932 022200 012737 102574 001402
5933 022206 012737 000062 001376
5934 022214 012737 001432 001174
5935 022222 012737 000001 001176
5936 022230 004737 036114
5937 022234

150$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160$ ;GO TO 160$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

160$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 170$ ;GO TO 170$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

170$:
;VERIFY DATA
JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
;WORD BUFO1E ;STARTING ADDRESS OF WRITE BUFFER
;WORD BUFTWO ;STARTING ADDRESS OF READ BUFFER
BR 180$ ;GO TO 180$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

180$:
;INCREMENT ADDRESS AND FORMAT NEXT TRACK
ADD #400,RMDAO ;ADVANCE TRACK COUNT
CMP #2000,RMDAO ;DONE ALL TRACKS??
BLO 190$ ;YES!!
JMP 15$ ;GO DO NEXT SECTOR

190$:
;*****
;*TEST 22 FORMAT PRIME CYLINDERS
;*****
TST22:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #22,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #1,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = SECTOR = 0
15$: MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #(<C(2+256.)+1>),RMWCO ;2 + 256 WORDS
MOV #BUFO1E,RMBAO ;DATA BUFFER ADDRESS
MOV #WH,RMCS10 ;WRITE HEADER AND DATA
MOV #RMDCO,$TMPD ;USE CYLINDER FOR DATA PATTERN
MOV #1,$TMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20$:

```

```

5938
5939
5940 022234 004737 033274
5941 022240 154130
5942 022242 000404
5943 022244 000240
5944 022246 104000
5945 022250 000137 022644
5946 022254
5947
5948
5949 022254 012737 000063 001376
5950 022262 012702 001533
5951 022266 112722 000006
5952 022272 112722 000034
5953 022276 112722 000032
5954 022302 112722 000002
5955 022306 112722 000004
5956 022312 112722 000000
5957 022316 112722 000200
5958 022322 004737 037262
5959 022326 000404
5960 022330 000240
5961 022332 104000
5962 022334 000137 022644
5963 022340
5964
5965
5966 022340 004737 036726
5967
5968
5969 022344 004737 037622
5970 022350 004737 037012
5971 022354 000404
5972 022356 000240
5973 022360 104000
5974 022362 000137 022644
5975 022366
5976
5977
5978 022366 004737 040006
5979 022372 000405
5980 022374 000240
5981 022376 104000
5982 022400 004736
5983 022402 000137 022644
5984 022406
5985 022406 004737 052312
5986 022412 000405
5987 022414 000240
5988 022416 104000
5989 022420 004736
5990 022422 000137 022644
5991 022426
5992 022426 004737 040640
5993 022432 000405

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

30$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
MOV #WH!GO, RMCS10 ;WRITE HEADER AND DATA
MOV #PUTINX, R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA, (R2)+
MOVB #RMDC, (R2)+
MOVB #RMOF, (R2)+
MOVB #RMWC, (R2)+
MOVB #RMBA, (R2)+
MOVB #RMCS1, (R2)+
MOVB #200, (R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

80$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC,GETSTS

;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 90$ ;GO TO 90$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

90$:
;VERIFY RESULTS OF WRITE COMMAND
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 100$ ;GO TO 100$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

100$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 110$ ;GO TO 110$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

110$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 120$ ;GO TO 120$ IF NO ERROR

```

```

5994 022434 000240          NOP          ;RETURN HERE IF ERROR
5995 022436 104000          ERROR       ;ERROR # DEFINED BY SECERR SUBROUTINE
5996 022440 004736          JSR        PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5997 022442 000137 022644          JMP        190$      ;GO TO 190$ IF ERROR WAS FOUND
5998 022446
5999
6000
6001          ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
6002 022446 012737 000073 001376  MOV        #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
6003 022454 012737 103600 001402  MOV        #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
6004 022462 004737 037262          JSR        PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
6005 022466 000404          BR         130$      ;GO TO 130$ IF NO ERROR
6006 022470 000240          NOP
6007 022472 104000          ERROR       ;RETURN HERE IF ERROR
6008 022474 000137 022644          JMP        190$      ;ERROR DEFINED BY PUT SUB
6009 022500
6010          130$:
6011          ;WAIT FOR READ TO COMPLETE AND GET STATUS
6012 022500 004737 037622          JSR        PC,TIMOUT  ;WAIT FOR READ TO COMPLETE
6013 022504 004737 037012          JSR        PC,GET     ;GO READ REGISTERS VIA GET SUB
6014 022510 000404          BR         140$      ;GO TO 140$ IF NO ERROR
6015 022512 000240          NOP
6016 022514 104000          ERROR       ;RETURN HERE IF ERROR
6017 022516 000137 022644          JMP        190$      ;ERROR DEFINED BY GET SUB
6018 022522
6019          140$:
6020          ;VERIFY THE RESULTS OF READ OPERATION
6021 022522 004737 040006          JSR        PC,PRIERR  ;GO CHECK FOR PRIMARY ERRORS
6022 022526 000405          BR         150$      ;GO TO 150$ IF NO ERROR
6023 022530 000240          NOP
6024 022532 104000          ERROR       ;RETURN HERE IF ERROR
6025 022534 004736          JSR        PC,2(SP)+ ;ERROR # DEFINED BY PRIERR SUBROUTINE
6026 022536 000137 022644          JMP        190$      ;GO BACK FOR MORE ERROR CHECKS
6027 022542
6028          150$:
6029 022542 004737 052312          JSR        PC,DTASTS  ;GO VERIFY RESULTS OF DATA TRANSFER
6030 022546 000405          BR         160$      ;GO TO 160$ IF NO ERROR
6031 022550 000240          NOP
6032 022552 104000          ERROR       ;RETURN HERE IF ERROR
6033 022554 004736          JSR        PC,2(SP)+ ;ERROR # DEFINED BY DTASTS SUBROUTINE
6034 022556 000137 022644          JMP        190$      ;GO BACK FOR MORE ERROR CHECKS
6035 022562
6036          160$:
6037 022562 004737 040640          JSR        PC,SECERR  ;GO CHECK FOR SECONDARY ERRORS
6038 022566 000405          BR         170$      ;GO TO 170$ IF NO ERROR
6039 022570 000240          NOP
6040 022572 104000          ERROR       ;RETURN HERE IF ERROR
6041 022574 004736          JSR        PC,2(SP)+ ;ERROR # DEFINED BY SECERR SUBROUTINE
6042 022602
6043          170$:
6044          ;VERIFY DATA
6045 022602 004737 036360          JSR        PC,CMPBUF  ;GO COMPARE WRITE, READ DATA BUFFERS
6046 022606 102574          .WORD     BUFONE     ;STARTING ADDRESS OF WRITE BUFFER
6047 022610 103600          .WORD     BUFTWO     ;STARTING ADDRESS OF READ BUFFER
6048 022612 000404          BR         180$      ;GO TO 180$ IF NO ERROR
6049 022614 000240          NOP
6050 022616 104000          ERROR       ;RETURN HERE IF ERROR
6051          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
    
```

```

6050 022620 000137 022644          JMP      190$          ;GO TO 190$ IF ERROR WAS FOUND
6051 022624
6052
6053          ;INCREMENT ADDRESS AND FORMAT NEXT PRIME CYLINDER
6054 022624 006337 001432          ASL      RMDCO          ;ADVANCE CYLINDER COUNT
6055 022630 022737 001466 001432    CMP      #822.,RMDCO    ;DONE ALL CYLINDERS??
6056 022636 103402
6057 022640 000137 022164          BLO     190$          ;YES!!
6058 022644          JMP      15$          ;GO DO NEXT CYLINDER
6059
6060          ;*****
6061          ;#TEST 23          FORMAT LAST SECTOR
6062          ;*****
6063 022644          TST23:
6064 022644 000004          SCOPE          ;SCOPE CALL
6065 022646 000240          NOP           ;START OF TEST
6066 022650 012706 001100    MOV      #STACK,SP    ;INITIALIZE STACK POINTER
6067 022654 013700 001276    MOV      $BASE,R0     ;R0=UNIBUS ADDRESS
6068 022660 013701 001446    MOV      TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
6069 022664 012737 000023 001226    MOV      #23,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
6070 022672
6071
6072          ;PREPARE DEVICE FOR DATA TRANSFER
6073 022672 004737 033274    JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
6074 022676 154130          .WORD    154130 ;TASK DESCRIPTOR
6075 022700 000404          BR       30$          ;GO TO 30$ IF NO ERROR
6076 022702 000240          NOP           ;RETURN HERE IF ERROR
6077 022704 104000          ERROR    #DEFINED BY TSTPRP SUBROUTINE
6078 022706 000137 023142    JMP      190$          ;GO TO 190$ IF ERROR WAS FOUND
6079 022712
6080
6081          ;SETUP PARAMETERS FOR READING LAST SECTOR
6082 022712 012737 001466 001432    MOV      #822.,RMDCO  ;LAST CYLINDER
6083 022720 012737 002037 001404    MOV      #2037,RMDAO  ;TRACK = 4 SECTOR = 31.
6084 022726 012737 010000 001430    MOV      #FMT16,RMOFO ;16 BIT FORMAT
6085 022734 012737 177376 001400    MOV      #(<C(2+256.)+1>),RMWCO ;2 + 256 WORDS
6086 022742 012737 103600 001402    MOV      #BUFTWO,RMBAO ;DATA BUFFER ADDRESS
6087 022750 012737 000073 001376    MOV      #RH!GO,RMCS10 ;WRITE HEADER AND DATA
6088
6089 022756 012702 001533          MOV      #PUTINX,R2   ;WRITE REGISTER INDEX TABLE
6090 022762 112722 000006    MOVB    #RMDA,(R2)+
6091 022766 112722 000034    MOVB    #RMDC,(R2)+
6092 022772 112722 000032    MOVB    #RMOF,(R2)+
6093 022776 112722 000004    MOVB    #RMBA,(R2)+
6094 023002 112722 000002    MOVB    #RMWC,(R2)+
6095 023006 112722 000000    MOVB    #RMCS1,(R2)+
6096 023012 112722 000200    MOVB    #200,(R2)+
6097
6098
6099 023016          120$:
6100
6101          ;READ HEADER AND DATA OF LAST SECTOR
6102 023016 004737 037262    JSR      PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
6103 023022 000404          BR       130$        ;GO TO 130$ IF NO ERROR
6104 023024 000240          NOP           ;RETURN HERE IF ERROR
6105 023026 104000          ERROR    #DEFINED BY PUT SUB

```

H10

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 124
T23 FORMAT LAST SECTOR

SEQ 0126

6106 023030 000137 023142
6107 023034
6108
6109
6110 023034 004737 036726
6111
6112 023040 004737 037622
6113 023044 004737 037012
6114 023050 000404
6115 023052 000240
6116 023054 104000
6117 023056 000137 023142
6118 023062
6119
6120
6121 023062 004737 040006
6122 023066 000405
6123 023070 000240
6124 023072 104000
6125 023074 004736
6126 023076 000137 023142
6127 023102
6128 023102 004737 052312
6129 023106 000405
6130 023110 000240
6131 023112 104000
6132 023114 004736
6133 023116 000137 023142
6134 023122
6135 023122 004737 040640
6136 023126 000405
6137 023130 000240
6138 023132 104000
6139 023134 004736
6140 023136 000137 023142
6141 023142
6142
6143 023142
6144
6145
6146
6147
6148 023142
6149 023142 000004
6150 023144 000240
6151 023146 012706 001100
6152 023152 013700 001276
6153 023156 013701 001446
6154 023162 012737 000024 001226
6155
6156
6157 023170 004737 033274
6158 023174 154130
6159 023176 000404
6160 023200 000240
6161 023202 104000

```

JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
130$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC,GETSTS
;WAIT FOR READ TO COMPLETE AND GET STATUS
JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
140$:
;VERIFY THE RESULTS OF READ OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
150$:
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160$ ;GO TO 160$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
160$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 170$ ;GO TO 170$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
170$:
190$:
;*****
;TEST 24 FORMAT W/ AOE ERROR
;*****
TST24:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #24,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE

```

```

6162 023204 000137 023440
6163 023210
6164
6165
6166 023210 012737 001466 001432
6167 023216 012737 002037 001404
6168 023224 012737 010000 001430
6169 023232 012737 176774 001400
6170 023240 012737 103600 001402
6171 023246 012737 000073 001376
6172 023254 012702 001533
6173 023260 112722 000006
6174 023264 112722 000034
6175 023270 112722 000032
6176 023274 112722 000002
6177 023300 112722 000004
6178 023304 112722 000000
6179 023310 112722 000200
6180 023314
6181
6182 023314
6183 023314 004737 037262
6184 023320 000404
6185 023322 000240
6186 023324 104000
6187 023326 000137 023440
6188 023332
6189
6190
6191 023332 004737 036726
6192
6193
6194 023336 004737 037622
6195 023342 004737 037012
6196 023346 000404
6197 023350 000240
6198 023352 104000
6199 023354 000137 023440
6200 023360
6201
6202
6203 023360 004737 040006
6204 023364 000405
6205 023366 000240
6206 023370 104000
6207 023372 004736
6208 023374 000137 023440
6209 023400
6210 023400 004737 052312
6211 023404 000405
6212 023406 000240
6213 023410 104000
6214 023412 004736
6215 023414 000137 023440
6216 023420
6217 023420 004737 040640
    
```

```

JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
30$:
;SETUP PARAMETERS FOR READING 2 SECTORS STARTING WITH LAST SECTOR
MOV #822, RMDCO ;START AT LAST CYLINDER
MOV #2037, RMDAO ;TRACK = 4 SECTOR = 31.
MOV #FMT16, RMOFO ;16 BIT FORMAT
MOV #(C<<2+256.>*2)+1, RMWCO ;FORMAT 2 SECTORS
MOV #BUFTWO, RMBAO ;DATA BUFFER ADDRESS
MOV #RH!GO, RMCS10 ;WRITE HEADER AND DATA
MOV #PUTINX, R2 ;WRITE REGISTER INDEX TABLE
MOV #RMDA, (R2)+
MOV #RMDC, (R2)+
MOV #RMOF, (R2)+
MOV #RMWC, (R2)+
MOV #RMBA, (R2)+
MOV #RMCS1, (R2)+
MOV #200, (R2)+
80$:
125$:
JSR PC, PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 130$ ;GO TO 130$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
130$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC, GETSTS
;WAIT FOR READ TO COMPLETE AND GET STATUS
JSR PC, TIMEOUT ;WAIT FOR READ TO COMPLETE
JSR PC, GET ;GO READ REGISTERS VIA GET SUB
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
140$:
;VERIFY THE RESULTS OF READ OPERATION
JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
150$:
JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160$ ;GO TO 160$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
160$:
JSR PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
    
```

```

6218 023424 000405 BR 170$ ;GO TO 170$ IF NO ERROR
6219 023426 000240 NOP ;RETURN HERE IF ERROR
6220 023430 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
6221 023432 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6222 023434 000137 023440 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6223 023440 170$:
6224
6225 023440 190$:
6226
6227 ;*****
6228 ;*TEST 25 FORMAT INVALID SECTOR ADDRESS
6229 ;*****
6230 TST25:
6231 023440 000004 SCOPE ;SCOPE CALL
6232 023442 000240 NOP ;START OF TEST
6233 023444 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
6234 023450 013700 001276 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
6235 023454 013701 001446 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
6236 023460 012737 000025 001226 MOV #25,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
6237 023466 10$:
6238
6239 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
6240 023466 012737 000000 001432 MOV #0,RMDCO ;CYLINDER = 0
6241 023474 012737 000036 001404 MOV #30,RMDAO ;TRACK = 0, SECTOR = 30
6242 023502 012737 000000 001430 MOV #0,RMOFO ;18 BIT FORMAT
6243 023510 012737 177376 001400 MOV #<C(2+256.)+1>,RMWCO ;2 + 256 WORDS
6244 023516 012737 102574 001402 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
6245 023524 012737 000062 001376 MOV #WH,RMCS10 ;WRITE HEADER AND DATA
6246 023532 012737 001404 001174 MOV #RMDAO,$TMPD ;USE SECTOR FOR DATA PATTERN
6247 023540 012737 000001 001176 MOV #1,$TMP1
6248 023546 004737 036114 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
6249 023552 20$:
6250
6251 ;PREPARE DEVICE FOR DATA TRANSFER
6252 023552 004737 033274 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
6253 023556 154130 .WORD 154130 ;TASK DESCRIPTOR
6254 023560 000404 BR 30$ ;GO TO 30$ IF NO ERROR
6255 023562 000240 NOP ;RETURN HERE IF ERROR
6256 023564 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6257 023566 000137 024166 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6258 023572 30$:
6259
6260 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
6261 023572 012737 000063 001376 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
6262 023600 012702 001533 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
6263 023604 112722 000006 MOVB #RMDA,(R2)+
6264 023610 112722 000034 MOVB #RMDC,(R2)+
6265 023614 112722 000032 MOVB #RMOF,(R2)+
6266 023620 112722 000002 MOVB #RMWC,(R2)+
6267 023624 112722 000004 MOVB #RMBA,(R2)+
6268 023630 112722 000000 MOVB #RMCS1,(R2)+
6269 023634 112722 000200 MOVB #200,(R2)+
6270 023640 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
6271 023644 000404 BR 80$ ;GO TO 80$ IF NO ERROR
6272 023646 000240 NOP ;RETURN HERE IF ERROR
6273 023650 104000 ERROR ;ERROR DEFINED BY PUT SUB

```


K10

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 127
T25 FORMAT INVALID SECTOR ADDRESS

SEQ 0129

6274	023652	000137	024166		JMP	190\$;GO TO 190\$ IF ERROR WAS FOUND
6275	023656			80\$:			
6276							
6277							
6278	023656	004737	036726				
6279							
6280							
6281	023662	004737	037622				
6282	023666	004737	037012				
6283	023672	000404					
6284	023674	000240					
6285	023676	104000					
6286	023700	000137	024166				
6287	023704			90\$:			
6288							
6289							
6290	023704	004737	040006				
6291	023710	000405					
6292	023712	000240					
6293	023714	104000					
6294	023716	004736					
6295	023720	000137	024166				
6296	023724			100\$:			
6297	023724	004737	052312				
6298	023730	000405					
6299	023732	000240					
6300	023734	104000					
6301	023736	004736					
6302	023740	000137	024166				
6303	023744			110\$:			
6304	023744	004737	040640				
6305	023750	000405					
6306	023752	000240					
6307	023754	104000					
6308	023756	004736					
6309	023760	000137	024166				
6310	023764			120\$:			
6311							
6312							
6313	023764	004737	033274				
6314	023770	154130					
6315	023772	000404					
6316	023774	000240					
6317	023776	104000					
6318	024000	000137	024166				
6319	024004			125\$:			
6320							
6321							
6322	024004	012737	000073	001376			
6323	024012	012737	103600	001402			
6324	024020	004737	037262				
6325	024024	000404					
6326	024026	000240					
6327	024030	104000					
6328	024032	000137	024166				
6329	024036			130\$:			

```

6330
6331 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
6332 024036 004737 036726 JSR PC,GETSTS
6333
6334 ;WAIT FOR READ TO COMPLETE AND GET STATUS
6335 024042 004737 037622 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
6336 024046 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
6337 024052 000404 BR 140$ ;GO TO 140$ IF NO ERROR
6338 024054 000240 NOP ;RETURN HERE IF ERROR
6339 024056 104000 ERROR ;ERROR DEFINED BY GET SUB
6340 024060 000137 024166 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6341 024064
6342 140$:
6343 ;VERIFY THE RESULTS OF READ OPERATION
6344 024064 004737 040006 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
6345 024070 000405 BR 150$ ;GO TO 150$ IF NO ERROR
6346 024072 000240 NOP ;RETURN HERE IF ERROR
6347 024074 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
6348 024076 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6349 024100 000137 024166 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6350 024104
6351 024104 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6352 024110 000405 BR 160$ ;GO TO 160$ IF NO ERROR
6353 024112 000240 NOP ;RETURN HERE IF ERROR
6354 024114 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
6355 024116 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6356 024120 000137 024166 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6357 024124
6358 024124 004737 040640 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
6359 024130 000405 BR 170$ ;GO TO 170$ IF NO ERROR
6360 024132 000240 NOP ;RETURN HERE IF ERROR
6361 024134 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
6362 024136 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6363 024140 000137 024166 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6364 024144
6365 024144
6366
6367 ;INCREMENT ADDRESS AND FORMAT NEXT SECTOR
6368 024144 062737 000001 001404 ADD #1,RMDAO ;ADVANCE SECTOR COUNT
6369 024152 022737 000037 001404 CMP #31,RMDAO ;DONE ALL SECTORS??
6370 024160 103402 BLO 190$ ;YES!!
6371 024162 000137 023502 JMP 15$ ;GO DO NEXT SECTOR
6372 024166
6373 190$:
6374
6375 ;*****
6376 ;*TEST 26 FORMAT INVALID TRACK ADDRESS
6377 ;*****
6378 TST26: SCOPE ;SCOPE CALL
6379 024166 000004 NOP ;START OF TEST
6380 024170 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
6381 024172 012706 001100 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
6382 024176 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
6383 024202 013701 001446 MOV #26,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
6384 024206 012737 000026 001226
6385 10$:

```

```

6386 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
6387 024214 012737 000000 001432 MOV #0,RMDCO ;CYLINDER = 0
6388 024222 012737 002400 001404 MOV #2400,RMDAO ;TRACK = 5,SECTOR = 0
6389 024230 012737 010000 001430 15$: MOV #FMT16,RMOFO ;16 BIT FORMAT
6390 024236 012737 177376 001400 MOV #(<C<2+256.>+1),RMWCO ;2 + 256 WORDS
6391 024244 012737 102574 001402 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
6392 024252 012737 000062 001376 MOV #MH,RMCS10 ;WRITE HEADER AND DATA
6393 024260 012737 001404 001174 MOV #RMDAO,$TMP0 ;USE SECTOR FOR DATA PATTERN
6394 024266 012737 000001 001176 MOV #1,$TMP1
6395 024274 004737 036114 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
6396 024300 20$:
6397
6398 ;PREPARE DEVICE FOR DATA TRANSFER
6399 024300 004737 033274 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
6400 024304 154130 .WORD 154130 ;TASK DESCRIPTOR
6401 024306 000404 BR 30$ ;GO TO 30$ IF NO ERROR
6402 024310 000240 NOP ;RETURN HERE IF ERROR
6403 024312 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6404 024314 000137 024714 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6405 024320 30$:
6406
6407 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
6408 024320 012737 000063 001376 MOV #MH!GO,RMCS10 ;WRITE HEADER AND DATA
6409 024326 012702 001533 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
6410 024332 112722 000006 MOVB #RMDA,(R2)+
6411 024336 112722 000034 MOVB #RMDC,(R2)+
6412 024342 112722 000032 MOVB #RMOF,(R2)+
6413 024346 112722 000002 MOVB #RMWC,(R2)+
6414 024352 112722 000004 MOVB #RMBA,(R2)+
6415 024356 112722 000000 MOVB #RMCS1,(R2)+
6416 024362 112722 000200 MOVB #200,(R2)+
6417 024366 004737 037262 JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
6418 024372 000404 BR 80$ ;GO TO 80$ IF NO ERROR
6419 024374 000240 NOP ;RETURN HERE IF ERROR
6420 024376 104000 ERROR ;ERROR DEFINED BY PUT SUB
6421 024400 000137 024714 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6422 024404 80$:
6423
6424 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
6425 024404 004737 036726 JSR PC,GETSTS
6426
6427 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
6428 024410 004737 037622 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
6429 024414 004737 037012 JSR PC,GET ;GO READ REGISTERS VIA GET SUB
6430 024420 000404 BR 90$ ;GO TO 90$ IF NO ERROR
6431 024422 000240 NOP ;RETURN HERE IF ERROR
6432 024424 104000 ERROR ;ERROR DEFINED BY GET SUB
6433 024426 000137 024714 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6434 024432 90$:
6435
6436 ;VERIFY RESULTS OF WRITE COMMAND
6437 024432 004737 040006 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
6438 024436 000405 BR 100$ ;GO TO 100$ IF NO ERROR
6439 024440 000240 NOP ;RETURN HERE IF ERROR
6440 024442 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
6441 024444 004736 JSR PC,$(SP)+ ;GO BACK FOR MORE ERROR CHECKS

```

```

6442 024446 000137 024714
6443 024452
6444 024452 004737 052312
6445 024456 000405
6446 024460 000240
6447 024462 104000
6448 024464 004736
6449 024466 000137 024714
6450 024472
6451 024472 004737 040640
6452 024476 000405
6453 024500 000240
6454 024502 104000
6455 024504 004736
6456 024506 000137 024714
6457 024512
6458
6459
6460 024512 004737 033274
6461 024516 154130
6462 024520 000404
6463 024522 000240
6464 024524 104000
6465 024526 000137 024714
6466 024532
6467
6468
6469 024532 012737 000073 001376
6470 024540 012737 103600 001402
6471 024546 004737 037262
6472 024552 000404
6473 024554 000240
6474 024556 104000
6475 024560 000137 024714
6476 024564
6477
6478
6479 024564 004737 036726
6480
6481
6482 024570 004737 037622
6483 024574 004737 037012
6484 024600 000404
6485 024602 000240
6486 024604 104000
6487 024606 000137 024714
6488 024612
6489
6490
6491 024612 004737 040006
6492 024616 000405
6493 024620 000240
6494 024622 104000
6495 024624 004736
6496 024626 000137 024714
6497 024632

100$: JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
      JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
      BR 110$ ;GO TO 110$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

110$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS.
      BR 120$ ;GO TO 120$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
      JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

120$: ;CLEAR "IAE" ERROR AND PREPARE DEVICE
      JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
      .WORD 154130 ;TASK DESCRIPTOR
      BR 125$ ;GO TO 125$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

125$: ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
      MOV #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
      MOV #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
      JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
      BR 130$ ;GO TO 130$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      ERROR ;ERROR DEFINED BY PUT SUB
      JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

130$: ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
      JSR PC,GETSTS

;WAIT FOR READ TO COMPLETE AND GET STATUS
      JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
      JSR PC,GET ;GO READ REGISTERS VIA GET SUB
      BR 140$ ;GO TO 140$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      ERROR ;ERROR DEFINED BY GET SUB
      JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

140$: ;VERIFY THE RESULTS OF READ OPERATION
      JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      BR 150$ ;GO TO 150$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

150$:

```

```

6498 024632 004737 052312      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
6499 024636 000405              BR     160$          ;GO TO 160$ IF NO ERROR
6500 024640 000240              NOP                    ;RETURN HERE IF ERROR
6501 024642 104000              ERROR                 ;ERROR # DEFINED BY DTASTS SUBROUTINE
6502 024644 004736              JSR    PC,2(SP)+     ;GO BACK FOR MORE ERROR CHECKS
6503 024646 000137 024714      JMP    190$          ;GO TO 190$ IF ERROR WAS FOUND
6504 024652
6505 024652 004737 040640      160$: JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
6506 024656 000405              BR     170$          ;GO TO 170$ IF NO ERROR
6507 024660 000240              NOP                    ;RETURN HERE IF ERROR
6508 024662 104000              ERROR                 ;ERROR # DEFINED BY SECERR SUBROUTINE
6509 024664 004736              JSR    PC,2(SP)+     ;GO BACK FOR MORE ERROR CHECKS
6510 024666 000137 024714      JMP    190$          ;GO TO 190$ IF ERROR WAS FOUND
6511 024672
6512 024672
6513
6514
6515 024672 062737 000400 001404 ;INCREMENT ADDRESS AND FORMAT NEXT TRACK
6516 024700 022737 003400 001404      ADD    #400,RMDAO    ;ADVANCE TRACK COUNT
6517 024706 103402              CMP    #3400,RMDAO  ;DONE ALL TRACKS??
6518 024710 000137 024230      BLO   190$          ;YES!!
6519 024714              JMP    15$           ;GO DO NEXT TRACK
6520
6521
6522
6523
6524 024714
6525 024714 000004              ;*****
6526 024716 000240              ;TEST 27      FORMAT INVALID CYLINDER ADDRESS
6527 024720 012706 001100              ;*****
6528 024724 013700 001276      TST27:
6529 024730 013701 001446              SCOPE                 ;SCOPE CALL
6530 024734 012737 000027 001226      NOP                    ;START OF TEST
6531 024742
6532
6533
6534 024742 012737 001467 001432      MOV    #823,RMDC0    ;START AT LAST CYLINDER + 1
6535 024750 012737 000000 001404      MOV    #0,RMDAO     ;TRACK = SECTOR = 0
6536 024756 012737 010000 001430      15$: MOV    #FMT16,RMOFO ;16 BIT FORMAT
6537 024764 012737 177376 001400      MOV    #(<C(2+256.)+1>),RMWCO ;2 + 256 WORDS
6538 024772 012737 102574 001402      MOV    #BUFONE,RMBAO ;DATA BUFFER ADDRESS
6539 025000 012737 000062 001376      MOV    #WH,RMCS10   ;WRITE HEADER AND DATA
6540 025006 012737 001432 001174      MOV    #RMDCO,STMP0 ;USE CYLINDER FOR DATA PATTERN
6541 025014 012737 000001 001176      MOV    #1,STMP1
6542 025022 004737 036114      JSR    PC,GENBUF    ;GO GENERATE DATA BUFFER
6543 025026
6544
6545
6546 025026 004737 033274      20$: ;PREPARE DEVICE FOR DATA TRANSFER
6547 025032 154130              JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
6548 025034 000404              .WORD 154130 ;TASK DESCRIPTOR
6549 025036 000240              BR     30$          ;GO TO 30$ IF NO ERROR
6550 025040 104000              NOP                    ;RETURN HERE IF ERROR
6551 025042 000137 025442      ERROR                 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6552 025046              JMP    190$          ;GO TO 190$ IF ERROR WAS FOUND
6553
30$:

```

```

6554 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
6555 025046 012737 000063 001376 MOV #WH!GO, RMCS10 ;WRITE HEADER AND DATA
6556 025054 012702 001533 MOV #PUTINX, R2 ;WRITE REGISTER INDEX TABLE
6557 025060 112722 000006 MOV #RMDA, (R2)+
6558 025064 112722 000034 MOV #RMDC, (R2)+
6559 025070 112722 000032 MOV #RMOF, (R2)+
6560 025074 112722 000002 MOV #RMMC, (R2)+
6561 025100 112722 000004 MOV #RMDA, (R2)+
6562 025104 112722 000000 MOV #RMCS1, (R2)+
6563 025110 112722 000200 MOV #200, (R2)+
6564 025114 004737 037262 JSR PC, PUT ;GO WRITE REGISTERS VIA PUT SUB
6565 025120 000404 BR 80$ ;GO TO 80$ IF NO ERROR
6566 025122 000240 NOP ;RETURN HERE IF ERROR
6567 025124 104000 ERROR ;ERROR DEFINED BY PUT SUB
6568 025126 000137 025442 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6569 025132
6570
6571 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
6572 025132 004737 036726 JSR PC, GETSTS
6573
6574 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
6575 025136 004737 037622 JSR PC, TIMEOUT ;WAIT FOR COMMAND TO COMPLETE
6576 025142 004737 037012 JSR PC, GET ;GO READ REGISTERS VIA GET SUB
6577 025146 000404 BR 90$ ;GO TO 90$ IF NO ERROR
6578 025150 000240 NOP ;RETURN HERE IF ERROR
6579 025152 104000 ERROR ;ERROR DEFINED BY GET SUB
6580 025154 000137 025442 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6581 025160
6582
6583 ;VERIFY RESULTS OF WRITE COMMAND
6584 025160 004737 040006 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
6585 025164 000405 BR 100$ ;GO TO 100$ IF NO ERROR
6586 025166 000240 NOP ;RETURN HERE IF ERROR
6587 025170 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
6588 025172 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
6589 025174 000137 025442 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6590 025200
6591 025200 004737 052312 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6592 025204 000405 BR 110$ ;GO TO 110$ IF NO ERROR
6593 025206 000240 NOP ;RETURN HERE IF ERROR
6594 025210 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
6595 025212 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
6596 025214 000137 025442 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6597 025220
6598 025220 004737 040640 JSR PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
6599 025224 000405 BR 120$ ;GO TO 120$ IF NO ERROR
6600 025226 000240 NOP ;RETURN HERE IF ERROR
6601 025230 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
6602 025232 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
6603 025234 000137 025442 JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
6604 025240
6605
6606 ;CLEAR "IAE" ERROR AND PREPARE DEVICE
6607 025240 004737 033274 JSR PC, TSTPRP ;PREPARE DEVICE FOR TEST
6608 025244 154130 .WORD 154130 ;TASK DESCRIPTOR
6609 025246 000404 BR 125$ ;GO TO 125$ IF NO ERROR

```

```

6610 025250 000240      NOP      ;RETURN HERE IF ERROR
6611 025252 104000      ERROR    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6612 025254 000137 025442  JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
6613 025260
6614
6615
125$:
;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
6616 025260 012737 000073 001376  MOV      #RH!GO,RMCS10      ;READ HEADER & DATA COMMAND
6617 025266 012737 103600 001402  MOV      #BUFTWO,RMBAO      ;CHANGE BUS ADDRESS
6618 025274 004737 037262      JSR      PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
6619 025300 000404      BR      130$      ;GO TO 130$ IF NO ERROR
6620 025302 000240      NOP      ;RETURN HERE IF ERROR
6621 025304 104000      ERROR    ;ERROR DEFINED BY PUT SUB
6622 025306 000137 025442  JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
6623 025312
130$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
6625
6626 025312 004737 036726  JSR      PC,GETSTS
6627
6628
;WAIT FOR READ TO COMPLETE AND GET STATUS
6629 025316 004737 037622  JSR      PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
6630 025322 004737 037012  JSR      PC,GET      ;GO READ REGISTERS VIA GET SUB
6631 025326 000404      BR      140$      ;GO TO 140$ IF NO ERROR
6632 025330 000240      NOP      ;RETURN HERE IF ERROR
6633 025332 104000      ERROR    ;ERROR DEFINED BY GET SUB
6634 025334 000137 025442  JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
6635 025340
140$:
;VERIFY THE RESULTS OF READ OPERATION
6637
6638 025340 004737 040006  JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
6639 025344 000405      BR      150$      ;GO TO 150$ IF NO ERROR
6640 025346 000240      NOP      ;RETURN HERE IF ERROR
6641 025350 104000      ERROR    ;ERROR # DEFINED BY PRIERR SUBROUTINE
6642 025352 004736      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
6643 025354 000137 025442  JMP      190$      ;GO TO 190$ IF ERROR WAS FOUND
6644 025360
150$:
;GO VERIFY RESULTS OF DATA TRANSFER
6645 025360 004737 052312  JSR      PC,DTASTS      ;GO TO 160$ IF NO ERROR
6646 025364 000405      BR      160$      ;RETURN HERE IF ERROR
6647 025366 000240      NOP      ;ERROR # DEFINED BY DTASTS SUBROUTINE
6648 025370 104000      ERROR    ;GO BACK FOR MORE ERROR CHECKS
6649 025372 004736      JSR      PC,@(SP)+      ;GO TO 190$ IF ERROR WAS FOUND
6650 025374 000137 025442  JMP      190$
160$:
;GO CHECK FOR SECONDARY ERRORS
6651 025400
6652 025400 004737 040640  JSR      PC,SECERR      ;GO TO 170$ IF NO ERROR
6653 025404 000405      BR      170$      ;RETURN HERE IF ERROR
6654 025406 000240      NOP      ;ERROR # DEFINED BY SECERR SUBROUTINE
6655 025410 104000      ERROR    ;GO BACK FOR MORE ERROR CHECKS
6656 025412 004736      JSR      PC,@(SP)+      ;GO TO 190$ IF ERROR WAS FOUND
6657 025414 000137 025442  JMP      190$
170$:
;INCREMENT ADDRESS AND FORMAT NEXT CYLINDER
6659
6660
6661 025420 062737 000001 001432  ADD      #1,RMDCO      ;ADVANCE CYLINDER COUNT
6662 025426 022737 001777 001432  CMP      #1777,RMDCO      ;DONE ALL CYLINDERS??
6663 025434 103402      BLO     190$      ;YES!!
6664 025436 000137 024756      JMP      15$      ;GO DO NEXT SECTOR
6665 025442
190$:

```

```

6666
6667
6668
6669
6670 025442
6671 025442 000004
6672 025444 000240
6673 025446 012706 001100
6674 025452 013700 001276
6675 025456 013701 001446
6676 025462 012737 000030 001226
6677 025470
6678
6679
6680 025470 012737 000000 001432
6681 025476 012737 000000 001404
6682 025504 012737 010000 001430
6683 025512 012737 177376 001400
6684 025520 012737 102574 001402
6685 025526 012737 000063 001376
6686 025534 012737 001404 001174
6687 025542 012737 000001 001176
6688 025550 004737 036114
6689 025554
6690
6691
6692 025554 004737 033274
6693 025560 154130
6694 025562 000404
6695 025564 000240
6696 025566 104000
6697 025570 000137 026334
6698 025574
6699
6700
6701 025574 012737 000015 001376
6702 025602 012702 001533
6703 025606 112722 000006
6704 025612 112722 000034
6705 025616 112722 000032
6706 025622 112722 000000
6707 025626 112712 000200
6708 025632 004737 037262
6709 025636 000404
6710 025640 000240
6711 025642 104000
6712 025644 000137 026334
6713 025650
6714
6715
6716 025650 012737 000063 001376
6717 025656 012702 001533
6718 025662 112722 000002
6719 025666 112722 000004
6720 025672 112722 000000
6721 025676 112722 000200

```

```

*****
;TEST 30 FORMAT AT OFFSET
*****
TST30:
        SCOPE                                ;SCOPE CALL
        NOP                                  ;START OF TEST
        MOV #STACK,SP                       ;INITIALIZE STACK POINTER
        MOV $BASE,R0                         ;R0=UNIBUS ADDRESS
        MOV TSTQUE,R1                       ;(R1) = DEVICE BEING TESTED
        MOV #30,$TESTN                      ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
        MOV #0,RMDCO                        ;CYLINDER = 0
        MOV #0,RMDAO                        ;TRACK = SECTOR = 0
        MOV #FAT16,RMOFO                   ;16 BIT FORMAT
        MOV #(<C(2+256.)+1>),RMWCO        ;2 + 256 WORDS
        MOV #BUFONE,RMBAO                  ;DATA BUFFER ADDRESS
        MOV #WH!GO,RMCS10                  ;WRITE HEADER AND DATA
        MOV #RMDAO,$TMP0                   ;USE SECTOR FOR DATA PATTERN
        MOV #1,$TMP1
15$:    JSR PC,GENBUF                       ;GO GENERATE DATA BUFFER
20$:

;PREPARE DEVICE FOR DATA TRANSFER
        JSR PC,TSTPRP                       ;PREPARE DEVICE FOR TEST
        .WORD 154130 ;TASK DESCRIPTOR
        BR 30$
        NOP
        ERROR
        JMP 190$

30$:
;OFFSET DEVICE FOR WRITE
        MOV #OFFSET!GO,RMCS10              ;CHANGE TO OFFSET COMMAND
        MOV #PUTINX,R2                     ;WRITE PUT INDEX TABLE
        MOVB #RMDA,(R2)+
        MOVB #RMDC,(R2)+
        MOVB #RMOF,(R2)+
        MOVB #RMCS1,(R2)+
        MOVB #200,(R2)
        JSR PC,PUT                           ;TERMINATE TABLE
        BR 35$ ;GO TO 35$ IF NO ERROR
        NOP
        ERROR
        JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

35$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND AT OFFSET
        MOV #WH!GO,RMCS10                  ;WRITE HEADER AND DATA COMMAND
        MOV #PUTINX,R2                     ;WRITE REGISTER INDEX TABLE
        MOVB #RMDC,(R2)+
        MOVB #RMDA,(R2)+
        MOVB #RMCS1,(R2)+
        MOVB #200,(R2)+

```



```

6722 025702 004737 037262      JSR    PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
6723 025706 000404              BR     80$        ;GO TO 80$ IF NO ERROR
6724 025710 000240              NOP                    ;RETURN HERE IF ERROR
6725 025712 104000              ERROR          ;ERROR DEFINED BY PUT SUB
6726 025714 000137 026334      JMP    190$       ;GO TO 190$ IF ERROR WAS FOUND
6727 025720
6728
6729
6730 025720 004737 036726      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
        JSR    PC,GETSTS
6731
6732
6733 025724 004737 037622      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
        JSR    PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
6734 025730 004737 037012      JSR    PC,GET      ;GO READ REGISTERS VIA GET SUB
6735 025734 000404              BR     90$        ;GO TO 90$ IF NO ERROR
6736 025736 000240              NOP                    ;RETURN HERE IF ERROR
6737 025740 104000              ERROR          ;ERROR DEFINED BY GET SUB
6738 025742 000137 026334      JMP    190$       ;GO TO 190$ IF ERROR WAS FOUND
6739 025746
6740
6741
6742 025746 004737 040006      ;VERIFY RESULTS OF WRITE COMMAND
        JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
6743 025752 000405              BR     100$       ;GO TO 100$ IF NO ERROR
6744 025754 000240              NOP                    ;RETURN HERE IF ERROR
6745 025756 104000              ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
6746 025760 004736              JSR    PC,2(SP)+    ;GO BACK FOR MORE ERROR CHECKS
6747 025762 000137 026334      JMP    190$       ;GO TO 190$ IF ERROR WAS FOUND
6748 025766
6749 025766 004737 052312      100$: JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
6750 025772 000405              BR     110$       ;GO TO 110$ IF NO ERROR
6751 025774 000240              NOP                    ;RETURN HERE IF ERROR
6752 025776 104000              ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
6753 026000 004736              JSR    PC,2(SP)+    ;GO BACK FOR MORE ERROR CHECKS
6754 026002 000137 026334      JMP    190$       ;GO TO 190$ IF ERROR WAS FOUND
6755 026006
6756 026006 004737 040640      110$: JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
6757 026012 000405              BR     120$       ;GO TO 120$ IF NO ERROR
6758 026014 000240              NOP                    ;RETURN HERE IF ERROR
6759 026016 104000              ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
6760 026020 004736              JSR    PC,2(SP)+    ;GO BACK FOR MORE ERROR CHECKS
6761 026022 000137 026334      JMP    190$       ;GO TO 190$ IF ERROR WAS FOUND
6762 026026
6763
6764 026026 012737 000015 001376 ;OFFSET DEVICE FOR READ
        MOV    #OFFSET!GO,RMCS10 ;CHANGE TO OFFSET COMMAND
6765 026034 012702 001533      MOV    #PUTINX,R2    ;WRITE PUT INDEX TABLE
6766 026040 112722 000006      MOVB   #RMDA,(R2)+
6767 026044 112722 000034      MOVB   #RMDC,(R2)+
6768 026050 112722 000032      MOVB   #RMOF,(R2)+
6769 026054 112722 000000      MOVB   #RMCS1,(R2)+
6770 026060 112722 000200      MOVB   #200,(R2)+
6771 026064 004737 037262      JSR    PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
6772 026070 000404              BR     125$       ;GO TO 125$ IF NO ERROR
6773 026072 000240              NOP                    ;RETURN HERE IF ERROR
6774 026074 104000              ERROR          ;ERROR DEFINED BY PUT SUB
6775 026076 000137 026334      JMP    190$       ;GO TO 190$ IF ERROR WAS FOUND
6776 026102
6777
125$:

```

```

6778
6779
6780 026102 012737 000073 001376 ;READ HEADER AND DATA AT OFFSET
6781 026110 012737 103600 001402 MOV      #RH!GO, RMCS10      ;READ HEADER & DATA COMMAND
6782 026116 012702 001533      MOV      #BUFTWO, RMBAO      ;CHANGE BUS ADDRESS
6783 026122 112722 000002      MOV      #PUTINX, R2        ;WRITE PUT INDEX TABLE
6784 026126 112722 000004      MOVB     #RMWC, (R2)+
6785 026132 112722 000000      MOVB     #RMB, (R2)+
6786 026136 112722 000200      MOVB     #RMCS1, (R2)+
6787 026142 004737 037262      JSR      PC, PUT            ;GO WRITE REGISTERS VIA PUT SUB
6788 026146 000404      BR       130$              ;GO TO 130$ IF NO ERROR
6789 026150 000240      NOP
6790 026152 104000      ERROR   ;RETURN HERE IF ERROR
6791 026154 000137 026334      JMP      190$              ;ERROR DEFINED BY PUT SUB
6792 026160
6793
6794
6795 026160 004737 037622      ;WAIT FOR READ TO COMPLETE AND GET STATUS
6796 026164 004737 037012      JSR      PC, TIMEOUT        ;WAIT FOR READ TO COMPLETE
6797 026170 000404      JSR      PC, GET           ;GO READ REGISTERS VIA GET SUB
6798 026172 000240      BR       140$              ;GO TO 140$ IF NO ERROR
6799 026174 104000      NOP                        ;RETURN HERE IF ERROR
6800 026176 000137 026334      ERROR   ;ERROR DEFINED BY GET SUB
6801 026202      JMP      190$              ;GO TO 190$ IF ERROR WAS FOUND
6802
6803
6804 026202 004737 040006      ;VERIFY THE RESULTS OF READ OPERATION
6805 026206 000405      JSR      PC, PRIERR        ;GO CHECK FOR PRIMARY ERRORS
6806 026210 000240      BR       150$              ;GO TO 150$ IF NO ERROR
6807 026212 104000      NOP                        ;RETURN HERE IF ERROR
6808 026214 004736      ERROR   ;ERROR # DEFINED BY PRIERR SUBROUTINE
6809 026216 000137 026334      JSR      PC, J(SP)+        ;GO BACK FOR MORE ERROR CHECKS
6810 026222      JMP      190$              ;GO TO 190$ IF ERROR WAS FOUND
6811 026222 004737 052312      150$: JSR      PC, DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
6812 026226 000405      BR       160$              ;GO TO 160$ IF NO ERROR
6813 026230 000240      NOP                        ;RETURN HERE IF ERROR
6814 026232 104000      ERROR   ;ERROR # DEFINED BY DTASTS SUBROUTINE
6815 026234 004736      JSR      PC, J(SP)+        ;GO BACK FOR MORE ERROR CHECKS
6816 026236 000137 026334      JMP      190$              ;GO TO 190$ IF ERROR WAS FOUND
6817 026242
6818 026242 004737 040640      160$: JSR      PC, SECERR        ;GO CHECK FOR SECONDARY ERRORS
6819 026246 000405      BR       170$              ;GO TO 170$ IF NO ERROR
6820 026250 000240      NOP                        ;RETURN HERE IF ERROR
6821 026252 104000      ERROR   ;ERROR # DEFINED BY SECERR SUBROUTINE
6822 026254 004736      JSR      PC, J(SP)+        ;GO BACK FOR MORE ERROR CHECKS
6823 026256 000137 026334      JMP      190$              ;GO TO 190$ IF ERROR WAS FOUND
6824 026262
6825
6826
6827 026262 004737 036360      ;VERIFY DATA
6828 026266 102574      JSR      PC, CMPBUF        ;GO COMPARE WRITE, READ DATA BUFFERS
6829 026270 103600      .WORD   BUFTWO            ;STARTING ADDRESS OF WRITE BUFFER
6830 026272 000404      .WORD   BUFTWO            ;STARTING ADDRESS OF READ BUFFER
6831 026274 000240      BR       180$              ;GO TO 180$ IF NO ERROR
6832 026276 104000      NOP                        ;RETURN HERE IF ERROR
6833 026300 000137 026334      ERROR   ;ERROR # DEFINED BY CMPBUF SUBROUTINE
        JMP      190$              ;GO TO 190$ IF ERROR WAS FOUND

```

```

180$:
6834 026304
6835 026304 032737 000200 001430 BIT #OFD,RMOFO ;DONE BOTH OFFSETS??
6836 026312 001010 BNE 190$ ;YES!!
6837 026314 052737 000200 001430 BIS #OFD,RMOFO ;SET FORWARD OFFSET
6838 026322 012737 102574 001402 MOV #BUFONE,RMBAO ;CHANGE DATA BUFFER
6839 026330 000137 025554 JMP 20$
6840 026334
6841
6842
6843
6844
6845 026334
6846 026334 000004
6847 026336 000240
6848 026340 012706 001100
6849 026344 013700 001276
6850 026350 013701 001446
6851 026354 012737 000031 001226
6852 026362
6853
6854
6855 026362 012737 000000 001432
6856 026370 012737 000000 001404
6857 026376 012737 010000 001430
6858 026404 012737 177376 001400
6859 026412 012737 102574 001402
6860 026420 012737 000062 001376
6861 026426 012737 001404 001174
6862 026434 012737 000001 001176
6863 026442 004737 036114
6864 026446
6865
6866
6867 026446 004737 033274
6868 026452 154130
6869 026454 000404
6870 026456 000240
6871 026460 104000
6872 026462 000137 027142
6873 026466
6874
6875
6876 026466 012737 000001 001422
6877 026474 112737 000024 001533
6878 026502 112737 000200 001534
6879 026510 004737 037262
6880 026514 000404
6881 026516 000240
6882 026520 104000
6883 026522 000137 027142
6884 026526
6885
6886
6887 026526 012737 000063 001376
6888 026534 012737 000000 001422
6889 026542 012702 001533

```

```

;*****
;*TEST 31 IVC FORMAT TEST
;*****
TST31:
        SCOPE ;SCOPE CALL
        NOP ;START OF TEST
        MOV #STACK,SP ;INITIALIZE STACK POINTER
        MOV $BASE,R0 ;R0=UNIBUS ADDRESS
        MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
        MOV #31,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
        MOV #0,RMDCO ;CYLINDER = 0
        MOV #0,RMDAO ;TRACK = SECTOR = 0
        MOV #FMT16,RMOFO ;16 BIT FORMAT
        MOV #(<1C<2+256.>+1),RMWCO ;2 + 256 WORDS
        MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
        MOV #WH,RMCS10 ;WRITE HEADER AND DATA
        MOV #RMDAO,$TMPO ;USE SECTOR FOR DATA PATTERN
        MOV #1,$TMP1
15$: JSR PC,GENBUF ;GO GENERATE DATA BUFFER
20$:

;PREPARE DEVICE FOR DATA TRANSFER
        JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
        .WORD 154130 ;TASK DESCRIPTOR
        BR 30$ ;GO TO 30$ IF NO ERROR
        NOP ;RETURN HERE IF ERROR
        ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
        JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

30$:
;RESET VOLUME VALID BY SETTING AND RESETTING DIAGNOSTIC MODE
        MOV #DMD,RMMR10 ;SET DIAGNOSTIC MODE
        MOVB #RMMR1,PUTINX ;WRITE REGISTER INDEX TABLE
        MOVB #200,PUTINX+1
        JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
        BR 35$ ;GO TO 35$ IF NO ERROR
        NOP ;RETURN HERE IF ERROR
        ERROR ;ERROR DEFINED BY PUT SUB
        JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND

35$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
        MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
        MOV #0,RMMR10 ;RESET DIAGNOSTIC MODE
        MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE

```

6890	026546	112722	000024	MOV	#RMR1,(R2)+	
6891	026552	112722	000006	MOV	#RMDA,(R2)+	
6892	026556	112722	000034	MOV	#RMDC,(R2)+	
6893	026562	112722	000032	MOV	#RMOF,(R2)+	
6894	026566	112722	000002	MOV	#RMNC,(R2)+	
6895	026572	112722	000004	MOV	#RMB A,(R2)+	
6896	026576	112722	000000	MOV	#RMCS1,(R2)+	
6897	026602	112722	000200	MOV	#200,(R2)+	
6898	026606	004737	037262	JSR	PC,PUT	;GO WRITE REGISTERS VIA PUT SUB
6899	026612	000404		BR	80\$;GO TO 80\$ IF NO ERROR
6900	026614	000240		NOP		;RETURN HERE IF ERROR
6901	026616	104000		ERROR		;ERROR DEFINED BY PUT SUB
6902	026620	000137	027142	JMP	190\$;GO TO 190\$ IF ERROR WAS FOUND
6903	026624					
6904						
6905						
6906	026624	004737	036726	JSR	PC,GETSTS	;SETUP INPUT REGISTER BUFFER FOR READING STATUS
6907						
6908						
6909	026630	004737	037622	JSR	PC,TIMOUT	;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
6910	026634	004737	037012	JSR	PC,GET	;WAIT FOR COMMAND TO COMPLETE
6911	026640	000404		BR	90\$;GO TO 90\$ IF NO ERROR
6912	026642	000240		NOP		;RETURN HERE IF ERROR
6913	026644	104000		ERROR		;ERROR DEFINED BY GET SUB
6914	026646	000137	027142	JMP	190\$;GO TO 190\$ IF ERROR WAS FOUND
6915	026652					
6916						
6917						
6918	026652	004737	040006	JSR	PC,PRIERR	;VERIFY RESULTS OF WRITE COMMAND
6919	026656	000405		BR	100\$;GO CHECK FOR PRIMARY ERRORS
6920	026660	000240		NOP		;GO TO 100\$ IF NO ERROR
6921	026662	104000		ERROR		;RETURN HERE IF ERROR
6922	026664	004736		JSR	PC,2(SP)+	;ERROR # DEFINED BY PRIERR SUBROUTINE
6923	026666	000137	027142	JMP	190\$;GO BACK FOR MORE ERROR CHECKS
6924	026672					;GO TO 190\$ IF ERROR WAS FOUND
6925	026672	032737	010000	BIT	#IVC,RMER2I	;IS IVC STATUS SET??
6926	026700	001012		BNE	110\$;YES!!
6927	026702	013737	001370	MOV	RMER2I,\$BDDAT	;RECEIVED STATUS FOR TYPEOUT
6928	026710	013737	001370	MOV	RMER2I,\$GDDAT	;EXPECTED STATUS
6929	026716	052737	010000	BIS	#IVC,\$GDDAT	
6930	026724	104342		ERROR	342	;IVC NOT SET DURING FORMAT
6931	026726					
6932	026726	004737	040640	JSR	PC,SECERR	;GO CHECK FOR SECONDARY ERRORS
6933	026732	000405		BR	120\$;GO TO 120\$ IF NO ERROR
6934	026734	000240		NOP		;RETURN HERE IF ERROR
6935	026736	104000		ERROR		;ERROR # DEFINED BY SECERR SUBROUTINE
6936	026740	004736		JSR	PC,2(SP)+	;GO BACK FOR MORE ERROR CHECKS
6937	026742	000137	027142	JMP	190\$;GO TO 190\$ IF ERROR WAS FOUND
6938	026746					
6939						
6940						
6941	026746	004737	033274	JSR	PC,TSTPRP	;CLEAR IVC ERROR
6942	026752	040100		.WORD	040100	;PREPARE DEVICE FOR TEST
6943	026754	000404		BR	125\$;TASK DESCRIPTOR
6944	026756	000240		NOP		;GO TO 125\$ IF NO ERROR
6945	026760	104000		ERROR		;RETURN HERE IF ERROR
						;ERROR # DEFINED BY TSTPRP SUBROUTINE

```

6946 026762 000137 027142
6947 026766
6948
6949
6950 026766 012737 000073 001376
6951 026774 012737 103600 001402
6952 027002 004737 037262
6953 027006 000404
6954 027010 000240
6955 027012 104000
6956 027014 000137 027142
6957 027020
6958
6959
6960 027020 004737 036726
6961
6962
6963 027024 004737 037622
6964 027030 004737 037012
6965 027034 000404
6966 027036 000240
6967 027040 104000
6968 027042 000137 027142
6969 027046
6970
6971
6972 027046 004737 040006
6973 027052 000405
6974 027054 000240
6975 027056 104000
6976 027060 004736
6977 027062 000137 027142
6978 027066
6979 027066 032737 010000 001370
6980 027074 001012
6981 027076 013737 001370 001142
6982 027104 013737 001370 001140
6983 027112 052737 010000 001140
6984 027120 104342
6985 027122
6986 027122 004737 040640
6987 027126 000405
6988 027130 000240
6989 027132 104000
6990 027134 004736
6991 027136 000137 027142
6992 027142
6993 027142
6994
6995
6996
6997
6998 027142
6999 027142 000004
7000 027144 000240
7001 027146 012706 001100

```

```

JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
125$:
;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
MOV #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
MOV #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 130$ ;GO TO 130$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
130$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC,GETSTS
;WAIT FOR READ TO COMPLETE AND GET STATUS
JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
140$:
;VERIFY THE RESULTS OF READ OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
150$:
BIT #IVC,RMER2I ;IS IVC STATUS SET??
BNE 160$ ;YES!!
MOV RMER2I,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
MOV RMER2I,$GDDAT ;EXPECTED STATUS
BIS #IVC,$GDDAT
ERROR 342 ;IVC NOT SET DURING FORMAT
160$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 170$ ;GO TO 170$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 190$ ;GO TO 190$ IF ERROR WAS FOUND
170$:
190$:
;*****
;#TEST 32 FORMAT ERROR TEST - 18
;*****
†ST32:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER

```

```

7002 027152 013700 001276      MOV      $BASE,R0      ;RO=UNIBUS ADDRESS
7003 027156 013701 001446      MOV      TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
7004 027162 012737 000032 001226  MOV      #32,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
7005 027170
7006
7007 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
7008 027170 012737 000000 001432  MOV      #0,RMDCO      ;CYLINDER = 0
7009 027176 012737 000000 001404  MOV      #0,RMDAO      ;TRACK = SECTOR = 0
7010 027204 012737 000000 001430  MOV      #0,RMOFO      ;18 BIT FORMAT
7011 027212 012737 177376 001400  MOV      #(<C<2+256.>+1),RMWCO ;2 + 256 WORDS
7012 027220 012737 102574 001402  MOV      #BUFONE,RMBAO ;DATA BUFFER ADDRESS
7013 027226 012737 000062 001376  MOV      #WH,RMCS10    ;WRITE HEADER AND DATA
7014 027234 012737 064110 001174  MOV      #ZEROS,$TMP0  ;USE ALL ZEROS DATA PATTERN
7015 027242 012737 000001 001176  MOV      #1,$TMP1
7016 027250 004737 036114      JSR      PC,GENBUF     ;GO GENERATE DATA BUFFER
7017 027254
7018
7019 ;PREPARE DEVICE FOR DATA TRANSFER
7020 027254 004737 033274      JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
7021 027260 154130      .WORD   154130 ;TASK DESCRIPTOR
7022 027262 000404      BR      30$          ;GO TO 30$ IF NO ERROR
7023 027264 000240      NOP
7024 027266 104000      ERROR
7025 027270 000137 027664      JMP      180$        ;RETURN HERE IF ERROR
7026 027274
7027 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7028 ;GO TO 180$ IF ERROR WAS FOUND
7029 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
7029 027274 012737 000063 001376  MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
7030 027302 012702 001533      MOV      #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
7031 027306 112722 000006      MOV      #RMDA,(R2)+
7032 027312 112722 000034      MOV      #RMDC,(R2)+
7033 027316 112722 000032      MOV      #RMOF,(R2)+
7034 027322 112722 000002      MOV      #RMWC,(R2)+
7035 027326 112722 000004      MOV      #RMBA,(R2)+
7036 027332 112722 000000      MOV      #RMCS1,(R2)+
7037 027336 112722 000200      MOV      #200,(R2)+
7038 027342 004737 037262      JSR      PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
7039 027346 000404      BR      80$          ;GO TO 80$ IF NO ERROR
7040 027350 000240      NOP
7041 027352 104000      ERROR
7042 027354 000137 027664      JMP      180$        ;RETURN HERE IF ERROR
7043 027360
7044 ;ERROR DEFINED BY PUT SUB
7045 ;GO TO 180$ IF ERROR WAS FOUND
7046 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
7046 027360 004737 036726      JSR      PC,GETSTS
7047
7048 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
7049 027364 004737 037622      JSR      PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
7050 027370 004737 037012      JSR      PC,GET      ;GO READ REGISTERS VIA GET SUB
7051 027374 000404      BR      90$          ;GO TO 90$ IF NO ERROR
7052 027376 000240      NOP
7053 027400 104000      ERROR
7054 027402 000137 027664      JMP      180$        ;RETURN HERE IF ERROR
7055 027406
7056 ;ERROR DEFINED BY GET SUB
7057 ;GO TO 180$ IF ERROR WAS FOUND
;VERIFY RESULTS OF WRITE COMMAND
    
```

```

7058 027406 004737 040006      JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
7059 027412 000405              BR    100$          ;GO TO 100$ IF NO ERROR
7060 027414 000240              NOP                    ;RETURN HERE IF ERROR
7061 027416 104000              ERROR                 ;ERROR # DEFINED BY PRIERR SUBROUTINE
7062 027420 004736              JSR   PC,(SP)+       ;GO BACK FOR MORE ERROR CHECKS
7063 027422 000137 027664      JMP   180$          ;GO TO 180$ IF ERROR WAS FOUND
7064 027426
7065 027426 004737 052312      100$: JSR   PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
7066 027432 000405              BR    110$          ;GO TO 110$ IF NO ERROR
7067 027434 000240              NOP                    ;RETURN HERE IF ERROR
7068 027436 104000              ERROR                 ;ERROR # DEFINED BY DTASTS SUBROUTINE
7069 027440 004736              JSR   PC,(SP)+       ;GO BACK FOR MORE ERROR CHECKS
7070 027442 000137 027664      JMP   180$          ;GO TO 180$ IF ERROR WAS FOUND
7071 027446
7072 027446 004737 040640      110$: JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
7073 027452 000405              BR    120$          ;GO TO 120$ IF NO ERROR
7074 027454 000240              NOP                    ;RETURN HERE IF ERROR
7075 027456 104000              ERROR                 ;ERROR # DEFINED BY SECERR SUBROUTINE
7076 027460 004736              JSR   PC,(SP)+       ;GO BACK FOR MORE ERROR CHECKS
7077 027462 000137 027664      JMP   180$          ;GO TO 180$ IF ERROR WAS FOUND
7078 027466
7079
7080
7081
7082 027466 012737 010000 001430 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN AT 16 BIT FORMAT
7083 027474 012737 000073 001376      MOV   #FMT16,RMOFO   ;CHANGE TO 16 BIT FORMAT
7084 027502 012737 103600 001402      MOV   #RH!GO,RMCS10  ;READ HEADER & DATA COMMAND
7085 027510 004737 037262      MOV   #BUFTWO,RMBA0  ;CHANGE BUS ADDRESS
7086 027514 000404              JSR   PC,PUT         ;GO WRITE REGISTERS VIA PUT SUB
7087 027516 000240              BR    130$          ;GO TO 130$ IF NO ERROR
7088 027520 104000              NOP                    ;RETURN HERE IF ERROR
7089 027522 000137 027664      ERROR                 ;ERROR DEFINED BY PUT SUB
7090 027526      JMP   180$          ;GO TO 180$ IF ERROR WAS FOUND
7091
7092
7093 027526 004737 037622      130$: ;WAIT FOR READ TO COMPLETE AND GET STATUS
7094 027532 004737 037012      JSR   PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
7095 027536 000404              JSR   PC,GET         ;GO READ REGISTERS VIA GET SUB
7096 027540 000240              BR    140$          ;GO TO 140$ IF NO ERROR
7097 027542 104000              NOP                    ;RETURN HERE IF ERROR
7098 027544 000137 027664      ERROR                 ;ERROR DEFINED BY GET SUB
7099 027550      JMP   180$          ;GO TO 180$ IF ERROR WAS FOUND
7100
7101
7102 027550 004737 040006      140$: ;VERIFY THE RESULTS OF READ OPERATION
7103 027554 000405              JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
7104 027556 000240              BR    150$          ;GO TO 150$ IF NO ERROR
7105 027560 104000              NOP                    ;RETURN HERE IF ERROR
7106 027562 004736              ERROR                 ;ERROR # DEFINED BY PRIERR SUBROUTINE
7107 027564 000137 027664      JSR   PC,(SP)+       ;GO BACK FOR MORE ERROR CHECKS
7108 027570      JMP   180$          ;GO TO 180$ IF ERROR WAS FOUND
7109
7110
7111 027570 032737 000020 001342 ;VERIFY THAT FORMAT ERROR IS SET
7112 027576 001022              BIT   #FER,RMER11    ;IS FORMAT ERROR SET??
7113 027600 004737 052312      BNE  160$          ;YES!!
              JSR   PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER

```

```

7114 027604 000405 BR 155$ ;GO TO 155$ IF NO ERROR
7115 027606 000240 NOP ;RETURN HERE IF ERROR
7116 027610 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7117 027612 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7118 027614 000137 027664 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
7119 027620 013737 001342 001142 155$: MOV RMER11,$BDDAT ;BAD DATA FOR TYPEOUT
7120 027626 013737 001342 001140 MOV RMER11,$GDDAT ;EXPECTED DATA
7121 027634 052737 000020 001140 BIS #FER,$GDDAT
7122 027642 104343 ERROR 343 ;FORMAT ERROR NOT SET
7123 027644 160$:
7124 027644 004737 040640 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
7125 027650 000405 BR 170$ ;GO TO 170$ IF NO ERROR
7126 027652 000240 NOP ;RETURN HERE IF ERROR
7127 027654 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
7128 027656 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7129 027660 000137 027664 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
7130 027664 170$:
7131 027664 180$:
7132
7133 ;*****
7134 ;*TEST 33 FORMAT ERROR TEST - 16
7135 ;*****
7136 027664 TST33:
7137 027664 000004 SCOPE ;SCOPE CALL
7138 027666 000240 NOP ;START OF TEST
7139 027670 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
7140 027674 013700 001276 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
7141 027700 013701 001446 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
7142 027704 012737 000033 001226 MOV #33,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
7143 027712 10$:
7144
7145 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
7146 027712 012737 000000 001432 MOV #0,RMDCO ;CYLINDER = 0
7147 027720 012737 000000 001404 MOV #0,RMDAO ;TRACK = SECTOR = 0
7148 027726 012737 010000 001430 MOV #FMT16,RMOFO ;16 BIT FORMAT
7149 027734 012737 177376 001400 MOV #(<C<2+256.>+1),RMWCO ;2 + 256 WORDS
7150 027742 012737 102574 001402 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
7151 027750 012737 000062 001376 MOV #WH,RMCS10 ;WRITE HEADER AND DATA
7152 027756 012737 064110 001174 MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
7153 027764 012737 000001 001176 MOV #1,$TMP1
7154 027772 004737 036114 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
7155 027776 20$:
7156
7157 ;PREPARE DEVICE FOR DATA TRANSFER
7158 027776 004737 033274 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
7159 030002 154130 .WORD 154130 ;TASK DESCRIPTOR
7160 030004 000404 BR 30$ ;GO TO 30$ IF NO ERROR
7161 030006 000240 NOP ;RETURN HERE IF ERROR
7162 030010 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7163 030012 000137 030406 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
7164 030016 30$:
7165
7166 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
7167 030016 012737 000063 001376 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
7168 030024 012702 001533 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
7169 030030 112722 000006 MOVB #RMDA,(R2)+

```


7170	030034	112722	000034
7171	030040	112722	000032
7172	030044	112722	000002
7173	030050	112722	000004
7174	030054	112722	000000
7175	030060	112722	000200
7176	030064	004737	037262
7177	030070	000404	
7178	030072	000240	
7179	030074	104000	
7180	030076	000137	030406
7181	030102		
7182			
7183			
7184	030102	004737	036726
7185			
7186			
7187	030106	004737	037622
7188	030112	004737	037012
7189	030116	000404	
7190	030120	000240	
7191	030122	104000	
7192	030124	000137	030406
7193	030130		
7194			
7195			
7196	030130	004737	040006
7197	030134	000405	
7198	030136	000240	
7199	030140	104000	
7200	030142	004736	
7201	030144	000137	030406
7202	030150		
7203	030150	004737	052312
7204	030154	000405	
7205	030156	000240	
7206	030160	104000	
7207	030162	004736	
7208	030164	000137	030406
7209	030170		
7210	030170	004737	040640
7211	030174	000405	
7212	030176	000240	
7213	030200	104000	
7214	030202	004736	
7215	030204	000137	030406
7216	030210		
7217			
7218			
7219			
7220	030210	012737	000000 001430
7221	030216	012737	000073 001376
7222	030224	012737	103600 001402
7223	030232	004737	037262
7224	030236	000404	
7225	030240	000240	

```

MOVW  #RMD0,(R2)+
MOVW  #RMOF,(R2)+
MOVW  #RMC,(R2)+
MOVW  #RMB0,(R2)+
MOVW  #RMC51,(R2)+
MOVW  #200,(R2)+
JSR   PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
BR    80$         ;GO TO 80$ IF NO ERROR
NOP                   ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP   180$        ;GO TO 180$ IF ERROR WAS FOUND

80$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR   PC,GETSTS

;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
JSR   PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
JSR   PC,GET      ;GO READ REGISTERS VIA GET SUB
BR    90$         ;GO TO 90$ IF NO ERROR
NOP                   ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY GET SUB
JMP   180$        ;GO TO 180$ IF ERROR WAS FOUND

90$:
;VERIFY RESULTS OF WRITE COMMAND
JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
BR    100$        ;GO TO 100$ IF NO ERROR
NOP                   ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
JMP   180$        ;GO TO 180$ IF ERROR WAS FOUND

100$:
JSR   PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
BR    110$        ;GO TO 110$ IF NO ERROR
NOP                   ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
JMP   180$        ;GO TO 180$ IF ERROR WAS FOUND

110$:
JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
BR    120$        ;GO TO 120$ IF NO ERROR
NOP                   ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
JMP   180$        ;GO TO 180$ IF ERROR WAS FOUND

120$:
;READ HEADER AND DATA FOR SECTOR JUST WRITTEN AT 18 BIT FORMAT
MOV   #0,RMOFO    ;SET FOR 18 BIT FORMAT
MOV   #RH!GO,RMC51 ;READ HEADER & DATA COMMAND
MOV   #BUFTW0,RMB0 ;CHANGE BUS ADDRESS
JSR   PC,PUT      ;GO WRITE REGISTERS VIA PUT SUB
BR    130$        ;GO TO 130$ IF NO ERROR
NOP                   ;RETURN HERE IF ERROR
  
```

```

7226 030242 104000          ERROR          ;ERROR DEFINED BY PUT SUB
7227 030244 000137 030406  JMP          180$      ;GO TO 180$ IF ERROR WAS FOUND
7228 030250
7229
7230
7231 030250 004737 037622  ;WAIT FOR READ TO COMPLETE AND GET STATUS
7232 030254 004737 037012  JSR          PC,TIMOUT ;WAIT FOR READ TO COMPLETE
7233 030260 000404          JSR          PC,GET    ;GO READ REGISTERS VIA GET SUB
7234 030262 000240          BR           140$     ;GO TO 140$ IF NO ERROR
7235 030264 104000          NOP          ;RETURN HERE IF ERROR
7236 030266 000137 030406  ERROR      ;ERROR DEFINED BY GET SUB
7237 030272          JMP          180$     ;GO TO 180$ IF ERROR WAS FOUND
7238
7239
7240 030272 004737 040006  ;VERIFY THE RESULTS OF READ OPERATION
7241 030276 000405          JSR          PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
7242 030300 000240          BR           150$     ;GO TO 150$ IF NO ERROR
7243 030302 104000          NOP          ;RETURN HERE IF ERROR
7244 030304 004736          ERROR      ;ERROR # DEFINED BY PRIERR SUBROUTINE
7245 030306 000137 030406  JSR          PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7246 030312          JMP          180$     ;GO TO 180$ IF ERROR WAS FOUND
7247
7248
7249 030312 032737 000020 001342 ;VERIFY THAT FORMAT ERROR IS SET
7250 030320 001022          BIT          #FER,RMER1I ;IS FORMAT ERROR SET??
7251 030322 004737 052312  BNE          160$     ;YES!!
7252 030326 000405          JSR          PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7253 030330 000240          BR           155$     ;GO TO 155$ IF NO ERROR
7254 030332 104000          NOP          ;RETURN HERE IF ERROR
7255 030334 004736          ERROR      ;ERROR # DEFINED BY DTASTS SUBROUTINE
7256 030336 000137 030406  JSR          PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7257 030342 013737 001342 001142 155$: MOV          RMER1I,$BDDAT ;GO TO 180$ IF ERROR WAS FOUND
7258 030350 013737 001342 001140  MOV          RMER1I,$GDDAT ;RECEIVED STATUS FOR TYPEOUT
7259 030356 052737 000020 001140  BIS          #FER,$GDDAT ;EXPECTED STATUS
7260 030364 104343          ERROR      343      ;FORMAT ERROR NOT SET
7261 030366
7262 030366 004737 040640  ;GO CHECK FOR SECONDARY ERRORS
7263 030372 000405          JSR          PC,SECERR ;GO TO 170$ IF NO ERROR
7264 030374 000240          BR           170$     ;RETURN HERE IF ERROR
7265 030376 104000          NOP          ;ERROR # DEFINED BY SECERR SUBROUTINE
7266 030400 004736          ERROR      ;GO BACK FOR MORE ERROR CHECKS
7267 030402 000137 030406  JSR          PC,(SP)+ ;GO TO 180$ IF ERROR WAS FOUND
7268 030406          JMP          180$
7269 030406
7270
7271
7272
7273
7274 030406
7275 030406 000004          ;*****
7276 030410 000240          ;*TEST 34      FORMAT HCE, FIRST HEADER WORD
7277 030412 012706 001100          ;*****
7278 030416 013700 001276  ;ST34:
7279 030422 013701 001446          SCOPE
7280 030426 012737 000034 001226  MOV          #STACK,SP ;SCOPE CALL
7281 030434 012737 000001 031412  MOV          $BASE,R0 ;START OF TEST
                                MOV          TSTQUE,R1 ;INITIALIZE STACK POINTER
                                MOV          #34,$TESTN ;R0=UNIBUS ADDRESS
                                MOV          #1,$190$ ;(R1) = DEVICE BEING TESTED
                                ;SET TEST NUMBER IN APT MAIL BOX
                                ;INIT BIT POSITION

```

```

7282 030442
7283
7284
7285 030442 012737 000000 001432
7286 030450 012737 000000 001404
7287 030456 012737 010000 001430
7288 030464 012737 177376 001400
7289 030472 012737 102574 001402
7290 030500 012737 000062 001376
7291 030506 012737 064110 001174
7292 030514 012737 000001 001176
7293 030522 004737 036114
7294 030526
7295
7296
7297 030526 004737 033274
7298 030532 154130
7299 030534 000404
7300 030536 000240
7301 030540 104000
7302 030542 000137 031410
7303 030546
7304
7305
7306 030546 033737 031412 102574
7307 030554 001404
7308 030556 043737 031412 102574
7309 030564 000403
7310 030566 053737 031412 102574
7311
7312 030574
7313
7314
7315 030574 012737 000063 001376
7316 030602 012702 001533
7317 030606 112722 000005
7318 030612 112722 000034
7319 030616 112722 000032
7320 030622 112722 000002
7321 030626 112722 000004
7322 030632 112722 000000
7323 030636 112722 000200
7324 030642 004737 037262
7325 030646 000404
7326 030650 000240
7327 030652 104000
7328 030654 000137 031410
7329 030660
7330
7331
7332 030660 004737 036726
7333
7334
7335 030664 004737 037622
7336 030670 004737 037012
7337 030674 000404

10$:
;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = SECTOR = 0
MOV #FMT16,RMFO ;16 BIT FORMAT
MOV #(<C<2+256.>+1),RMWCO ;2 + 256 WORDS
MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
MOV #WH,RMCS10 ;WRITE HEADER AND DATA
MOV #ZEROS,STMP0 ;USE ALL ZEROS DATA PATTERN
MOV #1,STMP1
JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20$:
;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 154130 ;TASK DESCRIPTOR
BR 30$ ;GO TO 30$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

30$:
;COMPLEMENT DATA BIT IN FIRST HEADER WORD
BIT 190$,BUFONE ;IS BIT IN HEADER ON??
BEQ 31$ ;NO!!
BIC 190$,BUFONE ;RESET BIT IN HEADER
BR 32$
31$: BIS 190$,BUFONE ;SET BIT IN HEADER
32$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
MOV #WH,GO,RMCS10 ;WRITE HEADER AND DATA
MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+
JSR PC,PUT ;GO WRITE REGISTERS VIA PUT SUB
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR DEFINED BY PUT SUB
JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND

80$:
;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC,GETSTS

;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
JSR PC,GET ;GO READ REGISTERS VIA GET SUB
BR 90$ ;GO TO 90$ IF NO ERROR

```

```

7338 030676 000240      NOP      ;RETURN HERE IF ERROR
7339 030700 104000      ERROR    ;ERROR DEFINED BY GET SUB
7340 030702 000137 031410  JMP      180$ ;GO TO 180$ IF ERROR WAS FOUND
7341 030706
7342
7343
90$:
;VERIFY RESULTS OF WRITE COMMAND
7344 030706 004737 040006  JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
7345 030712 000405      BR      100$      ;GO TO 100$ IF NO ERROR
7346 030714 000240      NOP      ;RETURN HERE IF ERROR
7347 030716 104000      ERROR    ;ERROR # DEFINED BY PRIERR SUBROUTINE
7348 030720 004736      JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7349 030722 000137 031410  JMP      180$      ;GO TO 180$ IF ERROR WAS FOUND
7350 030726
100$:
7351 030726 004737 052312  JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7352 030732 000405      BR      110$      ;GO TO 110$ IF NO ERROR
7353 030734 000240      NOP      ;RETURN HERE IF ERROR
7354 030736 104000      ERROR    ;ERROR # DEFINED BY DTASTS SUBROUTINE
7355 030740 004736      JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7356 030742 000137 031410  JMP      180$      ;GO TO 180$ IF ERROR WAS FOUND
7357 030746
110$:
7358 030746 004737 040640  JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
7359 030752 000405      BR      120$      ;GO TO 120$ IF NO ERROR
7360 030754 000240      NOP      ;RETURN HERE IF ERROR
7361 030756 104000      ERROR    ;ERROR # DEFINED BY SECERR SUBROUTINE
7362 030760 004736      JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7363 030762 000137 031410  JMP      180$      ;GO TO 180$ IF ERROR WAS FOUND
7364 030766
120$:
7365
7366
7367
;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
7368 030766 012737 000073 001376  MOV      #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
7369 030774 012737 103600 001402  MOV      #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
7370 031002 004737 037262      JSR      PC,PUT    ;GO WRITE REGISTERS VIA PUT SUB
7371 031006 000404      BR      130$      ;GO TO 130$ IF NO ERROR
7372 031010 000240      NOP      ;RETURN HERE IF ERROR
7373 031012 104000      ERROR    ;ERROR DEFINED BY PUT SUB
7374 031014 000137 031410  JMP      180$      ;GO TO 180$ IF ERROR WAS FOUND
7375 031020
130$:
7376
7377
;WAIT FOR READ TO COMPLETE AND GET STATUS
7378 031020 004737 037622      JSR      PC,TIMOUT ;WAIT FOR READ TO COMPLETE
7379 031024 004737 037012      JSR      PC,GET    ;GO READ REGISTERS VIA GET SUB
7380 031030 000404      BR      140$      ;GO TO 140$ IF NO ERROR
7381 031032 000240      NOP      ;RETURN HERE IF ERROR
7382 031034 104000      ERROR    ;ERROR DEFINED BY GET SUB
7383 031036 000137 031410  JMP      180$      ;GO TO 180$ IF ERROR WAS FOUND
7384 031042
140$:
7385
7386
;VERIFY THE RESULTS OF READ OPERATION
7387 031042 004737 040006  JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
7388 031046 000405      BR      150$      ;GO TO 150$ IF NO ERROR
7389 031050 000240      NOP      ;RETURN HERE IF ERROR
7390 031052 104000      ERROR    ;ERROR # DEFINED BY PRIERR SUBROUTINE
7391 031054 004736      JSR      PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7392 031056 000137 031410  JMP      180$      ;GO TO 180$ IF ERROR WAS FOUND
7393 031062
150$:

```

```

7394 031062 032737 140000 031412 BIT #MSE!USE,190$ ;IS THIS BSE ??
7395 031070 001033 BNE 152$ ;YES !!
7396 031072 032737 010000 031412 BIT #FMT16,190$ ;IS THIS FER ??
7397 031100 001466 BEQ 155$ ;NO !!
7398
7399 ;VERIFY THAT FER IS SET
7400 031102 032737 000020 001342 BIT #FER,RMER1I ;IS FER SET ??
7401 031110 001122 BNE 160$ ;YES !!
7402 031112 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7403 031116 000405 BR 151$ ;GO TO 151$ IF NO ERROR
7404 031120 000240 NOP ;RETURN HERE IF ERROR
7405 031122 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7406 031124 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7407 031126 000137 031410 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
7408 031132 013737 001342 001142 151$: MOV RMER1I,$BDDAT ;RECEIVED STATUS
7409 031140 013737 001342 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
7410 031146 052737 000020 001140 BIS #FER,$GDDAT
7411 031154 104343 ERROR 343 ;FORMAT ERROR NOT SET
7412 031156 000514 BR 180$
7413 031160
7414 ;VERIFY THAT BAD SECTOR ERROR IS SET
7415 031160 032737 100000 001370 BIT #BSE,RMER2I ;IS BSE SET ??
7416 031166 001073 BNE 160$ ;YES !!
7417 031170 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7418 031174 000405 BR 153$ ;GO TO 153$ IF NO ERROR
7419 031176 000240 NOP ;RETURN HERE IF ERROR
7420 031200 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7421 031202 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7422 031204 000137 031410 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
7423 031210
7424 031210 013737 001370 001142 153$: MOV RMER2I,$BDDAT ;RECEIVED STATUS
7425 031216 013737 001370 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
7426 031224 052737 100000 001140 BIS #BSE,$GDDAT
7427 031232 013737 102574 001174 MOV BUFO1,$STMP0 ;WRITTEN HEADER
7428 031240 013737 103600 001176 MOV BUFTWO,$STMP1 ;READ HEADER WORD
7429 031246 013737 031412 001200 MOV 190,$STMP2 ;FAILING BIT POSITION
7430 ; ERROR 345 ;FAILED TO DETECT BSE
7431 031254 000455 BR 180$ ;SKIP REST OF TEST
7432 031256
7433
7434 ;VERIFY THAT HEADER COMPARE ERROR IS SET
7435 031256 032737 000200 001342 BIT #HCE,RMER1I ;IS "HCE" SET??
7436 031264 001034 BNE 160$ ;YES!!
7437 031266 004737 052312 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7438 031272 000405 BR 156$ ;GO TO 156$ IF NO ERROR
7439 031274 000240 NOP ;RETURN HERE IF ERROR
7440 031276 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7441 031300 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7442 031302 000137 031410 JMP 180$ ;GO TO 180$ IF ERROR WAS FOUND
7443 031306 013737 001342 001142 156$: MOV RMER1I,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
7444 031314 013737 001342 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
7445 031322 052737 000200 001140 BIS #HCE,$GDDAT
7446 031330 013737 102574 001174 MOV BUFO1,$STMP0 ;WRITTEN HEADER
7447 031336 013737 103600 001176 MOV BUFTWO,$STMP1 ;READ HEADER WORD
7448 031344 013737 031412 001200 MOV 190,$STMP2 ;FAILING BIT POSITION
7449 031352 104344 ERROR 344 ;FORMAT ERROR NOT SET

```

```

7450 031354 000415          BR      180$
7451
7452 031356          160$:
7453
7454          ;CHECK FOR OTHER ERRORS
7455 031356 004737 040640      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
7456 031362 000405          BR      165$      ;GO TO 165$ IF NO ERROR
7457 031364 000240          NOP
7458 031366 104000          ERROR      ;RETURN HERE IF ERROR
7459 031370 004736          JSR      PC,(SP)+      ;ERROR # DEFINED BY SECERR SUBROUTINE
7460 031372 000137 031410      JMP      180$      ;GO BACK FOR MORE ERROR CHECKS
7461 031376          165$:      ;GO TO 180$ IF ERROR WAS FOUND
7462
7463
7464
7465          ;ADVANCE THE BIT POSITION AND FORMAT AGAIN IF NOT DONE
7466 031376 006337 031412      ASL      190$      ;SHIFT TO NEXT BIT POSITION
7467 031402 001402          BEQ      180$      ;EXIT IF DONE
7468 031404 000137 030442      JMP      10$      ;GO DO NEXT SECTOR
7469 031410          180$:
7470 031410 000401          BR      200$      ;JUMP OVER STORAGE
7471 031412          190$:
7472 031412 000000          .WORD
7473 031414          200$:      ;STORAGE FOR BIT POSITION
7474
7475          ;*****
7476          ;*TEST 35      FORMAT HCE, SECOND HEADER WORD
7477          ;*****
7478 031414          †TST35:
7479 031414 000004          SCOPE      ;SCOPE CALL
7480 031416 000240          NOP      ;START OF TEST
7481 031420 012706 001100      MOV      #STACK_SP      ;INITIALIZE STACK POINTER
7482 031424 013700 001276      MOV      $BASE,R0      ;R0=UNIBUS ADDRESS
7483 031430 013701 001446      MOV      TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
7484 031434 012737 000035 001226      MOV      #35,$TESTN      ;SET TEST NUMBER IN APT MAIL BOX
7485 031442 012737 000001 032224      MOV      #1,190$      ;INIT BIT POSITION
7486 031450          10$:
7487
7488          ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
7489 031450 012737 000000 001432      MOV      #0,RMDC0      ;CYLINDER = 0
7490 031456 012737 000000 001404      MOV      #0,RMDA0      ;TRACK = SECTOR = 0
7491 031464 012737 010000 001430      MOV      #FAT16,RMOFO      ;16 BIT FORMAT
7492 031472 012737 177376 001400      MOV      #(<C(2+256.)+1),RMWCO      ;2 + 256 WORDS
7493 031500 012737 102574 001402      MOV      #BUFONE,RMBA0      ;DATA BUFFER ADDRESS
7494 031506 012737 000062 001376      MOV      #WH,RMC$10      ;WRITE HEADER AND DATA
7495 031514 012737 064110 001174      MOV      #ZEROS,$TMP0      ;USE ALL ZEROS DATA PATTERN
7496 031522 012737 000001 001176      MOV      #1,$TMP1
7497 031530 004737 036114          JSR      PC,$GENBUF      ;GO GENERATE DATA BUFFER
7498 031534          20$:
7499
7500          ;PREPARE DEVICE FOR DATA TRANSFER
7501 031534 004737 033274          JSR      PC,$TSTPRP      ;PREPARE DEVICE FOR TEST
7502 031540 154130          .WORD 154130 ;TASK DESCRIPTOR
7503 031542 000404          BR      30$      ;GO TO 30$ IF NO ERROR
7504 031544 000240          NOP
7505 031546 104000          ERROR      ;RETURN HERE IF ERROR
          ;ERROR # DEFINED BY TSTPRP SUBROUTINE

```

```

7506 031550 000137 032222          JMP      180$          ;GO TO 180$ IF ERROR WAS FOUND
7507 031554          30$:
7508
7509          ;COMPLEMENT DATA BIT IN SECOND HEADER WORD
7510 031554 033737 032224 102576      BIT      190$,BUFONE+2      ;IS BIT IN HEADER ON??
7511 031562 001404          BEQ      31$              ;NO!!
7512 031564 043737 032224 102576      BIC      190$,BUFONE+2      ;RESET BIT IN HEADER
7513 031572 000403          BR       32$
7514 031574 053737 032224 102576      31$:    BIS      190$,BUFONE+2      ;SET BIT IN HEADER
7515
7516 031602          32$:
7517
7518          ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
7519 031602 012737 000063 001376      MOV      #WH!GO, RMCS10      ;WRITE HEADER AND DATA
7520 031610 012702 001533          MOV      #PUTINX, R2        ;WRITE REGISTER INDEX TABLE
7521 031614 112722 000006          MOVVB   #RMDA, (R2)+
7522 031620 112722 000034          MOVVB   #RMDC, (R2)+
7523 031624 112722 000032          MOVVB   #RMOF, (R2)+
7524 031630 112722 000002          MOVVB   #RMNC, (R2)+
7525 031634 112722 000004          MOVVB   #RMBA, (R2)+
7526 031640 112722 000000          MOVVB   #RMCS1, (R2)+
7527 031644 112722 000200          MOVVB   #200, (R2)+
7528 031650 004737 037262          JSR     PC, PUT            ;GO WRITE REGISTERS VIA PUT SUB
7529 031654 000404          BR      80$              ;GO TO 80$ IF NO ERROR
7530 031656 000240          NOP
7531 031660 104000          ERROR   ;RETURN HERE IF ERROR
7532 031662 000137 032222          JMP     180$             ;ERROR DEFINED BY PUT SUB
7533 031666          80$:    ;GO TO 180$ IF ERROR WAS FOUND
7534
7535          ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
7536 031666 004737 036726          JSR     PC, GETSTS
7537
7538          ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
7539 031672 004737 037622          JSR     PC, TIMEOUT        ;WAIT FOR COMMAND TO COMPLETE
7540 031676 004737 037012          JSR     PC, GET            ;GO READ REGISTERS VIA GET SUB
7541 031702 000404          BR      90$              ;GO TO 90$ IF NO ERROR
7542 031704 000240          NOP
7543 031706 104000          ERROR   ;RETURN HERE IF ERROR
7544 031710 000137 032222          JMP     180$             ;ERROR DEFINED BY GET SUB
7545 031714          90$:    ;GO TO 180$ IF ERROR WAS FOUND
7546
7547          ;VERIFY RESULTS OF WRITE COMMAND
7548 031714 004737 040006          JSR     PC, PRIERR        ;GO CHECK FOR PRIMARY ERRORS
7549 031720 000405          BR      100$            ;GO TO 100$ IF NO ERROR
7550 031722 000240          NOP
7551 031724 104000          ERROR   ;RETURN HERE IF ERROR
7552 031726 004736          JSR     PC, J(SP)+        ;ERROR # DEFINED BY PRIERR SUBROUTINE
7553 031730 000137 032222          JMP     180$             ;GO BACK FOR MORE ERROR CHECKS
7554 031734          100$:  ;GO TO 180$ IF ERROR WAS FOUND
7555 031734 004737 052312          JSR     PC, DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
7556 031740 000405          BR      110$            ;GO TO 110$ IF NO ERROR
7557 031742 000240          NOP
7558 031744 104000          ERROR   ;RETURN HERE IF ERROR
7559 031746 004736          JSR     PC, J(SP)+        ;ERROR # DEFINED BY DTASTS SUBROUTINE
7560 031750 000137 032222          JMP     180$             ;GO BACK FOR MORE ERROR CHECKS
7561 031754          110$:  ;GO TO 180$ IF ERROR WAS FOUND

```

```

7562 031754 004737 040640      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
7563 031760 000405              BR     120$           ;GO TO 120$ IF NO ERROR
7564 031762 000240              NOP                    ;RETURN HERE IF ERROR
7565 031764 104000              ERROR                   ;ERROR # DEFINED BY SECERR SUBROUTINE
7566 031766 004736              JSR    PC,2(SP)+      ;GO BACK FOR MORE ERROR CHECKS
7567 031770 000137 032222      JMP    180$           ;GO TO 180$ IF ERROR WAS FOUND
7568 031774
7569
7570
7571
7572 031774 012737 000073 001376 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
7573 032002 012737 103600 001402 MOV    #RH!GO,RMCS10  ;READ HEADER & DATA COMMAND
7574 032010 004737 037262      JSR    PC,PUT         ;GO WRITE REGISTERS VIA PUT SUB
7575 032014 000404              BR     130$           ;GO TO 130$ IF NO ERROR
7576 032016 000240              NOP                    ;RETURN HERE IF ERROR
7577 032020 104000              ERROR                   ;ERROR DEFINED BY PUT SUB
7578 032022 000137 032222      JMP    180$           ;GO TO 180$ IF ERROR WAS FOUND
7579 032026
7580
7581
7582 032026 004737 037622      JSR    PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
7583 032032 004737 037012      JSR    PC,GET         ;GO READ REGISTERS VIA GET SUB
7584 032036 000404              BR     140$           ;GO TO 140$ IF NO ERROR
7585 032040 000240              NOP                    ;RETURN HERE IF ERROR
7586 032042 104000              ERROR                   ;ERROR DEFINED BY GET SUB
7587 032044 000137 032222      JMP    180$           ;GO TO 180$ IF ERROR WAS FOUND
7588 032050
7589
7590
7591 032050 004737 040006      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
7592 032054 000405              BR     150$           ;GO TO 150$ IF NO ERROR
7593 032056 000240              NOP                    ;RETURN HERE IF ERROR
7594 032060 104000              ERROR                   ;ERROR # DEFINED BY PRIERR SUBROUTINE
7595 032062 004736              JSR    PC,2(SP)+      ;GO BACK FOR MORE ERROR CHECKS
7596 032064 000137 032222      JMP    180$           ;GO TO 180$ IF ERROR WAS FOUND
7597 032070
7598
7599
7600 032070 032737 000200 001342 ;VERIFY THAT HEADER COMPARE ERROR IS SET
7601 032076 001034              BIT    #HCE,RMER11    ;IS "HCE" SET??
7602 032100 004737 052312      BNE    160$           ;YES!!
7603 032104 000405              JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
7604 032106 000240              BR     155$           ;GO TO 155$ IF NO ERROR
7605 032110 104000              NOP                    ;RETURN HERE IF ERROR
7606 032112 004736              ERROR                   ;ERROR # DEFINED BY DTASTS SUBROUTINE
7607 032114 000137 032222      JSR    PC,2(SP)+      ;GO BACK FOR MORE ERROR CHECKS
7608 032120 013737 001342 001142 155$: JMP    180$           ;GO TO 180$ IF ERROR WAS FOUND
7609 032126 013737 001342 001140 MOV    RMER11,$BDDAT  ;RECEIVED STATUS FOR TYPEOUT
7610 032134 052737 000200 001140 MOV    RMER11,$GDDAT  ;EXPECTED STATUS
7611 032142 013737 102576 001174 MOV    #HCE,$GDDAT
7612 032150 013737 103602 001176 MOV    BUFOE+2,$TMP0  ;WRITTEN HEADER
7613 032156 013737 032224 001200 MOV    BUFTWO+2,$TMP1 ;READ HEADER WORD
7614 032164 104344              MOV    190$,$TMP2    ;FAILING BIT POSITION
7615 032166 000415              ERROR                   ;FORMAT ERROR NOT SET
7616
7617 032170

```



```

7618
7619
7620 032170 004737 040640
7621 032174 000405
7622 032176 000240
7623 032200 104000
7624 032202 004736
7625 032204 000137 032222
7626 032210
7627
7628
7629
7630 032210 006337 032224
7631 032214 001402
7632 032216 000137 031450
7633 032222
7634 032222 000401
7635 032224
7636 032224 000000
7637 032226
7638
7639

```

```

;CHECK FOR OTHER ERRORS
      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      BR     165$          ;GO TO 165$ IF NO ERROR
      NOP
      ERROR  ;RETURN HERE IF ERROR
      JSR    PC,(SP)+      ;ERROR # DEFINED BY SECERR SUBROUTINE
      JMP    180$          ;GO BACK FOR MORE ERROR CHECKS
165$:                               ;GO TO 180$ IF ERROR WAS FOUND

;ADVANCE THE BIT POSITION AND FORMAT NEXT SECTOR IF NOT DONE
      ASL    190$          ;SHIFT TO NEXT BIT POSITION
      BEQ    180$          ;EXIT IF DONE
      JMP    10$           ;GO DO NEXT SECTOR
180$:
      BR     200$          ;JUMP OVER STORAGE
190$:
      .WORD
200$:                               ;STORAGE FOR BIT POSITION

;PUT NEWTEST HERE

```

```

7640 032226
7641 032226 000240
7642 032230 013700 001446
7643 032234 062700 000002
7644 032240 010037 001446
7645 032244 005710
7646 032246 001402
7647 032250 000137 006764
7648 032254 012737 001450 001446
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658 032262
7659 032262 000240
7660 032264 005037 001116
7661 032270 005037 001206
7662 032274 005237 001230
7663 032300 042737 100000 001230
7664 032306 005327
7665 032310 000001
7666 032312 003063
7667 032314 012737
7668 032316 000001
7669 032320 032310
7670 032322 104401 032330
7671 032326 000407
7672
7673 032346
7674 032346 013746 001230
7675
7676 032352 104405
7677 032354 104401 032362
7678 032360 000421
7679
7680 032424
7681 032424 013746 001126
7682
7683 032430 104405
7684 032432 104401 001217
7685 032436 005037 001126
7686 032442 013700 000042
7687 032446 001405
7688 032450 000005
7689 032452 004710
7690 032454 000240
7691 032456 000240
7692 032460 000240
7693 032462
7694 032462 000137
7695 032464 006764

```

```

SEOSP:
NOP
MOV TSTQUE,RO
ADD #2,RO
MOV RO,TSTQUE
TST (RO)
BEQ IS
JMP READY
MOV #TSTQUE+2,TSTQUE
IS:
.SBTTL END OF PASS ROUTINE

*****
;INCREMENT THE PASS NUMBER ($PASS)
;TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYY"
;WHERE XXXXX AND YYYY ARE DECIMAL NUMBERS
;IF THERES A MONITOR GO TO IT
;IF THERE ISN'T JUMP TO READY

SEOP:
NOP
CLR $STNM ;:ZERO THE TEST NUMBER
CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
INC $PASS ;:INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
SEOPCT: .WORD 1
BGT $DOAGN ;:YES
MOV (PC)+,a(PC)+ ;:RESTORE COUNTER
SENDCT: .WORD 1
TYPE ,65$ ;:TYPE ASCIZ STRING
BR ,64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <12><15>/END PASS #/
;:64$: MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
;:TYPE PASS NUMBER
;:GO TYPE--DECIMAL ASCII WITH SIGN
TYPDS
TYPE ,67$ ;:TYPE ASCIZ STRING
BR ,66$ ;:GET OVER THE ASCIZ
;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
;:66$: MOV $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
;:TOTAL NUMBER OF ERRORS
;:GO TYPE--DECIMAL ASCII WITH SIGN
TYPDS
TYPE $CRLF ;:TYPE CARRIAGE RETURN, LINE FEED
CLR $ERTTL ;:CLEAR ERROR TOTAL
$GET42: MOV a#42,RO ;:GET MONITOR ADDRESS
BEQ $DOAGN ;:BRANCH IF NO MONITOR
RESET ;:CLEAR THE WORLD
SENDAD: JSR PC,(RO) ;:GO TO MONITOR
NOP ;:SAVE ROOM
NOP ;:FOR
NOP ;:ACT11
$DOAGN:
$RTNAD: JMP a(PC)+ ;:RETURN
.WORD READY

```

K12

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 153
END OF PASS ROUTINE

SEQ 0155

7696 032466 377 377 000 SNULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
7697 032472 .EVEN

```

7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713 032472
7714 032472 104414
7715 032474 032777 020000 146452
7716 032502 001402
7717 032504 000137 033222
7718
7719 032510 104401 001217
7720 032514 104401 033236
7721 032520 013746 001234
7722
7723 032524 104403
7724 032526 003
7725 032527 000
7726 032530 005037 033226
7727 032534 013737 001226 033226
7728 032542 104401 033244
7729 032546 013746 033226
7730
7731 032552 104403
7732 032554 003
7733 032555 000
7734 032556 005037 033230
7735 032562 113737 001130 033230
7736 032570 001406
7737 032572 104401 033254
7738 032576 013746 033230
7739
7740 032602 104403
7741 032604 003
7742 032605 000
7743 032606 104401 033263
7744 032612 013746 001132
7745
7746 032616 104403
7747 032620 006
7748 032621 001
7749
7750 032622 005737 033230
7751 032626 001575
7752 032630 104401 001217
7753 032634 105037 033234

```

```

.SBTTL SUBROUTINES
;*****
.SBTTL ERROR TYPEOUT ROUTINE
;*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
;*REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
;*
;*.UNIT NUMBER, TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER ARE
;*PRINTED ON THE FIRST LINE;
;*.ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
;*ONE OR MORE SUCCEEDING LINES;
;*.PAIRED LINES OF ERROR HEADERS AND ERROR DATA
;*ARE PRINTED AFTER THE ERROR MESSAGE.

ERRTYP:
    SAVREG
    BIT     #SW13,JSWR      ;INHIBIT TYPEOUTS??
    BEQ    1$              ;NO!!
    JMP    21$             ;YES!!
;TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
1$:
    TYPE   $CRLF
    TYPE   $ERTY00
    MOV    $UNIT,-(SP)
;TYPE "UNT#"
;SAVE $UNIT FOR TYPEOUT
;TYPE UNIT NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 3 DIGIT(S)
;SUPPRESS LEADING ZEROS
;LOAD TEST NUMBER FOR
    TYPOS
    .BYTE  3
    .BYTE  0
    CLR    TSTNMB
    MOV    $TSTN,TSTNMB
    TYPE   $ERTY01
    MOV    $TSTNMB,-(SP)
;TYPE "TST#"
;SAVE TSTNMB FOR TYPEOUT
;TYPE TEST NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 3 DIGIT(S)
;SUPPRESS LEADING ZEROS
;LOAD ERROR NUMBER FOR
    TYPOS
    .BYTE  3
    .BYTE  0
    CLR    ERRNMB
    MOV    $ITEMB,ERRNMB
    TYPEOUT
    BEQ    2$
;SKIP IF NO ERROR CALLED
;TYPE "ERR#"
;SAVE ERRNMB FOR TYPEOUT
;TYPE ERROR NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 3 DIGIT(S)
;SUPPRESS LEADING ZEROS
    TYPOS
    .BYTE  3
    .BYTE  0
    TYPE   $ERTY03
    MOV    $ERRPC,-(SP)
;SAVE $ERRPC FOR TYPEOUT
;TYPE PROGRAM COUNTER
;GO TYPE--OCTAL ASCII
;TYPE 6 DIGIT(S)
;TYPE LEADING ZEROS
;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
3$:
    TST    ERRNMB
    BEQ    21$
    TYPE   $CRLF
    CLRB   $BOTFLG
;CLEAR BOT FLAG
2$:
    TYPE   $PC=
;TYPE "PC="

```

7754	032640	105037	033235	CLRB	CHRCNT		:CLEAR CHARACTER COUNTER
7755	032644	013700	033230	MOV	ERRNMB,R0		:R0 POINTS TO FIRST OF
7756	032650	006300		ASL	R0		:FOUR ENTRIES IN ERROR
7757	032652	006300		ASL	R0		:TABLE
7758	032654	006300		ASL	R0		
7759	032656	062700	001552	ADD	#SERRTB-8.,R0		
7760	032662	011001		MOV	(R0),R1		:R1 POINTS TO ERROR MESSAGE
7761							:TABLE
7762	032664	001507		BEQ	13\$:BRANCH IF NO ERROR MESSAGE
7763				:TYPE THE ERROR	MESSAGE		
7764	032666	012102		4\$:	MOV	(R1)+,R2	:R2=ADDRESS OF MESSAGE STRING
7765	032670	001505		BEQ	12\$:BRANCH IF END OF MESSAGE
7766	032672	010237	033040	MOV	R2,11\$:LOAD ADDRESS OF STRING
7767	032676	005037	033232	CLR	BOTADR		:CLEAR BOT ADDRESS
7768	032702	112203		5\$:	MOVB	(R2)+,R3	:END OF STRING??
7769	032704	001454		BEQ	10\$:YES!!
7770	032706	122703	000015	CMPB	#CR,R3		:CARRIAGE RETURN??
7771	032712	001003		BNE	6\$:NO!!
7772	032714	105037	033235	CLRB	CHRCNT		:YES-CLEAR CHAR COUNT
7773	032720	000770		BR	5\$:GET NEXT CHARACTER
7774	032722	122703	000012	6\$:	CMPB	#LF,R3	:LINE FEED??
7775	032726	001765		BEQ	5\$:YES-GET NEXT CHARACTER
7776	032730	122703	000011	CMPB	#HT,R3		:HORIZONTAL TAB??
7777	032734	001007		BNE	8\$:NO!!
7778	032736	105237	033235	7\$:	INCB	CHRCNT	:ADJUST CHARACTER COUNT
7779	032742	132737	000007	033235	BITB	#7,CHRCNT	
7780	032750	001372		BNE	7\$		
7781	032752	000407		BR	9\$		
7782	032754	105237	033235	8\$:	INCB	CHRCNT	:INCREMENT CHARACTER COUNT
7783	032760	122703	000040	CMPB	#',R3		:SPACE??
7784	032764	001002		BNE	9\$:NO!!
7785	032766	010237	033232	MOV	R2,BOTADR		:SAVE ADDRESS OF SPACE
7786	032772	122737	000100	033235	9\$:	CMPB	#64.,CHRCNT
7787	033000	103340		BHIS	5\$:END OF LINE??
7788	033002	013704	033232	MOV	BOTADR,R4		:GET ADDRESS OF LAST SPACE
7789	033006	001007		BNE	90\$:BRANCH IF SPACE DETECTED
7790	033010	104401	001217	TYPE	,SCRLF		:TYPE CRLF
7791	033014	105037	033235	CLRB	CHRCNT		:CLEAR CHARACTER COUNT
7792	033020	013702	033040	MOV	11\$,R2		:SET UP R2 FOR TESTING
7793	033024	000726		BR	5\$		
7794	033026	105044		90\$:	CLRB	-(R4)	:REPLACE SPACE
7795	033030	112737	177777	033234	MOVB	#-1,BOTFLG	:SET BOT FLAG
7796	033036	104401		10\$:	TYPE		:TYPE ERROR MESSAGE STRING
7797	033040	000000		11\$:	.WORD		:STRING ADDRESS GOES HERE
7798	033042	105737	033234	TSTB	BOTFLG		:WAS STRING TRUNCATED??
7799	033046	001707		BEQ	4\$:NO!!
7800	033050	104401	001217	TYPE	,SCRLF		:YES-TYPE CRLF
7801	033054	105037	033234	CLRB	BOTFLG		:CLEAR BOT FLAG
7802	033060	105037	033235	CLRB	CHRCNT		:CLEAR CHARACTER COUNT
7803	033064	013702	033232	MOV	BOTADR,R2		:SETUP R2 FOR TESTING
7804	033070	010237	033040	MOV	R2,11\$:SETUP 11\$ FOR TYPING
7805	033074	112742	000040	MOVB	#',-(R2)		:RESTORE SPACE
7806	033100	105722		TSTB	(R2)+		:RESTORE R2
7807	033102	000677		BR	5\$:TYPE REST OF STRING
7808	033104			12\$:			
7809				:TYPE ERROR HEADER AND ERROR DATA			

7810	033104				13\$:	MOV	2(R0),R1		;R1 POINTS TO ERROR HEADER TABLE
7811	033104	016001	000002			BEQ	21\$;BRANCH IF NO HEADER
7812	033110	001444				TYPE	\$CRLF		;(ASSUME NO DATA)
7813	033112	104401	001217			MOV	4(R0),R2		;R2 POINTS TO DATA ADDRESS TABLE
7814	033116	016002	000004			MOV	6(R0),R3		;R3 POINTS TO FORMAT TABLE
7815	033122	016003	000006		14\$:	MOV	(R1)+,15\$;PUT HEADER ADDRESS FOR TYPE
7816	033126	012137	033136			BEQ	21\$;BRANCH IF END OF HEADERS
7817	033132	001433							;(ASSUME END OF DATA)
7818									
7819	033134	104401				TYPE			
7820	033136	000000			15\$:	.WORD	0		;HEADER ADDRESS GOES HERE
7821	033140	104401	001217			TYPE	\$CRLF		
7822	033144	005702				TST	R2		;DATA WITH HEADER??
7823	033146	001767				BEQ	14\$;NO!!
7824	033150	012204				MOV	(R2)+,R4		;R4 POINTS TO DATA ADDRESS
7825	033152	012305				MOV	(R3)+,R5		;R5 POINTS TO FORMAT
7826	033154	105725			16\$:	TSTB	(R5)+		;WHAT KIND OF DATA??
7827	033156	100407				BMI	18\$;BINARY
7828	033160	001403				BEQ	17\$;OCTAL
7829	033162	013446				MOV	2(R4)+,-(SP)		;DECIMAL
7830	033164	104405				TYPDS			
7831	033166	000405				BR	19\$		
7832	033170	013446			17\$:	MOV	2(R4)+,-(SP)		
7833	033172	104402				TYPOC			
7834	033174	000402				BR	19\$		
7835	033176	013446			18\$:	MOV	2(R4)+,-(SP)		
7836	033200	104406				TYPBN			
7837	033202	005714			19\$:	TST	(R4)		;MORE DATA??
7838	033204	001403				BEQ	20\$;NO!!
7839	033206	104401	033271			TYPE	ERTY04		;YES-TYPE 2 SPACES
7840	033212	000760				BR	16\$;AND CONTINUE
7841	033214	104401	001217		20\$:	TYPE	\$CRLF		;TYPE ONE BLANK LINE
7842	033220	000742				BR	14\$;BEFORE NEXT HEADER
7843	033222	104415			21\$:	RESREG			
7844	033224	000207				RTS	PC		
7845									
7846	033226	000000				TSTNMB:	.WORD 0		;TEST NUMBER
7847	033230	000000				ERRNMB:	.WORD 0		;ERROR NUMBER
7848	033232	000000				BOTADR:	.WORD 0		;BEGINNING OF TEXT ADDRESS
7849	033234	000				BOTFLG:	.BYTE 0		;BOT FLAG
7850	033235	000				CHRCNT:	.BYTE 0		;CHARACTER COUNT
7851									
7852	033236	047125	052111	000043	ERTY00:	.ASCIZ	2UNIT#2		
7853	033244	020054	042524	052123	ERTY01:	.ASCIZ	2, TEST#2		
7854	033252	000043							
7855	033254	020054	051105	021522	ERTY02:	.ASCIZ	2, ERR#2		
7856	033262	000							
7857	033263	054	050040	036503	ERTY03:	.ASCIZ	2, PC=2		
7858	033270	000							
7859	033271	040	000040		ERTY04:	.ASCIZ	2 2		
7860									
7861						.EVEN			

7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917

033274
033274 017637 000000 034206
033302 062716 000006
033306 105076 000000
033312 162716 000004

033316 032737 100000 034206
033324 001414
033326 004737 044502
033332 000411
033334 000401
033336 000000
033340 062716 000004 000000
033344 113776 033336 000000
033352 000137 034174
033356
033364 032737 040000 034206
033364 001453

.SBTTL TEST PREPARATION MODULE

; THIS MODULE PREPARES THE RMO3 SUBSYSTEM FOR THE EXECUTION OF A TEST,
; REPORTING AN ERROR TO THE USER IF AN ERROR IS DETECTED. THE USER
; SPECIFIES TASKS TO BE PERFORMED, WHICH THE MODULE EXECUTES
; USING SUBROUTINES.

:CALL:
JSR PC,TSTPRP
.WORD
BR ??
NOP
ERROR

TASK/VERIFY DESCRIPTOR
RETURN HERE IF NO ERROR
RETURN HERE IF ERROR
ERROR DEFINED BY MODULE

:TASK/VERIFY DESCRIPTOR
BIT 15 = 1 SELECT DEVICE AND VERIFY DEVICE IS AVAILABLE
BIT 14 = 1 CLEAR CONTROLLER AND SELECT DEVICE
BIT 13 (RESERVED FOR DRIVE CLEAR)
BIT 12 = 1 PACK ACKNOWLEDGE IF VOLUME NOT VALID
BIT 11 = 1 RECALIBRATE IF POSITIONING IN PROGRESS
BIT 10
BIT 9
BIT 8
BIT 7
BIT 6 = 1 VERIFY CONTROLLER CLEAR OPERATION
BIT 5 (RESERVED FOR DRIVE CLEAR)
BIT 4 = 1 VERIFY PACK ACKNOWLEDGE
BIT 3 = 1 VERIFY RECALIBRATION
BIT 2
BIT 1
BIT 0

TSTPRP:
;STORE TASK DESCRIPTOR AND CLEAR USER'S ERROR CALL
MOV @ (SP),500\$;STORE DESCRIPTOR
ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
CLRB @ (SP) ;CLEAR ERROR CALL
SUB #4,(SP) ;MOVE SP TO NO ERROR RETURN
;*****
;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
BIT #BIT15,500\$;SELECT DEVICE??
BEQ 30\$;NO!!
JSR PC,DEVSEL ;GO SELECT DEVICE
BR 30\$;NO ERROR - CONTINUE
BR 20\$
10\$: .WORD ;ERROR NUMBER FROM DEVSEL
20\$: ADD #4,(SP) ;TRANSFER ERROR TO USER
MOVW 10\$,@ (SP)
JMP 400\$
30\$:
;*****
;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
BIT #BIT14,500\$;CLEAR CONTROLLER??
BEQ 120\$;NO!!

```

7918 033366 004737 046154          JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
7919 033372 000411          BR     60$           ;CONTINUE - NO ERROR
7920 033374 000401          BR     50$
7921 033376 000000          40$: .WORD          ;ERROR NUMBER FROM CNTCLR
7922 033400 062716 000004          50$: ADD     #4,(SP)  ;TRANSFER ERROR TO USER
7923 033404 113776 033376 000000          MOVB  40$,2(SP)
7924 033412 000137 034174          JMP   400$
7925 033416          60$:
7926          ;*****
7927          ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
7928 033416 032737 000100 034206          BIT   #BIT6,500$    ;VERIFY??
7929 033424 001433          BEQ   120$          ;NO!!
7930 033426 004737 036726          JSR   PC,GETSTS     ;SETUP INDEX TABLE
7931 033432 004737 037012          JSR   PC,GET        ;GO GET STATUS
7932 033436 000411          BR    90$           ;NO ERROR GETTING STATUS
7933 033440 000401          BR    80$
7934 033442 000000          70$: .WORD          ;ERROR FROM GETTING STATUS
7935 033444 062716 000004          80$: ADD     #4,(SP)  ;TRANSFER ERROR TO USER
7936 033450 113776 033442 000000          MOVB  70$,2(SP)
7937 033456 000137 034174          JMP   400$
7938 033462 004737 046272          90$: JSR   PC,CLRSTS   ;GO VERIFY STATUS CLEAR
7939 033466 000412          BR    120$         ;NO ERROR IN CLEAR
7940 033470 000401          BR    110$
7941 033472 000000          100$: .WORD         ;ERROR IN STATUS CLEAR
7942 033474 005726          110$: TST   (SP)+    ;STRIP RETURN ADDRESS TO
7943 033476 062716 000004          ADD   #4,(SP)      ;SUBROUTINE AND TRANSFER
7944 033502 113776 033472 000000          MOVB  100$,2(SP)  ;ERROR TO USER
7945 033510 000137 034174          JMP   400$
7946 033514          120$:
7947          ;*****
7948          ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
7949          ;NOT VALID
7950          ;*****
7951 033514 032737 010000 034206          BIT   #BIT12,500$  ;PACK ACKNOWLEDGE??
7952 033522 001511          BEQ   240$          ;NO!!
7953 033524 112737 000012 001504          MOVB  #RMD5,GETINX ;GET RMD5
7954 033532 112737 000200 001505          MOVB  #200,GETINX+1
7955 033540 004737 037012          JSR   PC,GET
7956 033544 000411          BR    150$         ;NO ERROR GETTING RMD5
7957 033546 000401          BR    140$
7958 033550 000000          130$: .WORD
7959 033552 062716 000004          140$: ADD     #4,(SP)  ;TRANSFER ERROR TO USER
7960 033556 113776 033550 000000          MOVB  130$,2(SP)
7961 033564 000137 034174          JMP   400$
7962 033570 032737 000100 001340          150$: BIT   #VV,RMDSI ;IS VOLUME VALID??
7963 033576 001063          BNE   240$          ;YES!!
7964 033600 005037 001472          CLR   MEDENB       ;CLEAR MEDIA ENABLE
7965 033604 012737 000023 001376          MOV   #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
7966 033612 112737 000000 001533          MOVB  #RMCSI,PUTINX ;SETUP REGISTER INDEX TABLE
7967 033620 112737 000200 001534          MOVB  #200,PUTINX+1
7968 033626 004737 037262          JSR   PC,PUT
7969 033632 000410          BR    180$         ;GO WRITE COMMAND
7970 033634 000401          BR    170$         ;NO ERROR LOADING REGISTER
7971 033636 000000          160$: .WORD
7972 033640 062716 000004          170$: ADD     #4,(SP)  ;ERROR FROM PUT SUB
7973 033644 113776 033636 000000          MOVB  160$,2(SP)  ;TRANSFER ERROR TO USER

```



```

7974 033652 000550
7975 033654
7976
7977
7978
7979 033654 032737 000020 034206
7980 033662 001431
7981 033664 004737 036726
7982 033670 004737 037012
7983 033674 000410
7984 033676 000401
7985 033700 000000
7986 033702 062716 000004
7987 033706 113776 033700 000000
7988 033714 000527
7989 033716 004737 047152
7990 033722 000411
7991 033724 000401
7992 033726 000000
7993 033730 005726
7994 033732 062716 000004
7995 033736 113776 033726 000000
7996 033744 000513
7997 033746
7998
7999
8000
8001 033746 032737 004000 034206
8002 033754 001513
8003 033756 112737 000012 001504
8004 033764 112737 000200 001505
8005 033772 004737 037012
8006 033776 000410
8007 034000 000401
8008 034002 000000
8009 034004 062716 000004
8010 034010 113776 034002 000000
8011 034016 000466
8012 034020 032737 020000 001340
8013 034026 001466
8014 034030 012737 000007 001376
8015 034036 112737 000000 001533
8016 034044 112737 000200 001534
8017 034052 004737 037262
8018 034056 000410
8019 034060 000401
8020 034062 000000
8021 034064 062716 000004
8022 034070 113776 034062 000000
8023 034076 000436
8024 034100 004737 036726
8025 034104 004737 037622
8026
8027
8028
8029 034110 032737 000010 034206

```

```

BR 400$
180$:
;*****
;VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
BIT #BIT4,500$ ;VERIFY PACK ACKNOWLEDGE??
BEQ 240$ ;NO!!
JSR PC,GETSTS ;SETUP FOR STATUS
JSR PC,GET ;GO GET STATUS
BR 210$ ;NO ERROR GETTING STATUS
BR 200$
190$: .WORD ;ERROR FROM GET SUB
200$: ADD #4,(SP) ;TRANSFER ERROR TO USER
MOVB 190$,2(SP)
BR 400$
210$: JSR PC,ACKSTS ;GO CHECK ACKNOWLEDGE
BR 240$ ;NO ERROR
BR 230$
220$: .WORD ;PACK ACKNOWLEDGE ERROR
230$: TST (SP)+ ;STRIP RETURN TO SUB AND
ADD #4,(SP) ;TRANSFER ERROR TO USER
MOVB 220$,2(SP)
BR 400$
240$:
;*****
;RECALIBRATE DRIVE IF BIT 11 SET IN TASK AND PIP IS ACTIVE
BIT #BIT11,500$ ;RECALIBRATE??
BEQ 410$ ;NO!!
MOVB #RMD3,GETINX ;LOAD REGISTER INDEX TABLE
MOVB #200,GETINX+1
JSR PC,GET ;GO GET RMD3
BR 270$ ;NO ERROR GETTING RMD3
BR 260$
250$: .WORD ;ERROR FROM GET SUB
260$: ADD #4,(SP) ;TRANSFER ERROR TO USER
MOVB 250$,2(SP)
BR 400$
270$: BIT #PIP,RMDSI ;IS PIP ACTIVE??
BEQ 410$ ;NO!!
MOV #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
MOVB #RMCS1,POTINX ;AND REGISTER INDEX
MOVB #200,POTINX+1
JSR PC,POT ;GO ISSUE RECALIBRATE
BR 300$ ;NO ERROR
BR 290$
280$: .WORD ;ERROR IN REGISTER TRANSFER
290$: ADD #4,(SP) ;TRANSFER ERROR TO USER
MOVB 280$,2(SP)
BR 400$
300$: JSR PC,GETSTS ;SETUP FOR STATUS
JSR PC,TIMOUT ;WAIT FOR COMPLETION
;*****
;VERIFY RECALIBRATE IF BIT 3 SET IN TASK
BIT #BIT3,500$ ;VERIFY RECALIBRATE??

```

DZRM0A - RM03 FUNCTIONAL TEST, PART 2
 DZRM0A.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 160
 TEST PREPARATION MODULE

SEQ 0162

8030	034116	001432				BEQ	410\$;NO!!
8031	034120	004737	037012			JSR	PC,GET		;GO GET STATUS
8032	034124	000410				BR	330\$;NO ERROR GETTING STATUS
8033	034126	000401				BR	320\$		
8034	034130	000000			310\$:	.WORD			;ERROR FROM GET
8035	034132	062716	000004		320\$:	ADD	#4,(SP)		;TRANSFER ERROR TO USER
8036	034136	113776	034130	000000		MOVB	310\$,2(SP)		
8037	034144	000413				BR	400\$		
8038	034146	004737	047746		330\$:	JSR	PC,RCLSTS		;GO CHECK RECALIBRATE
8039	034152	000414				BR	410\$;NO ERROR DURING RECALIBRATE
8040	034154	000401				BR	350\$		
8041	034156	000000			340\$:	.WORD			;ERROR DURING RECALIBRATE
8042	034160	005726			350\$:	TST	(SP)+		;STRIP RETURN TO SUB AND
8043	034162	062716	000004			ADD	#4,(SP)		;TRANSFER ERROR TO USER
8044	034166	113776	034156	000000		MOVB	340\$,2(SP)		
8045	034174	162716	000002		400\$:	SUB	#2,(SP)		;MOVE SP BACK BEFORE ERROR
8046	034200	000240				NOP			
8047	034202	000240				NOP			
8048	034204	000207			410\$:	RTS	PC		;RETURN TO USER
8049									
8050	034206	000000			500\$:	.WORD			;TASK/VERIFY DESCRIPTOR

```

8051
8052
8053
8054
8055
8056
8057
8058
8059
8060
8061
8062
8063
8064
8065
8066
8067
8068
8069
8070
8071
8072
8073
8074
8075
8076
8077
8078
8079
8080
8081
8082
8083 034210
8084
8085
8086
8087 034210 005737 001472
8088 034214 001402
8089 034216 000137 035434
8090
8091
8092
8093 034222
8094
8095
8096 034222 010046
8097 034224 005000
8098 034226 016060 001376 102574
8099 034234 062700 000002
8100 034240 022700 000046
8101 034244 103370
8102
8103
8104 034246 012737 000003 036102
8105 034254 012737 002000 001404
8106 034262 012737 001466 001432

```

```

.SBTTL BAD SECTOR MODULE

;THE BAD SECTOR MODULE PERFORMS TWO MAJOR FUNCTIONS, I.E.,
;RECOVER THE BAD SECTOR FILE FROM THE LAST TRACK, AND,
;APPROVE THE USAGE OF A SECTOR BASED ON INFORMATION IN
;THE BAD SECTOR FILE OR ASSIGN A NEW SECTOR IF THE ONE ELECTED IS
;NOT AVAILABLE FOR USE.

;INFORMATION REQUIRED BY THE MODULE INCLUDES
;RMDCO, THE DESIRED CYLINDER,
;RMDAO, THE TRACK AND SECTOR ADDRESS,
;RMMC0, THE WORD COUNT,
;RMC510, THE COMMAND.

;MODULE CALL IS AS FOLLOWS,

JSR PC,BADSCT
BR ??? ;RETURN HERE IF NO ERROR
TYPE ,MESSAGE ;RETURN HERE IF THE BAD SECTOR FILE
;CANNOT BE RECOVERED
ERROR THE MODULE DEFINES THE ERROR NUMBER

;THE MODULE IS INTENDED TO BE CALLED PRIOR TO CALLING THE BUFFER
;GENERATOR SUBROUTINE, AND PRESERVES THE "PUT BUFFER" SO THAT THE
;BUFFER NEED ONLY BE FILLED ONCE FOR THE EXECUTION OF A FORMAT
;OPERATION.

;THE MODULE RETURNS TO THE CALLING TEST WITH THE APPROVED OR ASSIGNED
;SECTOR IN THE PUT BUFFER AND ALSO IN LOCATIONS "ASND" AND "ASNC"
;SO THAT A REFERENCE IS AVAILABLE TO THE TEST OUTSIDE OF THE PUT BUFFER.

BADSCT:

;TEST "MFDIA ENABLE" TO DETERMINE WHETHER OR NOT THE BAD SECTOR FILES
;HAVE BEEN RECOVERED.
TST MEDENB
BEQ 10$
JMP 300$ ;BAD SECTOR FILE IS AVAILABLE

;*****
;RECOVER THE BAD SECTOR FILE FROM THE LAST TRACK
10$:
;SAVE THE USER'S PUT BUFFER
MOV RO,-(SP) ;;PUSH RO ON STACK
CLR RO
20$: MOV PUTBUF(RO),BUFFER(RO)
ADD #2,RO ;ADVANCE TO NEXT BUFFER POSITION
CMP #46,RO ;END OF BUFFER
BHS 20$ ;NO !!

;SET RETRY COUNT AND LOAD PUT BUFFER AND REGISTER INDEX TABLE
MOV #3,500$ ;RETRY COUNT
MOV #002000,RMDAO ;STARTING SECTOR ADDRESS
MOV #822.,RMDCO ;DESIRED CYLINDER

```

```

0107 034270 012737 177376 001400      MOV      #(<C258.+1>,RMMCO      ;WORD COUNT
0108 034276 012737 010000 001430      MOV      #FMT16,RMOF0        ;16 BIT FORMAT
0109 034304 012737 100554 001402      MOV      #MFGFIL,RMBA0       ;BUFFER ADDRESS
0110
0111 034312 012700 001533      MOV      #PUTINX,RO          ;RO POINTS TO REGISTER INDEX TABLE
0112 034316 112720 000006      MOV      #RMDA,(RO)+
0113 034322 112720 000034      MOV      #RMDC,(RO)+
0114 034326 112720 000002      MOV      #RMMC,(RO)+
0115 034332 112720 000032      MOV      #RMOF,(RO)+
0116 034336 112720 000004      MOV      #RMBA,(RO)+
0117 034342 112720 000000      MOV      #RMCS1,(RO)+
0118 034346 112720 000200      MOV      #200,(RO)+
0119 034352 012600      MOV      (SP)+,RO            ;;POP STACK INTO RO
0120
0121      ;SET GET INDEX TABLE FOR READING STATUS
0122 034354 004737 036726      JSR      PC,GETSTS          ;SETUP GET INDEX REGISTER FOR STATUS
0123 034360      30$:
0124
0125      ;CLEAR THE DEVICE USING DRIVE CLEAR COMMAND
0126 034360 012737 000011 001376      MOV      #DRVCLR!GO,RMCS10   ;LOAD COMMAND IN PUT BUFFER
0127 034366 004737 037262      JSR      PC,PUT              ;OUTPUT COMMAND
0128 034372 000411      BR      45$                 ;RETURN HERE IF NO ERROR
0129 034374 000401      BR      40$                 ;GET AROUND ERROR #
0130 034376 000000      35$: .WORD                   ;ERROR # GOES HERE
0131
0132 034400 062716 000006      40$: ADD      #6,(SP)          ;MOVE SP TO USERS ERROR CALL
0133 034404 113776 034376 000000      MOV      35$,2(SP)          ;MOVE ERROR NUMBER TO USER
0134 034412 000137 035112      JMP      215$
0135 034416 004737 037012      45$: JSR      PC,GET              ;GO GET STATUS
0136 034422 000411      BR      60$                 ;RETURN HERE IF NO ERROR
0137 034424 000401      BR      55$                 ;GET AROUND ERROR #
0138 034426 000000      50$: .WORD                   ;ERROR # GOES HERE
0139
0140 034430 062716 000006      55$: ADD      #6,(SP)          ;MOVE SP TO USERS ERROR CALL
0141 034434 113776 034426 000000      MOV      50$,2(SP)          ;MOVE ERROR # TO USERS ERROR CALL
0142 034442 000137 035112      JMP      215$
0143
0144 034446 004737 051510      60$: JSR      PC,DRVSTS         ;GO VERIFY DRIVE CLEAR COMMAND
0145 034452 000412      BR      75$                 ;RETURN HERE IF NO ERROR
0146 034454 000401      BR      70$                 ;GET AROUND ERROR
0147 034456 000000      65$: .WORD                   ;ERROR # GOES HERE
0148
0149 034460 005726      70$: TST      (SP)+           ;STRIP RETURN TO SUBROUTINE
0150 034462 062716 000006      ADD      #6,(SP)           ;MOVE SP TO USERS ERROR CALL
0151 034466 113776 034456 000000      MOV      65$,2(SP)         ;MOVE ERROR # TO USER CALL
0152 034474 000137 035112      JMP      215$
0153
0154 034500      75$:
0155
0156      ;ISSUE A PACK ACKNOWLEDGE IF VOLUME VALID IS RESET
0157 034500 032737 000100 001340      BIT      #VV,RMDSI          ;IS VV RESET ??
0158 034506 001050      BNE     120$                ;NO !!
0159 034510 012737 000023 001376      MOV      #PACACK!GO,RMCS10   ;LOAD COMMAND
0160 034516 004737 037262      JSR      PC,PUT              ;GO PUT COMMAND TO DRIVE
0161 034522 000411      BR      90$                 ;RETURN HERE IF NO OUTPUT ERROR
0162 034524 000401      BR      85$                 ;GET AROUND ERROR #

```

```

8163 034526 000000      80$: .WORD          ;ERROR # GOES HERE
8164
8165 034530 062716 000006      85$: ADD #6,(SP)    ;MOVE SP TO USERS ERROR CALL
8166 034534 113776 034526 000000      MOVB 80$,2(SP)    ;MOVE ERROR # TO ERROR CALL
8167 034542 000137 035112      JMP 215$
8168 034546 004737 037012      90$: JSR PC,GET    ;GO GET STATUS FOR PACK ACK
8169 034552 000411      BR 105$          ;RETURN HERE IF NO ERROR
8170 034554 000401      BR 100$          ;GET AROUND ERROR #
8171 034556 000000      95$: .WORD          ;ERROR # GOES HERE
8172
8173 034560 062716 000006      100$: ADD #6,(SP)   ;MOVE SP TO USERS ERROR CALL
8174 034564 113776 034556 000000      MOVB 95$,2(SP)   ;MOVE ERROR # TO CALL
8175 034572 000137 035112      JMP 215$
8176
8177 034576 004737 047152      105$: JSR PC,ACKSTS ;GO VERIFY ACKNOWLEDGE STATUS
8178 034602 000412      BR 120$          ;RETURN HERE IF NO ERROR
8179 034604 000401      BR 115$          ;GET AROUND ERROR #
8180 034606 000000      110$: .WORD          ;ERROR # GOES HERE
8181
8182 034610 005726      115$: TST (SP)+    ;STRIP RETURN TO SUBROUTINE
8183 034612 062716 000006      ADD #6,(SP)     ;MOVE SP TO USERS ERROR CALL
8184 034616 113776 034606 000000      MOVB 110$,2(SP) ;MOVE ERROR # TO USERS ERROR CALL
8185 034624 000137 035112      JMP 215$
8186
8187 034630      120$:
8188
8189      ;RECALIBRATE THE DRIVE IF PIP IS SET
8190 034630 032737 020000 001340      BIT #PIP,RMSI   ;IS PIP SET ??
8191 034636 001452      BEQ 165$        ;NO !!
8192
8193 034640 012737 000007 001376      MOV #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
8194 034646 004737 037262      JSR PC,PUT      ;PUT THE RECAL COMMAND
8195 034652 000411      BR 135$        ;RETURN HERE IF NO ERROR
8196 034654 000401      BR 130$        ;GET AROUND ERROR #
8197 034656 000000      125$: .WORD          ;ERROR # GOES HERE
8198
8199 034660 062716 000006      130$: ADD #6,(SP)   ;MOVE SP TO USERS ERROR CALL
8200 034664 113776 034656 000000      MOVB 125$,2(SP) ;MOVE ERROR # TO USERS CALL
8201 034672 000137 035112      JMP 215$
8202
8203 034676 004737 037622      135$: JSR PC,TIMOUT   ;WAIT FOR RECALIBRATE TO COMPLETE
8204 034702 004737 037012      JSR PC,GET      ;GO GET RECAL STATUS
8205 034706 000411      BR 150$        ;RETURN HERE IF NO ERROR
8206 034710 000401      BR 145$        ;GET AROUND ERROR #
8207 034712 000000      140$: .WORD          ;ERROR # GOES HERE
8208
8209 034714 062716 000006      145$: ADD #6,(SP)   ;MOVE SP TO USERS ERROR CALL
8210 034720 113776 034712 000000      MOVB 140$,2(SP) ;MOVE ERROR TO USERS CALL
8211 034726 000137 035112      JMP 215$
8212
8213 034732 004737 047746      150$: JSR PC,RCLSTS  ;GO VERIFY RECALIBRATE STATUS
8214 034736 000412      BR 165$        ;RETURN HERE IF NO ERROR
8215 034740 000401      BR 160$        ;GET AROUND ERROR #
8216 034742 000000      155$: .WORD          ;ERROR # GOES HERE
8217
8218 034744 005726      160$: TST (SP)+    ;STRIP RETURN TO SUBROUTINE
    
```

```

8219 034746 062716 000006          ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
8220 034752 113776 034742 000000    MOVB  155$,2(SP)       ;MOVE ERROR # TO USERS CALL
8221 034760 000137 035112          JMP    215$
8222
8223 034764          165$:
8224
8225          ;READ THE SECTOR IDENTIFIED BY RMDAO, INCLUDING HEADER AND DATA
8226
8227 034764 012737 000073 001376    MOV    #RH!GO,RMCS10   ;LOAD READ HEADER AND DATA COMMAND
8228 034772 004737 037262          JSR    PC,PUT          ;PUT COMMAND
8229 034776 000411          BR    180$            ;RETURN HERE IF NO ERROR
8230 035000 000401          BR    175$            ;GET AROUND ERROR #
8231 035002 000000          170$: .WORD          ;ERROR # GOES HERE
8232
8233 035004 062716 000006          175$: ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
8234 035010 113776 035002 000000    MOVB  170$,2(SP)       ;MOVE ERROR # TO USERS ERROR CALL
8235 035016 000137 035112          JMP    215$
8236
8237 035022 004737 037622          180$: JSR    PC,TIMOUT      ;WAIT FOR READ OPERATION TO COMPLETE
8238 035026 004737 037012          JSR    PC,GET          ;GO GET STATUS FOR READ OPERATION
8239 035032 000411          BR    195$            ;RETURN HERE IF NO ERROR
8240 035034 000401          BR    190$            ;GET AROUND ERROR #
8241 035036 000000          185$: .WORD          ;ERROR # GOES HERE
8242
8243 035040 062716 000006          190$: ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
8244 035044 113776 035036 000000    MOVB  185$,2(SP)       ;MOVE ERROR # TO CALL
8245 035052 000137 035112          JMP    215$
8246
8247 035056 004737 052312          195$: JSR    PC,DTASTS      ;GO VERIFY RESULTS OF READ OPERATION
8248 035062 000412          BR    210$            ;RETURN HERE IF NO ERROR
8249 035064 000401          BR    205$            ;GET AROUND ERROR #
8250 035066 000000          200$: .WORD          ;ERROR # GOES HERE
8251
8252 035070 005726          205$: TST    (SP)+        ;STRIP RETURN ADDRESS TO SUBROUTINE
8253 035072 062716 000006          ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
8254 035076 113776 035066 000000    MOVB  200$,2(SP)       ;MOVE ERROR # TO USERS CALL
8255 035104 000137 035112          JMP    215$
8256
8257 035110 000456          210$: BR    240$
8258
8259 035112          215$:
8260
8261          ;AN ERROR HAS BEEN DETECTED IN TRYING TO READ THE BAD SECTOR FILE.
8262          ;THE SECTOR WILL BE RETRIED IF POSSIBLE.
8263
8264 035112 032737 010000 001340    BIT    #MOL,RMDSI      ;IS MEDIUM ON LINE ??
8265 035120 001446          BEQ    230$            ;YES !!
8266
8267 035122 005337 036102          DEC    500$            ;DECREMENT RETRY COUNT
8268 035126 100037          BPL    225$            ;RETRY IF COUNT NOT NEG
8269
8270          ;THE RETRY COUNT HAS EXPIRED - SEE IF THE ERROR IS MEDIA RELATED
8271 035130 013746 001342          MOV    RMER1I,-(SP)     ;GET ER1
8272 035134 042716 100720          BIC    #DCK!HCRC!HCE!FER!ECH,(SP)
8273 035140 032737 000010 001370    BIT    #DPE,RMER2I     ;WAS THER A DATA PARITY ERROR ??
8274 035146 001402          BEQ    220$            ;NO !!

```

8275 035150 042716 000010
 8276 035154 005726
 8277 035156 001027
 8278 035160 013746 001370
 8279 035164 042726 100010
 8280 035170 001022

```

220$: BIC #PAR,(SP) ;YES - CLEAR PAR
      TST (SP)+ ;ARE THERE ANY ERRORS NOT DUE TO MEDIA ?
      BNE 230$ ;YES !!
      MOV RMR2I,-(SP) ;GET ER2
      BIC #BSE!DPE,(SP)+ ;CLEAR MEDIA ERRORS
      BNE 230$ ;BRANCH IF NONMEDIA ERRORS DETECTED
    
```

8281
 8282
 8283
 8284
 8285

```

;THE ERRORS DETECTED WHILE TRYING TO RECOVER THE BAD SECTOR FILE ARE
;DUE TO THE MEDIA. SEE IF THE BAD SECTOR FILE CAN BE RECOVERED FROM
;ANOTHER AREA ON THE LAST TRACK
    
```

8286 035172 062737 000002 001404
 8287 035200 122737 000012 001404
 8288 035206 001413
 8289 035210 122737 000040 001404
 8290 035216 001407
 8291 035220 012737 000003 036102
 8292 035226 162716 000006
 8293 035232 000137 034360

```

      ADD #2,RMDAO ;ADVANCE SECTOR ADDRESS
      CMPB #10.,RMDAO ;QUIT IF ALL MFG SECTORS HAVE BEEN
      BEQ 230$ ;TRIED
      CMPB #32.,RMDAO ;QUIT IF ALL USER SECTORS HAVE
      BEQ 230$ ;BEEN TRIED
      MOV #3,500$ ;REINSTATE RETRY COUNT FOR THIS SECTOR
225$: SUB #6,(SP) ;MOVE SP BACK TO NO ERROR
      JMP 30$ ;RETRY THE READ OPERATION
    
```

8294
 8295 035236
 8296
 8297
 8298 035236 162716 000004
 8299 035242 000137 036076

```

230$:
;THE BAD SECTOR FILE CANNOT BE READ
      SUB #4,(SP) ;MOVE SP TO ERROR RETURN
      JMP 410$ ;GO TO MODULE EXIT
    
```

8300
 8301 035246
 8302
 8303
 8304
 8305
 8306

```

240$:
;THE SECTOR WAS RECOVERED WITHOUT ERROR -
;READ THE USER FILE IF THIS IS THE MFG FILE
;OR ELSE DONE
    
```

8307 035246 122737 000011 001404
 8308 035254 103451
 8309
 8310
 8311

```

      CMPB #9.,RMDAO ;WAS THE USER FILE READ ??
      BLO 260$ ;YES - READ IS COMPLETE
    
```

8312 035256 032737 140000 100554
 8313 035264 001410
 8314
 8315 035266 012737 002012 001404
 8316 035274 012737 000003 036102
 8317 035302 000137 034360
 8318 035306
 8319

```

;IF 144 IS IMPLEMENTED THEN READ THE USER FILE -
;ELSE DUMMY THE BAD SECTOR FILE
      BIT #MSE!USE,MFGFIL ;ARE THE BAD SECTOR FLAGS OFF ??
      BEQ 250$ ;YES - 144 IS DISABLED
    
```

8320
 8321 035306 010046
 8322 035310 010146
 8323 035312 012701 000374
 8324 035316 012700 000014
 8325 035322 012760 177777 100554
 8326 035330 012760 177777 101564
 8327 035336 062700 000002
 8328 035342 005301
 8329 035344 001366
 8330 035346 012701 000006

```

      MOV #002012,RMDAO ;READ THE USER FILE
      MOV #3,500$ ;RELOAD THE RETRY COUNT FOR THIS SECTOR
      JMP 30$ ;GO READ THE USER FILE
250$:
;DUMMY THE BAD SECTOR FILES
    
```

```

      MOV R0,-(SP) ;PUSH R0 ON STACK
      MOV R1,-(SP) ;PUSH R1 ON STACK
      MOV #252.,R1 ;R1 = NUMBER OF ENTRIES IN FILES
      MOV #14,R0 ;R0 = ADDRESS INDEX TO FILE STORAGE
255$: MOV #-1,MFGFIL(R0) ;ENTER ALL ONES IN MFG FILE
      MOV #-1,USRFIL(R0) ;ENTER ALL ONES IN USER FILE
      ADD #2,R0 ;ADVANCE ADDRESS
      DEC R1 ;DECREMENT COUNT
      BNE 255$ ;CONTINUE IF NOT DONE
      MOV #6.,R1 ;CLEAR ID, SERIAL NUMBER ETC
    
```

```

8331 035352 005000
8332 035354 005060 100554
8333 035360 005060 101564
8334 035364 062700 000002
8335 035370 005301
8336 035372 001370
8337
8338 035374 012601
8339 035376 012600
8340 035400
8341
8342
8343 035400 012737 177777 001472
8344
8345 035406 010046
8346 035410 005000
8347 035412 016060 102574 001376
8348 035420 062700 000002
8349 035424 022700 000046
8350 035430 103370
8351
8352 035432 012600
8353 035434
8354
8355 035434
8356
8357
8358
8359
8360
8361
8362
8363 035434 010046
8364 035436 010146
8365 035440 010246
8366 035442 013737 001432 001474
8367 035450 013737 001404 001476
8368 035456 005002
8369 035460 013700 001400
8370 035464 005300
8371 035466 005100
8372 035470 012701 000400
8373 035474 032737 000002 001376
8374 035502 001402
8375 035504 012701 000402
8376 035510 020100
8377 035512 101404
8378 035514 005700
8379 035516 001405
8380 035520 005202
8381 035522 000403
8382 035524 160100
8383 035526 005202
8384 035530 000767
8385 035532 010237 036102
8386

```

```

256$: CLR RO
      CLR MFGFIL(RO)
      CLR USRFIL(RO)
      ADD #2,RO
      DEC R1
      BNE 256$

      MOV (SP)+,R1      ;;POP STACK INTO R1
      MOV (SP)+,RO      ;;POP STACK INTO RO

260$:
;SET MEDIA ENABLE AND RESTORE THE USERS PUT BUFFER
      MOV #-1,MEDENB

      MOV RO,-(SP)      ;;PUSH RO ON STACK
      CLR RO            ;;RO IS REGISTER INDEX
265$: MOV BUFFER(RO),PUTBUF(RO)
      ADD #2,RO          ;;ADVANCE RO
      CMP #46,RO         ;;DONE ??
      BHIS 265$

      MOV (SP)+,RO      ;;POP STACK INTO RO

270$:
300$:
;*****
;VERIFY THAT THE DESIRED SECTOR IS NOT IN THE MFG BAD SECTOR FILE
;AND NOT IN THE USERS BAD SECTOR FILE. ASSIGN A NEW SECTOR IF THE
;SECTOR IS IN EITHER FILE.

;LOAD INITIAL VARIABLES AND COMPUTE THE NUMBER OF SECTORS
      MOV RO,-(SP)      ;;PUSH RO ON STACK
      MOV R1,-(SP)      ;;PUSH R1 ON STACK
      MOV R2,-(SP)      ;;PUSH R2 ON STACK
      MOV RMDCO,ASNDC   ;;LOAD REQUESTED CYLINDER, TRACK,
      MOV RMDAO,ASNDA   ;;AND SECTOR ADDRESS IN ASSIGNED STORAGE
      CLR R2             ;;R2 = NUMBER OF SECTORS
      MOV RMWCO,RO       ;;RO = WORD COUNT
      DEC RO             ;;CONVERT FROM 2'S COMPLEMENT

      MOV #256,R1        ;;R1 = NUMBER OF WORDS PER SECTOR
      BIT #BIT1,RMCS10  ;;IS THIS A HEADER AND DATA COMMAND ??
      BEQ 305$          ;;NO !!
      MOV #258,R1        ;;CHANGE WORDS PER SECTOR
      CMP R1,RO          ;;IS THERE A FULL SECTOR ??
      BLOS 310$        ;;YES !!
      TST RO             ;;IS RO ZERO ??
      BEQ 315$          ;;YES !!
      INC R2             ;;INCREMENT FOR PARTIAL SECTOR

      SUB R1,RO          ;;SUBTRACT ONE SECTOR FROM WORD COUNT
      INC R2             ;;INCREMENT SECTOR COUNT

315$: MOV R2,500$      ;;SAVE SECTOR COUNT

```



```

8387 035536
8388
8389
8390
8391
8392
8393 035536 012737 100570 036112      MOV      #MFGFIL+14,540$ ;MOVE THE STARTING ADDRESS OF MFG
8394                                     ;FILE TO BASE ADDRESS STORAGE
8395 035544 013737 001474 036106 320$: MOV      ASNDC,520$ ;LOAD COMPARING CYLINDER ADDRESS
8396 035552 013737 001476 036110      MOV      ASNDA,530$ ;LOAD COMPARING TRACK, SECTOR ADDRESS
8397 035560 013737 036102 036104      MOV      500$,510$ ;LOAD NUMBER OF SECTORS TO CONFIRM
8398
8399                                     ;SETUP FOR A BINARY SEARCH OF THE CURRENT FILE FOR THE COMPARING
8400                                     ;CYLINDER, TRACK AND SECTOR ADDRESS
8401 035566 013700 036112      325$: MOV      540$,R0 ;LOAD THE BASE ADDRESS IN R0
8402 035572 012701 000376      MOV      8<<(127.*4)/2>,R1 ;R1 = LENGTH OF FILE
8403 035576 060100      ADD      R1,R0 ;START BINARY DIVIDE AT CENTER
8404 035600 021037 036106      330$: CMP      (R0),520$ ;DOES TABLE ENTRY = COMPARING CYLINDER ?
8405 035604 001405      BEQ      345$ ;YES !!
8406 035606 101002      BHI      340$ ;BRANCH IF ENTRY > COMPARING CYLINDER
8407 035610
8408
8409                                     ;FILE ENTRY IS LESS THAN COMPARING CYLINDER (OR TRACK AND SECTOR),
8410                                     ;SO ADD DISPLACEMENT TO CURRENT POSITION
8411 035610 060100      ADD      R1,R0
8412 035612 000410      BR      350$
8413 035614
8414
8415                                     ;FILE ENTRY IS GREATER THAN COMPARING CYLINDER SO SUBTRACT DISPLACEMENT
8416                                     ;FROM CURRENT POSITION
8417 035614 160100      SUB      R1,R0
8418 035616 000406      BR      350$
8419 035620
8420
8421                                     ;FILE ENTRY EQUALS COMPARING CYLINDER. SEE IF THE NEXT ENTRY EQUALS
8422                                     ;THE COMPARING TRACK, AND SECTOR.
8423 035620 026037 000002 036110      340$: CMP      2(R0),530$ ;ARE THEY EQUAL ??
8424 035626 001413      BEQ      360$ ;YES !!
8425 035630 101371      BHI      340$ ;BRANCH IF HIGHER
8426 035632 000766      BR      335$ ;BRANCH IF LOWER
8427 035634
8428
8429                                     ;THE POINTER (R0) HAS BEEN ADJUSTED. HALVE THE DISPLACEMENT AND
8430                                     ;CONTINUE THE SEARCH IF DISPLACEMENT NOT ZERO
8431 035634 005701      TST      R1 ;IS DISPLACEMENT ZERO ??
8432 035636 001440      BEQ      370$ ;YES !!
8433 035640 006201      ASR      R1 ;HALVE THE DISPLACEMENT
8434 035642 042701 000003      BIC      #3,R1
8435 035646 020037 036112      CMP      R0,540$ ;IS THE NEW POINTER WITHIN BOUNDS ??
8436 035652 103432      BLO      370$ ;NO !!
8437 035654 000751      BR      330$ ;CONTINUE SEARCH
8438 035656
8439
8440                                     ;THE COMPARING CYLINDER, TRACK AND SECTOR IS IN THE BAD SECTOR FILE.
8441                                     ;ADVANCE THE ASSIGNED SECTOR AND START THE SEARCH ALL OVER.
8442 035656 105237 001476      INCB     ASNDA ;INCREMENT SECTOR

```

```

0443 035662 122737 000037 001476      CMPB  #31,ASNDA      ;SECTOR OK ??
0444 035670 103022                BHIS  365$          ;YES !!
0445 035672 105037 001476      CLRB  ASNDA         ;CLEAR SECTOR AND ADVANCE TRACK
0446 035676 105237 001477      INCB  ASNDA+1
0447 035702 122737 000004 001477      CMPB  #4,ASNDA+1   ;TRACK OK ??
0448 035710 103012                BHIS  365$          ;YES !!
0449 035712 005037 001476      CLR  ASNDA         ;CLEAR TRACK AND SECTOR
0450 035716 005237 001474      INC  ASNDC         ;INCREMENT CYLINDER
0451 035722 022737 001466 001474      CMP  #822.,ASNDC   ;CYLINDER OK ??
0452 035730 103002                BHIS  365$          ;YES !!
0453 035732 005037 001474      CLR  ASNDC
0454 035736 000677                BR    316$         ;REPEAT SEARCH
0455
0456 035740                370$:
0457
0458 ;THE COMPARING SECTOR IS NOT IN THE BAD SECTOR FILES. DECREMENT THE
0459 ;NUMBER OF SECTORS TO COMPARE AND SEARCH THE NEXT SECTOR IF THE NUMBER
0460 ;IS NOT ZERO.
0461
0462 035740 005337 036104      DEC  510$          ;DECREMENT NUMBER OF SECTORS TO COMPARE
0463 035744 001432                BEQ  380$          ;DONE IF ZERO
0464 035746 105237 036110      INCB  530$         ;INCREMENT THE COMPARING SECTOR
0465 035752 122737 000037 036110      CMPB  #31.,530$    ;SECTOR OK ??
0466 035760 103022                BHIS  375$          ;YES !!
0467 035762 105037 036110      CLRB  530$         ;CLEAR SECTOR
0468 035766 105237 036111      INCB  530$+1       ;INCREMENT TRACK
0469 035772 122737 000004 036111      CMPB  #4,530$+1    ;TRACK OK ??
0470 036000 103012                BHIS  375$          ;YES !!
0471 036002 005037 036110      CLR  530$         ;CLEAR SECTOR TRACK
0472 036006 005237 036106      INC  520$         ;INCREMENT CYLINDER
0473 036012 022737 001466 036106      CMP  #822.,520$    ;CYLINDER OK ??
0474 036020 103002                BHIS  375$          ;YES !!
0475 036022 005037 036106      CLR  520$         ;CLEAR CYLINDER
0476 036026 000137 035566      JMP  325$         ;SEARCH NEXT SECTOR
0477
0478 036032                380$:
0479
0480 ;THE ASSIGNED SECTOR (AND REQUIRED ADJACENT SECTORS) ARE NOT IN
0481 ;THE BAD SECTOR FILE JUST SEARCHED. SEARCH THE USER FILE IF IT
0482 ;HAS NOT YET BEEN SEARCHED, ELSE ASSIGN THE SECTOR AND RETURN TO USER.
0483
0484 036032 022737 101600 036112      CMP  #USRFIL+14,540$ ;TEST BASE ADDRESS
0485 036040 001405                BEQ  400$          ;DONE IF BASE = USER
0486 036042 012737 101600 036112      MOV  #USRFIL+14,540$ ;LOAD BASE ADDRESS
0487 036050 000137 035544      JMP  320$          ;SEARCH USER FILE
0488
0489
0490 036054                400$:
0491
0492 ;ASSIGN THE SECTOR AND RETURN TO USER
0493 036054 013737 001474 001432      MOV  ASNDC,RMDCO   ;LOAD CYLINDER
0494 036062 013737 001476 001404      MOV  ASNDA,RMDAO   ;LOAD TRACK AND SECTOR
0495 036070 012602                MOV  (SP)+,R2      ;POP STACK INTO R2
0496 036072 012601                MOV  (SP)+,R1      ;POP STACK INTO R1
0497 036074 012600                MOV  (SP)+,R0      ;POP STACK INTO R0
0498 036076 000240                410$: NOP

```

8499	036100	000207
8500		
8501		
8502		
8503	036102	000000
8504	036104	000000
8505	036106	000000
8506	036110	000000
8507	036112	000000

RTS PC

;THE FOLLOWING ARE STORAGE LOCATIONS FOR THE MODULE

500\$:	.WORD	;RETRY COUNT/ NUMBER OF SECTORS REQUIRED
510\$:	.WORD	;NUMBER OF SECTORS TO COMPARE
520\$:	.WORD	;COMPARING CYLINDER
530\$:	.WORD	;COMPARING TRACK AND SECTOR
540\$:	.WORD	;BASE ADDRESS OF BAD SECTOR FILE BEING SEARCHED

.SBTTL BUFFER GENERATOR SUBROUTINE

; THIS SUBROUTINE GENERATES A DATA BUFFER FOR WRITE COMMANDS. THE
 ; BUFFER STARTS AT RMB0 AND IS RMC WORDS LONG. THE BUFFER
 ; CONTAINS A REPETITIVE DATA PATTERN CONSISTING OF STMP1 WORDS
 ; FROM THE DATA PATTERN TABLE, BEGINNING AT ADDRESS STMP0.
 ; HEADER INFORMATION FOR THE BUFFER IS EXTRACTED FROM RMD0,
 ; RMDA AND RMOF.

;CALL:
 ;(1) JSR PC,GENBUF
 ;(2) ?? RETURN HERE

GENBUF:

8508										
8509										
8510										
8511										
8512										
8513										
8514										
8515										
8516										
8517										
8518										
8519										
8520										
8521	036114									
8522	036114	010046								
8523	036116	010146								
8524	036120	010246								
8525	036122	010346								
8526	036124	010446								
8527	036126	013700	001402							
8528	036132	013701	001400							
8529	036136	013737	001432	036354						
8530	036144	013737	001404	036356						
8531	036152	032737	000002	001376	10\$:					
8532	036160	001450								
8533	036162	013710	036354							
8534	036166	042710	176000							
8535	036172	052710	140000							
8536	036176	012702	000035							
8537	036202	032737	010000	001430						
8538	036210	001404								
8539	036212	052710	010000							
8540	036216	012702	000037							
8541	036222	005201			15\$:					
8542	036224	001444								
8543	036226	062700	000002							
8544	036232	013720	036356							
8545	036236	005201								
8546	036240	001436								
8547	036242	012703	036356							
8548	036246	105213								
8549	036250	120213								
8550	036252	103013								
8551	036254	105013								
8552	036256	105263	000001							
8553	036262	122763	000004	000001						
8554	036270	103004								
8555	036272	105063	000001							
8556	036276	105237	036354							
8557	036302	012704	000400		25\$:					
8558	036306	013702	001174		30\$:					
8559	036312	013703	001176							
8560	036316	012220			40\$:					
8561	036320	005201								
8562	036322	001405								
8563	036324	005304								

8564	036326	001711	BEQ	10\$;	START NEXT SECTOR IF 0
8565	036330	005303	DEC	R3	;	DECREMENT PATTERN COUNT
8566	036332	001765	BEQ	30\$;	RESTART PATTERN IF 0
8567	036334	000770	BR	40\$;	CONTINUE DATA PATTERN
8568						
8569	036336		50\$:			
8570	036336	012604	MOV	(SP)+,R4	;;	POP STACK INTO R4
8571	036340	012603	MOV	(SP)+,R3	;;	POP STACK INTO R3
8572	036342	012602	MOV	(SP)+,R2	;;	POP STACK INTO R2
8573	036344	012601	MOV	(SP)+,R1	;;	POP STACK INTO R1
8574	036346	012600	MOV	(SP)+,R0	;;	POP STACK INTO R0
8575	036350	000240	NOP			
8576	036352	000207	RTS	PC		
8577						
8578	036354	000000	60\$:	.WORD	;	CYLINDER ADDRESS STORAGE
8579	036356	000000	65\$:	.WORD	;	TRACK, SECTOR ADDRESS STORAGE

8580
8581
8582
8583
8584
8585
8586
8587
8588
8589
8590
8591
8592
8593
8594
8595
8596 036360
8597 036360 010046
8598 036362 010146
8599 036364 010246
8600 036366 010346
8601
8602 036370 005037 036724
8603
8604 036374 032737 100000 001342
8605 036402 001465
8606 036404 032737 000100 001342
8607 036412 001061
8608 036414 033737 004000 001360
8609 036422 001055
8610 036424 032737 010000 001360
8611 036432 001451
8612
8613
8614 036434 013700 001402
8615 036440 013701 001372
8616 036444 052737 100000 036724
8617
8618
8619 036452 022701 000020
8620 036456 103005
8621 036460 162701 000020
8622 036464 062700 000002
8623 036470 000770
8624 036472 012702 000001
8625 036476 010203
8626
8627
8628 036500 020301
8629 036502 001403
8630 036504 006302
8631 036506 005203
8632 036510 000773
8633 036512 012703 000013
8634
8635

```
.SBTTL COMPARE BUFFER SUBROUTINE

; THIS SUBROUTINE COMPARES THE CONTENTS OF BUFONE AND BUFTWO,
; ASSUMING THAT BUFONE IS THE BUFFER FROM WHICH DATA WAS WRITTEN
; AND BUFTWO IS THE BUFFER TO WHICH DATA WAS READ. ERRORS IN BUFFER
; COMPARISON ARE REPORTED TO THE USER VIA THE USER'S ERROR CALL.

;CALL:
;(1) JSR PC,CMPBUF
;(2) .WORD WRITE BUFFER ADDRESS
;(3) .WORD READ BUFFER ADDRESS
;(4) BR ?? RETURN HERE IF NO ERROR
;(5) NOP RETURN HERE IF ERROR
;(6) ERROR ERROR DEFINED BY SUBROUTINE
;(7) ???

CMPBUF:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK

CLR 150$ ;CLEAR CORRECTION FLAG
;DETERMINE IF DATA SHOULD BE CORRECTED
BIT #DCK,RMER11 ;WAS THERE A DATA CHECK ??
BEQ 80$ ;NO !!
BIT #ECH,RMER11 ;IS IT A HARD ERROR ??
BNE 80$ ;YES !!
BIT ECI,RMOFI ;WAS ECC CORRECTION ALLOWED ??
BNE 80$ ;NO !!
BIT #FMT16,RMOFI ;IS THIS 16 BIT FORMAT ??
BEQ 80$ ;NO !!

;CORRECT DATA USING ECC INFORMATION
MOV RMBAO,R0 ;R0 = STARTING BUFFER ADDRESS
MOV RMECI,R1 ;R1 = ECC POSITION
BIS #BIT15,150$ ;SET CORRECTION FLAG
;MOVE R0 TO WORD BOUNDARY OF ERROR BURST
10$: CMP #16.,R1 ;IS BIT POSITION > 1 WORD
BHIS 20$ ;NO !!
SUB #16.,R1 ;SUBTRACT 1 WORDS WORTH
ADD #2,R0 ;ADVANCE BUFFER ADDRESS 1 WORD
BR 10$
20$: MOV #1,R2 ;R2 = BIT POINTER
MOV R2,R3 ;R3 = BIT NUMBER
;MOVE R2 TO STARTING BIT OF ERROR BURST
30$: CMP R3,R1 ;IS R3 SAME AS R1 ??
BEQ 35$ ;YES !!
ASL R2 ;SHIFT BIT POINTER
INC R3 ;INCREMENT BIT NUMBER
BR 30$
35$: MOV #11.,R3 ;R3 = LENGTH OF ERROR BURST

;CORRECT THE ERROR BURST
```

```

8636 036516 030237 001374      40$: BIT R2,RMEC2I ;IS THIS BIT-SET IN ECC PATTERN ??
8637 036522 001405          BEQ 60$ ;NO - DO NOT CORRECT THIS BIT
8638 036524 030210          BIT R2,(R0) ;IS THE BIT PRESENTLY SET ??
8639 036526 001402          BEQ 50$ ;NO
8640 036530 040210          BIC R2,(R0) ;RESET THE BIT
8641 036532 000401          BR 60$
8642 036534 050210      50$: BIS R2,(R0) ;SET THE BIT
8643 036536 006302      60$: ASL R2 ;SHIFT TO NEXT BIT
8644 036540 001004          BNE 70$
8645 036542 012702 000001      MOV #1,R2 ;CONTINUE WITH FIRST BIT OF NEXT WORD
8646 036546 062700 000002      ADD #2,R0
8647 036552 005303      70$: DEC R3 ;END OF BURST ??
8648 036554 001360          BNE 40$ ;NO !!
8649 036556 017600 000010      80$: MOV #10(SP),R0 ;RO = WRITE BUFFER
8650 036562 062766 000002 000010      ADD #2,10(SP) ;MOVE SP TO READ ADDRESS
8651 036570 017601 000010      MOV #10(SP),R1 ;R1 = READ BUFFER
8652 036574 062766 000002 000010      ADD #2,10(SP) ;MOVE SP TO RETURN ADDRESS
8653 036602 C13702 001330      MOV RMWCI,R2 ;R2 = NUMBER OF WORDS TRANSFER
8654 036606 163702 001400      SUB RMWCO,R2
8655 036612 022021      90$: CMP (R0)+,(R1)+ ;COMPARE DATA WORDS
8656 036614 001003          BNE 100$ ;EXIT IF NOT EQUAL
8657 036616 005302          DEC R2 ;DECREMENT WORD COUNT
8658 036620 001374          BNE 90$ ;CONTINUE IF NOT DONE
8659 036622 000433          BR 110$ ;DONE COMPARE - NO ERROR
8660 036624
8661 036624 014037 001140      100$: MOV -(R0),SGDDAT ;STORE GOOD DATA FOR TYPEOUT
8662 036630 014137 001142      MOV -(R1),SBDDAT ;STORE BAD DATA FOR TYPEOUT
8663 036634 010037 001134      MOV RO,SGDADR ;STORE ADDRESS OF GOOD DATA
8664 036640 010137 001136      MOV R1,SBADR ;STORE ADDRESS OF BAD DATA
8665 036644 010237 001174      MOV R2,STMPD ;STORE WORD COUNT OF ERROR
8666 036650 062766 000004 000010      ADD #4,10(SP) ;MOVE SP TO USER'S ERROR CALL
8667 036656 112776 000336 000010      MOVB #336,#10(SP) ;WRITE ERROR NUMBER IN CALL
8668
8669 ;CHANGE ERROR NUMBER IF ECC CORRECTION FAILED
8670 036664 032737 100000 036724      BIT #BIT15,150$ ;WAS ECC CORRECTION USED ??
8671 036672 001403          BEQ 105$ ;NO !!
8672 036674 112776 000163 000010      MOVB #163,#10(SP) ;ECC CORRECTION FAILED
8673 036702 162766 000002 000010      105$: SUB #2,10(SP) ;MOVE SP TO RETURN IF ERROR
8674 036710 000240          NOP
8675 036712
8676 036712 012603      110$: MOV (SP)+,R3 ;POP STACK INTO R3
8677 036714 012602      MOV (SP)+,R2 ;POP STACK INTO R2
8678 036716 012601      MOV (SP)+,R1 ;POP STACK INTO R1
8679 036720 012600      MOV (SP)+,R0 ;POP STACK INTO R0
8680 036722 000207      RTS PC ;RETURN TO USER
8681
8682 036724 000000      150$: .WORD ;ECC CORRECTION FLAG

```

```

0683 .SBTTL GET STATUS SUBROUTINE
0684
0685 ;THIS SUBROUTINE SETS UP THE "GET INDEX TABLE" AND THE "GET
0686 ;BUFFER" FOR READING ALL SUBSYSTEM REGISTERS VIA THE GET SUBROUTINE
0687 ;AND THEN RETURNS TO THE USER.
0688
0689 ;CALL: JSR PC,GETSTS
0690 ;      ??? RETURN HERE
0691
0692 GETSTS:
0693 036726 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
0694 036726 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
0695 036732 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
0696 036734 012700 001504 MOV #GETINX,RO ;RO= ADDRESS OF INDEX TABLE
0697 036740 012701 001376 MOV #RMEC2I+2,R1 ;R1 = ADDRESS OF GET BUFFER
0698 036744 012702 000046 MOV #RMEC2,R2 ;R2 = REGISTER INDE
0699 036750 110220 2$: MOV R2,(RO)+ ;WRITE REGISTER INDEX IN TABLE
0700 036752 005041 CLR -(R1) ;CLEAR CORRESPONDING LOCATION
0701 036754 162702 000002 3$: SUB #2,R2 ;DECREMENT TO NEXT INDEX
0702 036760 100405 BMI 4$ ;BRANCH OUT IF DONE
0703 036762 022702 000022 CMP #RMDB,R2 ;DONT WRITE RMDB INDEX
0704 036766 001370 BNE 2$
0705 036770 005041 CLR -(R1)
0706 036772 000770 BR 3$
0707 036774 112720 000200 4$: MOV R2,(RO)+ ;WRITE TERMINATOR
0708 037000 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
0709 037002 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
0710 037004 012600 MOV (SP)+,RO ;;POP STACK INTO RO
0711 037006 000240 NOP
0712 037010 000207 RTS PC
0713
    
```


8714
8715
8716
8717
8718
8719
8720
8721
8722
8723
8724
8725
8726
8727
8728
8729
8730
8731
8732
8733
8734
8735
8736
8737
8738 037012 000240
8739 037014 062716 000004
8740 037020 105076 000000
8741 037024 162716 000004
8742 037030 010046
8743 037032 010146
8744 037034 010246
8745 037036 010346
8746 037040 010446
8747 037042 013746 000004
8748 037046 013746 000006
8749 037052 013700 001276
8750 037056 012702 001326
8751 037062 012704 001504
8752 037066 012737 037174 000004
8753 037074 012737 000300 000006
8754 037102 016037 000010 001174
8755 037110 016037 000000 001176
8756 037116 032737 004000 001176
8757 037124 001007
8758 037126 062766 000004 000016
8759 037134 112776 000112 000016
8760 037142 000423
8761 037144 105714
8762 037146 100433
8763 037150 111401
8764 037152 042701 177700
8765 037156 060001
8766 037160 112403
8767 037162 042703 177700
8768 037166 060203
8769 037170 011113

.SBTTL GET SUBROUTINE

: THIS SUBROUTINE READS THE REGISTERS WHICH ARE LISTED IN THE
: "GET INDEX TABLE" AND STORES THEIR VALUES IN THE CORRESPONDING
: LOCATION IN THE "GET REGISTER BUFFER". FOR EXAMPLE, AN
: ENTRY OF 04 IN THE TABLE WILL CAUSE THE SUBROUTINE TO
: READ "R04" AND STORE ITS CONTENTS AT THE LOCATION IN
: THE BUFFER ASSIGNED TO THAT REGISTER. THE NUMBER OF
: REGISTERS TO BE READ IS VARIABLE FROM 1 TO 22; THE INDEX
: TABLE MUST BE TERMINATED WITH A CONTROL BYTE (200)
: WHICH SHOULD FOLLOW THE LAST ENTRY.

SUBROUTINE CALL:

- (1) "GET INDEX TABLE" HAS BEEN LOADED WITH REGISTER INDEX VALUES AND TERMINATED WITH A CONTROL BYTE
- (2) "GET INPUT BUFFER" IS AVAILABLE FOR USE. (NOTE THAT UNUSED LOCATIONS, I.E., ENTRIES IN BUFFER CORRESPONDING TO REGISTERS NOT READ, ARE NOT CHANGED.)
- (3) JSR PC,GET
BR ??? RETURN HERE IF NO ERROR FOUND
NOP RETURN HERE IF ANY ERROR FOUND
ERROR SUB DEFINES ERROR NUMBER
???

GET: NOP
ADD #4,(SP) ;CLEAR ERROR NUMBER IN USER'S
CLR 2(SP) ;ERROR CALL
SUB #4,(SP)
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
MOV \$BASE,R0
MOV #GETBUF,R2
MOV #GETINX,R4
MOV #5\$,ERRVEC ;SETUP FOR TIMEOUT
MOV #PR6,ERRVEC+2
1\$: MOV RMCS2(R0),\$TMP0 ;GET "NED" STATUS
MOV RMCS1(R0),\$TMP1 ;GET "DVA" STATUS
BIT #DVA,\$TMP1 ;DEVICE AVAILABLE??
BNE 3\$;YES!!
ADD #4,16(SP) ;WRITE ERROR NUMBER IN USER'S
MOVB #112,216(SP) ;ERROR CALL
BR 7\$
3\$: TSTB (R4) ;DONE??
BMI 9\$;YES!!
MOVB (R4),R1 ;R1 = REGISTER ADDRESS
BIC #+CIDXMSK,R1 ;CLEAR ANY SIGN EXTENSION
ADD R0,R1
MOVB (R4)+,R3 ;R3 = STORAGE ADDRESS FOR REGISTER
BIC #+CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
ADD R2,R3
MOV (R1),(R3) ;READ REGISTER

```

8770 037172 000764          BR      3$
8771
8772 037174 022626          5$:   CMP      (SP)+,(SP)+      ;RESTORE STACK
8773 037176 062766 000004 000016  ADD      #4,16(SP)      ;WRITE ERROR NUMBER IN
8774 037204 112776 000007 000016  MOVVB   #7,216(SP)      ;USER'S ERROR CALL
8775 037212 162766 000002 000016  7$:   SUB      #2,16(SP)
8776 037220 105714          8$:   TSTB   (R4)          ;DONE CLEARING??
8777 037222 100405          BMI     9$             ;YES!!
8778 037224 005003          CLR     R3            ;CLEAR REMAINING STORAGE
8779 037226 112403          MOVVB  (R4)+,R3       ;LOCATIONS
8780 037230 060203          ADD     R2,R3
8781 037232 005013          CLR     (R3)
8782 037234 000771          BR      8$
8783 037236
8784 037236 012637 000006          9$:   MOV     (SP)+,ERRVEC+2  ;:POP STACK INTO ERRVEC+2
8785 037242 012637 000004          MOV     (SP)+,ERRVEC    ;:POP STACK INTO ERRVEC
8786 037246 012604          MOV     (SP)+,R4        ;:POP STACK INTO R4
8787 037250 012603          MOV     (SP)+,R3        ;:POP STACK INTO R3
8788 037252 012602          MOV     (SP)+,R2        ;:POP STACK INTO R2
8789 037254 012601          MOV     (SP)+,R1        ;:POP STACK INTO R1
8790 037256 012600          MOV     (SP)+,R0        ;:POP STACK INTO R0
8791 037260 000207          RTS     PC             ;RETURN
8792
    
```

8793
8794
8795
8796
8797
8798
8799
8800
8801
8802
8803
8804
8805
8806
8807
8808
8809
8810
8811
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
8825
8826
8827
8828
8829
8830
8831
8832
8833
8834
8835
8836
8837
8838
8839
8840
8841
8842
8843
8844
8845
8846
8847
8848

037262 000240
037264 010046
037266 010146
037270 010246
037272 010346
037274 010446
037276 013746 000004
037302 013746 000006
037306 013700 001276
037312 012702 001376
037316 012704 001533
037322 012737 037430 000004
037330 012737 000300 000006
037336 016037 000010 001174
037344 016037 000000 001176
037352 032737 004000 001176
037360 001007
037362 062766 000004 000016
037370 112776 000112 000016
037376 000423
037400 105714
037402 100424
037404 111401
037406 042701 177700
037412 060001
037414 112403
037416 042703 177700
037422 060203
037424 011311
037426 000764
037430 022626
037432 062766 000004 000016
037440 112776 000007 000016
037446 162766 000002 000016

.SBTTL PUT SUBROUTINE

: THIS SUBROUTINE WRITES THE REGISTERS WHICH ARE LISTED IN THE
: "PUT INDEX TABLE" WITH THE CONTENTS OF THE CORRESPONDING
: LOCATION IN THE "PUT REGISTER BUFFER". THE NUMBER OF
: REGISTERS WRITTEN IS VARIABLE; THE INDEX TABLE MUST
: BE TERMINATED WITH A CONTROL BYTE (200) WHICH SHOULD
: FOLLOW THE LAST ENTRY.

: SUBROUTINE CALL:

- (1) "PUT INDEX TABLE" HAS BEEN LOADED WITH INDEX VALUES OF REGISTERS TO BE WRITTEN.
- (2) "PUT REGISTER BUFFER" CONTAINS CONTENTS OF EACH REGISTER TO BE WRITTEN.
- (3) JSR PC,PUT
BR ??? RETURN HERE IF NO ERROR FOUND
NOP RETURN HERE IF ANY ERROR FOUND
ERROR SUB DEFINES ERROR NUMBER
???

```

PUT:  NOP
      MOV    RD,-(SP)      ;; PUSH RD 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    ERRVEC,-(SP)  ;; PUSH ERRVEC ON STACK
      MOV    ERRVEC+2,-(SP) ;; PUSH ERRVEC+2 ON STACK
      MOV    $BASE,RD
      MOV    #PUTBUF,R2
      MOV    #PUTINX,R4
      MOV    #55,ERRVEC    ; SETUP FOR TIMEOUT
      MOV    #PR6,ERRVEC+2
1$:   MOV    RMCS2(RD),$TMP0 ; GET "NED" STATUS
      MOV    RMCS1(RD),$TMP1 ; GET "DVA" STATUS
      BIT    #DVA,$TMP1     ; DEVICE AVAILABLE??
      BNE   3$             ; YES!!
      ADD   #4,16(SP)      ; WRITE ERROR NUMBER IN
      MOVB  #112,16(SP)   ; USER'S ERROR CALL
      BR   7$
3$:   TSTB  (R4)           ; DONE??
      BMI  9$             ; YES!!
      MOVB (R4),R1        ; R1 = REGISTER ADDRESS
      BIC  #CIDXMSK,R1    ; CLEAR ANY SIGN EXTENSION
      ADD  RD,R1
      MOVB (R4)+,R3       ; R3 = STORAGE ADDRESS
      BIC  #CIDXMSK,R3    ; CLEAR ANY SIGN EXTENSION
      ADD  R2,R3
      MOV  (R3),(R1)      ; WRITE REGISTER
      BR   3$
5$:   CMP  (SP)+,(SP)+    ; ADJUST STACK
      ADD  #4,16(SP)      ; WRITE ERROR NUMBER IN
      MOVB #7,16(SP)     ; USER'S ERROR CALL
7$:   SUB  #2,16(SP)

```

8849			
8850	037454		
8851	037454	012637	000006
8852	037460	012637	000004
8853	037464	012604	
8854	037466	012603	
8855	037470	012602	
8856	037472	012601	
8857	037474	012600	
8858	037476	000207	
8859			

95:

MOV	(SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
MOV	(SP)+,ERRVEC	::POP STACK INTO ERRVEC
MOV	(SP)+,R4	::POP STACK INTO R4
MOV	(SP)+,R3	::POP STACK INTO R3
MOV	(SP)+,R2	::POP STACK INTO R2
MOV	(SP)+,R1	::POP STACK INTO R1
MOV	(SP)+,R0	::POP STACK INTO R0
RTS	PC	::RETURN

```

      .SBTTL  SIZE CLOCK SUBROUTINE
      SIZCLK:
0860      MOV      ERRVEC, -(SP)      ;; PUSH ERRVEC ON STACK
0861      MOV      ERRVEC+2, -(SP)    ;; PUSH ERRVEC+2 ON STACK
0862      MOV      #1$, ERRVEC      ; SET UP FOR BUS TIMEOUT
0863      MOV      #PR6, ERRVEC+2
0864      MOV      #177546, CLKADR   ; LOAD ADDRESSES FOR KW11-L
0865      MOV      #100, CLKVCT
0866      TST     @CLKADR            ; TEST FOR KW11-L PRESENT
0867      BR      3$                ; YES - KW11-L IS PRESENT
0868      CMP     (SP)+, (SP)+      ; RESTORE SP
0869      MOV      #2$, ERRVEC      ; SET UP FOR BUS TIMEOUT
0870      MOV      #172540, CLKADR  ; LOAD ADDRESSES FOR KW11-P CLOCK
0871      MOV      #104, CLKVCT
0872      TST     @CLKADR            ; TEST FOR KW11-P PRESENT
0873      BR      3$                ; YES - KW11-P IS PRESENT
0874      CMP     (SP)+, (SP)+      ; RESTORE SP
0875      ADD     #2, 4(SP)         ; MOVE RETURN TO ERROR
0876      MOV      (SP)+, ERRVEC+2   ;; POP STACK INTO ERRVEC+2
0877      MOV      (SP)+, ERRVEC    ;; POP STACK INTO ERRVEC
0878      RTS     PC                ; RETURN TO USER
0879
0880
0881
0882

```

```

8883 .SBTTL TIMEOUT SUBROUTINE
8884
8885 ;THIS SUBROUTINE WAITS FOR GO TO RESET OR FOR A TIMEOUT GREATER THAN
8886 ;500 MS, WHICH EVER OCCURS FIRST, AND THEN RETURNS.
8887
8888 ;CALL: JSR PC, TIMEOUT
8889 ;      ???
8890 ;      RETURN HERE
8891
8892 TIMEOUT:
8893 MOV RO, -(SP) ;: PUSH RO ON STACK
8894 MOV R1, -(SP) ;: PUSH R1 ON STACK
8895 MOV R2, -(SP) ;: PUSH R2 ON STACK
8896 MOV ERVEC, -(SP) ;: PUSH ERVEC ON STACK
8897 MOV ERVEC+2, -(SP) ;: PUSH ERVEC+2 ON STACK
8898 MOV #4$, ERVEC ;: SETUP FOR BUS TIMEOUT - 04 TRAP
8899 MOV #PR6, ERVEC+2
8900 MOV $BASE, RO ;: RO=RMO3 ADDRESS
8901 MOV CLKADR, R1 ;: R1=CLOCK ADDRESS
8902 MOV #31, R2 ;: R2=NUMBER OF CLOCK CYCLES
8903 1$: CMP R1, #172540 ;: KW11-P CLOCK??
8904 BNE 2$ ;: NO!!
8905 MOV #1, 2(R1) ;: SET COUNTER
8906 2$: MOV #BIT2:BIT0, (R1) ;: START COUNTER
8907
8908 3$: MOV RMCS1(RO), -(SP) ;: GET STATUS
8909 BIC #1<RDY!GO>, (SP)
8910 CMP #RDY, (SP)+ ;: RDY=1, GO=0??
8911 BEQ 5$ ;: YES!!
8912 BIT #BIT7, (R1) ;: TIMER DONE??
8913 BEQ 3$ ;: NO!!
8914 DEC R2 ;: DEC NUMBER OF CYCLES
8915 BNE 1$ ;: CONTINUE IF NOT DONE
8916 BR 5$
8917 4$: CMP (SP)+, (SP)+ ;: ADJUST STACK
8918 ADD #4, 12(SP) ;: MOVE SP TO USER'S CALL
8919 MOVB #7, 212(SP) ;: WRITE ERROR NUMBER
8920 SUB #2, 12(SP)
8921
8922 5$: MOV (SP)+, ERVEC+2 ;: POP STACK INTO ERVEC+2
8923 MOV (SP)+, ERVEC ;: POP STACK INTO ERVEC
8924 MOV (SP)+, R2 ;: POP STACK INTO R2
8925 MOV (SP)+, R1 ;: POP STACK INTO R1
8926 MOV (SP)+, RO ;: POP STACK INTO RO
8927 RTS PC ;: RETURN TO USER
8928

```

8929
8930
8931
8932
8933
8934
8935
8936
8937
8938
8939
8940
8941
8942
8943
8944
8945
8946
8947
8948
8949
8950
8951
8952
8953
8954
8955
8956
8957
8958
8959
8960
8961
8962
8963
8964
8965
8966
8967
8968
8969
8970
8971
8972
8973
8974
8975
8976
8977
8978
8979
8980
8981
8982
8983
8984

.SBTTL PRIMARY ERROR CHECK SUBROUTINE

THE PURPOSE OF THIS SUBROUTINE IS TO VERIFY THAT STAUTS IS VALID AND
THAT FURTHER ERROR AND STATUS CHECKING SHOULD BE PERFORMED. THE
FOLLOWING CHECKS ARE MADE:

.CURRENT UNIT IS SELECTED, I.E., THE UNIT SELECT BITS OF RMCS2
(BITS 0-2) EQUAL THE UNIT BEING TESTED;

.SELECTED UNIT IS AVAILABLE, I.E., DVA (BIT 11 OF RMCS1) IS SET
AND NED (BIT 12 OF RMCS2) IS RESET;

.LAST COMMAND WAS COMPLETED, I.E., THE MASSBUS CONTROLLER IS
READY (BIT 7 OF RMCS1) AND THE GO BIT IS RESET (BIT 0 OF RMCS1) OR THE
DRIVE READY BIT (BIT 7 OF RMD5) IS SET.

.NO PARITY ERROR OCCURRED WHEN READING REMOTE REGISTERS,
I.E., MCPE = 0.

.NO PARITY ERROR OCCURRED WHEN WRITING REMOTE REGISTERS,
I.E., PAR = 0, OR, PAR = DPE = 1

THE SUBROUTINE ASSUMES THAT:

STATUS HAS BEEN STORED IN THE REGISTER INPUT BUFFER,
IN PARTICULAR, RMCS1, RMCS2 AND RMD5 HAVE BEEN STORED IN THEIR
CORRESPONDING LOCATIONS OF THE "GET" BUFFER.

.(SUNIT) CONTAINS THE DRIVE NUMBER

THE SUBROUTINE IS CALLED AS FOLLOWS:

```
(1)  JSR    PC,PRIERR      RETURN HERE IF NO ERROR
      BR     ???           RETURN HERE TO REPORT AN ERROR
      NOP                    ERROR NUMBER DEFINED BY SUB
      ERROR                                     GO BACK TO SUB FOR MORE ERROR CHECKS
      JSR    PC,@(SP)+    RETURN HERE IF NO MORE ERRORS
      ???
```

PRIERR:

;CLEAR USER'S ERROR CALL

```
ADD    #4,(SP)      ;MOVE (SP) TO ERROR CALL
CLRB   @(SP)        ;CLEAR ERROR NUMBER
SUB    #4,(SP)      ;MOVE (SP) TO NO ERROR RETURN
```

;REPORT AN ERROR IF THE WRONG UNIT IS SELECTED

```
MOV    RMCS2I,SBDAT ;CORRECT UNIT SELECTED??
BIC    #1CUNTMSK,SBDAT
MOV    SUNIT,SGDAT  ;GOOD DATA FOR TYPEOUT
BIC    #1CUNTMSK,SGDAT
CMPB   SGDAT,SBDAT  ;COMPARE EXPECTED AND RECEIVED
                        ;DRIVE NUMBERS
                        ;YES!!
BEQ    1$
ADD    #4,(SP)
MOVB   #1,@(SP)     ;ERROR 1
```

040006

040006 062716 000004
040012 105076 000000
040016 162716 000004

040022 013737 001336 001142
040030 042737 177770 001142
040036 013737 001234 001140
040044 042737 177770 001140
040052 123737 001140 001142

040060 001415
040062 062716 000004
040066 112776 000001 000000

8985	040074	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
8986	040100	004736			JSR	PC,@(SP)+	;REPORT WRONG UNIT SELECTED
8987	040102	162716	000010		SUB	#10,(SP)	;RESTORE (SP)
8988	040106	000240			NOP		
8989	040110	000137	040630		JMP	10\$;SKIP OTHER CHECKS
8990	040114						
8991							
8992							
8993							
8994	040114	032737	004000	001326			
8995	040122	001045			BIT	#DVA,RMCS1I	;DEVICE AVAILABLE??
8996	040124	013737	001326	001140	BNE	5\$;YES!!
8997	040132	052737	004000	001140	MOV	RMCS1I,\$GDDAT	;EXPECTED STATUS
8998	040140	013737	001326	001142	BIS	#DVA,\$GDDAT	
8999	040146	062716	000004		MOV	RMCS1I,\$BDDAT	;RECEIVED STATUS
9000	040152	112776	000002	000000	ADD	#4,(SP)	
9001	040160	032737	010000	001336	MOVB	#2,@(SP)	;ERROR #2
9002	040166	001414			BIT	#NED,RMCS2I	;WAS NED SET??
9003	040170	013737	001336	001140	BEQ	2\$;NO!!
9004	040176	013737	001336	001142	MOV	RMCS2I,\$GDDAT	;EXPECTED STATUS
9005	040204	042737	010000	001140	MOV	RMCS2I,\$BDDAT	;RECEIVED STATUS
9006	040212	112776	000003	000000	BIC	#NED,\$GDDAT	
9007	040220	162716	000002		MOVB	#3,@(SP)	;YES - CHANGE ERROR NUMBER
9008	040224	004736			SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
9009	040226	162716	000010		JSR	PC,@(SP)+	;REPORT DEVICE NOT AVAILABLE
9010	040232	000240			SUB	#10,(SP)	;RESTORE (SP)
9011	040234	000575			NOP		
9012	040236				BR	10\$;SKIP OTHER CHECKS
9013							
9014							
9015	040236	032737	000200	001326			
9016	040244	001030					
9017	040246	013737	001326	001140	BIT	#RDY,RMCS1I	;CONTROLLER READY??
9018	040254	052737	000200	001140	BNE	7\$;YES!!
9019	040262	042737	160001	001140	MOV	RMCS1I,\$GDDAT	;EXPECTED STATUS
9020	040270	013737	001326	001142	BIS	#RDY,\$GDDAT	
9021	040276	062716	000004		BIC	#SC!TRE!MCPE!GO,\$GDDAT	
9022	040302	112776	000004	000000	MOV	RMCS1I,\$BDDAT	;RECEIVED STATUS
9023	040310	162716	000002		ADD	#4,(SP)	
9024	040314	004736			MOVB	#4,@(SP)	;ERROR #4
9025	040316	162716	000010		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
9026	040322	000240			JSR	PC,@(SP)+	;REPORT CONTROLLER NOT READY
9027	040324	000541			SUB	#10,(SP)	;RESTORE (SP)
9028	040326				NOP		
9029					BR	10\$;SKIP OTHER CHECKS
9030							
9031	040326	032737	000001	001326			
9032	040334	001431					
9033	040336	032737	000200	001340	BEQ	8\$;GO RESET??
9034	040344	001025			BIT	#DRY,RMDSI	;DRIVE READY??
9035	040346	013737	001326	001140	BNE	8\$;YES!!
9036	040354	042737	160001	001140	MOV	RMCS1I,\$GDDAT	;EXPECTED STATUS
9037	040362	013737	001326	001142	BIC	#SC!TRE!MCPE!GO,\$GDDAT	
9038	040370	062716	000004		MOV	RMCS1I,\$BDDAT	;RECEIVED STATUS
9039	040374	112776	000005	000000	ADD	#4,(SP)	
9040	040402	162716	000002		MOVB	#5,@(SP)	;ERROR #5
					SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR

9041	040406	004736			JSR	PC, 2(SP)+	:REPORT DRIVE NOT READY
9042	040410	162716	000010		SUB	#10, (SP)	:RESTORE (SP)
9043	040414	000240			NOP		
9044	040416	000504			BR	10\$	
9045	040420						
9046							
9047							
9048							
9049	040420	032737	020000	001326			
9050	040426	001425			BIT	#MCPE, RMCS1I	:PARITY ERROR ??
9051	040430	013737	001326	001140	BEQ	9\$:NO!!
9052	040436	042737	160001	001140	MOV	RMCS1I, \$GDDAT	:EXPECTED STATUS
9053	040444	013737	001326	001142	BIC	#SC!TR!MCPE!GO, \$GDDAT	
9054	040452	062716	000004		MOV	RMCS1I, \$BDDAT	:RECEIVED STATUS
9055	040456	112776	000013	000000	ADD	#4, (SP)	:MOVE STACK TO USER'S ERROR
9056	040464	162716	000002		MOVB	#13, 2(SP)	:ERROR #47
9057	040470	004736			SUB	#2, (SP)	:MOVE SP TO RETURN FOR ERROR
9058	040472	162716	000010		JSR	PC, 2(SP)+	:REPORT ERROR VIA USER
9059	040476	000240			SUB	#10, (SP)	:RESTORE STACK
9060	040500	000453			NOP		
9061	040502				BR	10\$	
9062							
9063							
9064	040502	032737	000010	001342			
9065	040510	001451			BIT	#PAR, RMER1I	:WAS THERE A PARITY ERROR??
9066	040512	032737	000010	001370	BEQ	11\$:NO!!
9067	040520	001045			BIT	#DPE, RMER2I	:WAS IT THE CONTROL BUS??
9068	040522	032737	000010	001412	BNE	11\$:NOT SURE!!
9069	040530	001413			BIT	#PAR, RMER10	:DID TEST SET PAR ??
9070	040532	010046			BEQ	93\$:NO!!
9071	040534	012700	001533		MOV	RO, -(SP)	:PUSH RO ON STACK
9072	040540	122710	000014		MOV	#PUTINX, RO	:RO POINTS TO INDEX TABLE
9073	040544	001002			CMPB	#RMER1, (RO)	:SEARCH TABLE FOR RMER1
9074	040546	012600			BNE	92\$	
9075	040550	000431			MOV	(SP)+, RO	:POP STACK INTO RO
9076	040552	105720			BR	11\$:PAR WAS SET BY TEST
9077	040554	100371			TSTB	(RO)+	:END OF TABLE??
9078	040556	012600			BPL	91\$:NO!!
9079	040560	013737	001342	001140	MOV	(SP)+, RO	:POP STACK INTO RO
9080	040566	042737	000010	001140	MOV	RMER1I, \$GDDAT	:EXPECTED STATUS
9081	040574	013737	001342	001142	BIC	#PAR, \$GDDAT	
9082	040602	062716	000004		MOV	RMER1I, \$BDDAT	:RECEIVED STATUS
9083	040606	112776	000050	000000	ADD	#4, (SP)	:MOVE SP TO USER'S ERROR CALL
9084	040614	162716	000002		MOVB	#50, 2(SP)	:WRITE THE ERROR NUMBER
9085	040620	004736			SUB	#2, (SP)	:MOVE SP TO RETURN FOR ERROR
9086	040622	162716	000010		JSR	PC, 2(SP)+	:REPORT THE ERROR
9087	040626	000240			SUB	#10, (SP)	:MOVE SP TO NO ERROR RETURN
9088	040630	062716	000010		NOP		
9089	040634	000240			ADD	#10, (SP)	:RETURN TO ERROR
9090	040636	000207			NOP		:RETURN TO NO ERROR
9091					RTS	PC	

```

9092
9093
9094
9095
9096
9097
9098
9099
9100
9101
9102
9103
9104
9105
9106
9107
9108
9109
9110
9111
9112
9113
9114
9115 040640
9116
9117
9118
9119 040640 013737 001376 044500
9120 040646 042737 177701 044500
9121 040654 062716 000004
9122 040660 105076 000000
9123 040664 162716 000004
9124
9125
9126
9127
9128
9129 040670 032737 000200 001340
9130 040676 001024
9131 040700 013737 001340 001142
9132 040706 042737 177577 001142
9133 040714 012737 000200 001140
9134 040722 062716 000004
9135 040726 112776 000010 000000
9136 040734 162716 000002
9137 040740 004736
9138 040742 162716 000010
9139 040746 000240
9140
9141
9142 040750 032737 000001 001326
9143 040756 001423
9144 040760 013737 001326 001142
9145 040766 042737 177776 001142
9146 040774 005037 001140
9147 041000 062716 000004

```

```

.SBTTL SECONDARY ERROR CHECK SUBROUTINE
:THE ERROR CHECK SUBROUTINE PROVIDES DETECTION OF SECONDARY ERRORS
:SUCH AS UNEXPECTED ERRORS AND UNEXPECTED REGISTER
:CONTENTS. THESE ERRORS ARE DEEMED
:SECONDARY IN THAT THEY ARE NOT NECESSARILY ASSOCIATED WITH THE OPERATION
:BEING PERFORMED. WHEN THE SUBROUTINE IDENTIFIES SUCH AN ERROR, IT MOVES
:THE ERROR NUMBER TO THE ERROR CALL IN THE TEST ROUTINE AND THEN RETURNS
:TO THE TEST ROUTINE WHICH MAKES THE ERROR CALL. AFTER THE TEST ROUTINE
:MAKES THE ERROR CALL, IT RETURNS TO THE SUBROUTINE WHICH THEN LOOKS FOR
:OTHER ERRORS. WHEN ALL ERRORS HAVE BEEN REPORTED, THE SUBROUTINE
:RETURNS TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL.

```

```

CALL: JSR PC,SECERR
      BR ??? RETURN HERE IF NO ERROR
      NOP RETURN HERE TO REPORT AN ERROR
      ERROR ERROR NUMBER DEFINED BY SUB
      JSR PC,2(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
      ??? RETURN HERE IF NO MORE ERRORS

```

```

;NOTE: THE SUBROUTINE ASSUMES THAT REGISTERS HAVE BEEN STORED AT THE
;INPUT REGISTER BUFFER.

```

```
SECERR:
```

```

;*****
;STORE FUNCTION CODE AND CLEAR USER'S ERROR NUMBER
MOV RMCS10,5155 ;STORE FUNCTION CODE
BIC #1C<F0!F1!F2!F3!F4>,5155
ADD #4,(SP) ;MOVE (SP) TO ERROR CALL
CLRB 2(SP) ;CLEAR ERROR NUMBER
SUB #4,(SP) ;MOVE (SP) TO NO ERROR RETURN

```

```

;*****
;CHECK SECONDARY ERRORS COMMON TO ALL COMMANDS

```

```

;REPORT ERROR IF DRIVE IS NOT READY, I.E. IF DRY = 0
BIT #DRY,RMDSI ;DRIVE READY??
BNE $$ ;YES!!
MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
BIC #1CDRY,$BDDAT
MOV #DRY,$GDDAT ;GOOD DATA FOR TYPEOUT
ADD #4,(SP)
MOVB #10,2(SP) ;ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT NOT READY
SUB #10,(SP) ;RESTORE (SP) TO ERROR N
NOP

```

```

;REPORT ERROR IF GO BIT IS NOT RESET
SS: BIT #GO,RMCS11 ;GO BIT RESET??
     BEQ 10$ ;YES!!
     MOV RMCS11,$BDDAT ;BAD DATA FOR TYPEOUT
     BIC #1CGO,$BDDAT
     CLR $GDDAT ;GOOD DATA FOR TYPEOUT
     ADD #4,(SP)

```

```

9148 041004 112776 000011 000000      MOVB    #11,2(SP)      ;ERROR NUMBER
9149 041012 162716 000002              SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
9150 041016 004736              JSR    PC,2(SP)+      ;REPORT DEVICE NOT AVAILABLE
9151 041020 162716 000010      SUB     #10,(SP)      ;RESTORE (SP)
9152 041024 000240      NOP
9153
9154      ;REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
10$:  MOV    RMCS1I,$BDDAT ;IS FUNCTION CODE CORRECT??
     BIC    #1C76,$BDDAT
     MOV    515,$GDDAT ;EXPECTED FUNCTION CODE
     CMP    $BDDAT,$GDDAT
     BEQ    15$ ;YES!!
     ADD    #4,(SP)
     MOVB   #12,2(SP)   ;ERROR NUMBER
     SUB    #2,(SP)    ;MOVE SP TO RETURN FOR ERROR
     JSR   PC,2(SP)+  ;REPORT WRONG FUNCTION CODE
     SUB    #10,(SP)   ;RESTORE (SP)
     NOP
9155 041026 013737 001326 001142
9156 041034 042737 177701 001142
9157 041042 013737 044500 001140
9158 041050 023737 001142 001140
9159 041056 001413
9160 041060 062716 000004
9161 041064 112776 000012 000000
9162 041072 162716 000002
9163 041076 004736
9164 041100 162716 000010
9165 041104 000240
9166 041106
9167      15$:
9168      ;REPORT AN ERROR IF COMPOSITE ERROR IS SET AND NO OTHER
9169      ;ERRORS ARE SET, OR IF COMPOSITE ERROR IS NOT SET AND
9170      ;OTHER ERRORS ARE SET
9170 041106 005037 001140      CLR    $GDDAT ;EXPECT "ERR"=0
9171 041112 005737 001342      TST   RMER1I ;IS RMER1 =0??
9172 041116 001003              BNE    20$ ;NO!!
9173 041120 005737 001370      TST   RMER2I ;IS RMERZ=0??
9174 041124 001403              BEQ    25$ ;YES!!
9175 041126 052737 040000 001140 20$:  BIS    #ERR,$GDDAT ;"ERR" SHOULD BE SET
9176 041134 013737 001340 001142 25$:  MOV    RMDSI,$BDDAT
9177 041142 042737 137777 001142      BIC    #1CERR,$BDDAT
9178 041150 023737 001140 001142      CMP    $GDDAT,$BDDAT ;IS "ERR" OK??
9179 041156 001412              BEQ    30$ ;YES!!
9180 041160 062716 000004      ADD    #4,(SP) ;MOVE SP TO USER'S ERROR
9181 041164 112776 000047 000000      MOVB   #47,2(SP) ;WRITE ERROR NUMBER
9182 041172 162716 000002      SUB    #2,(SP) ;MOVE SP TO ERROR RETURN
9183 041176 004736              JSR   PC,2(SP)+ ;REPORT INVALID COMP ERROR
9184 041200 162716 000010      SUB    #10,(SP)
9185
9186      ;REPORT AN ERROR IF "TRE" IS SET AND NONE OF THE BITS WHICH SET
9187      ;TRE IS SET, OR IF TRE IS NOT SET AND ONE OR MORE BITS WHICH
9188      ;SET TRE IS SET
9189 041204 005037 001140 30$:  CLR    $GDDAT ;EXPECT "TRE" =0
9190 041210 013746 001336      MOV    RMCS2I,-(SP) ;WAS DLT, WCE, UPE, NED, NEM
9191 041214 042726 000377      BIC    #377,(SP)+ ;PGE, MXF OR MDPE SET
9192 041220 001010              BNE    35$ ;YES!!
9193 041222 032737 040000 001340      BIT    #ERR,RMDSI ;WAS EXCEPTION RECEIVED??
9194 041230 001407              BEQ    40$ ;NO!!
9195 041232 022737 000030 044500      CMP    #SEARCH,515$ ;WAS DATA TRANSFERRED??
9196 041240 103003              BHIS   40$ ;NO!!
9197 041242 052737 040000 001140 35$:  BIS    #TRE,$GDDAT ;"TRE" SHOULD BE SET
9198 041250 013737 001326 001142 40$:  MOV    RMCS1I,$BDDAT ;BAD DATA FOR TYPEOUT
9199 041256 042737 137777 001142      BIC    #1CTRE,$BDDAT
9200 041264 023737 001140 001142      CMP    $GDDAT,$BDDAT ;IS "TRE" OK??
9201 041272 001413              BEQ    45$ ;YES!!
9202 041274 062716 000004      ADD    #4,(SP) ;MOVE SP TO USER'S ERROR CALL
9203 041300 112776 000014 000000      MOVB   #14,2(SP) ;WRITE ERROR NUMBER

```

E15

DZRMND - RMD3 FUNCTIONAL TEST, PART 2
 DZRMND.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 186
 SECONDARY ERROR CHECK SUBROUTINE

SEQ 0188

9204	041306	162716	000002	
9205	041312	004736		
9206	041314	162716	000010	
9207	041320	000240		
9208	041322			
9209				
9210				
9211				
9212				
9213				
9214				
9215				
9216				
9217				
9218				
9219	041322	010046		
9220	041324	013700	044500	
9221	041330	016037	063676	044472
9222	041336	012600		
9223				
9224				
9225				
9226	041340	013737	044472	001140
9227	041346	032737	040000	001340
9228	041354	001403		
9229	041356	052737	100000	001140
9230	041364	042737	077777	001140
9231	041372	013737	001340	001142
9232	041400	042737	077777	001142
9233	041406	023737	001140	001142
9234	041414	001413		
9235	041416	062716	000004	
9236	041422	112776	000006	000000
9237	041430	162716	000002	
9238	041434	004736		
9239	041436	162716	000010	
9240	041442	000240		
9241	041444			
9242				
9243				
9244				
9245	041444	013737	044472	001140
9246	041452	042737	177776	001140
9247	041460	013737	001342	001142
9248	041466	042737	177776	001142
9249	041474	023737	001140	001142
9250	041502	001412		
9251	041504	062716	000004	
9252	041510	112776	000254	000000
9253	041516	162716	000002	
9254	041522	004736		
9255	041524	162716	000010	
9256	041530	005037	001140	
9257				
9258				
9259	041534	013746	044472	

```

SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
JSR      PC,@(SP)+    ;REPORT TRE ERROR
SUB      #10,(SP)     ;RESTORE (SP)
NOP

45$:
;*****
;USING THE FUNCTION CODE TABLE, CHECK FOR THE FOLLOWING ERRORS:
; .STATUS BITS NOT SET THAT SHOULD BE SET, E.G., ATA AND ILF
; .STATUS BITS SET THAT SHOULD NOT BE SET, E.G., WCE AND ECH
;NOTE THAT SOME ERROR BITS ARE CONDITIONAL ON THE COMMAND AND OTHER
;STATUS CONDITIONS, E.G., WRITE LOCK ERROR SHOULD ONLY BE SET IF
;WRITE LOCK IS ON AND THE COMMAND IS A WRITE.

;GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE
MOV      RO,-(SP)     ;PUSH RO ON STACK
MOV      515,RO      ;RO = FUNCTION CODE
MOV      FNCDB(RO),500$ ;STORE ENTRY
MOV      (SP)+,RO    ;POP STACK INTO RO

;REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
;ATA IS NOT SET AND SHOULD BE SET.
MOV      500$,SGDDAT ;GET EXPECTED ATA STATUS
BIT      #ERR,RMSI   ;IS COMPOSITE ERROR SET ??
BEQ      50$        ;NO !!
BIS      #ATA,SGDDAT ;EXPECT AN ATTENTION
50$:    BIC      #+CAT,SGDDAT ;STRIP DONT CARES
MOV      RMSI,$BDDAT ;GET RECEIVED ATA
BIC      #+CAT,$BDDAT ;STRIP DONT CARES
CMP      $GDDAT,$BDDAT ;IS ATA OK ??
BEQ      55$        ;YES !!
ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
MOVVB   #6,@(SP)    ;LOAD ERROR # IN CALL
SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
JSR      PC,@(SP)+  ;REPORT ERROR
SUB      #10,(SP)   ;RESTORE SP
NOP

55$:
;REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
;WITH FUNCTION CODE TABLE
MOV      500$,SGDDAT ;GET EXPECTED ILF
BIC      #+CILF,SGDDAT ;CLEAR ALL OTHER BITS
MOV      RMERI,$BDDAT ;GET RECEIVED ILF
BIC      #+CILF,$BDDAT ;CLEAR ALL OTHER BITS
CMP      $GDDAT,$BDDAT ;IS ILF OK ??
BEQ      60$        ;YES !!
ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
MOVVB   #254,@(SP)  ;WRITE ERROR NUMBER IN CALL
SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
JSR      PC,@(SP)+  ;REPORT ERROR AND RETURN
SUB      #10,(SP)   ;MOVE SP TO NO ERROR
60$:    CLR      $GDDAT ;CLEAR EXPECTED STATUS

;REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
MOV      500,-(SP)  ;GET WCE STATUS ENABLE
  
```

9260	041540	052716	137777		BIS	#1CWCE, (SP)	;SET ALL OTHER BITS
9261	041544	013737	001336	001142	MOV	RMCS2I, \$BDDAT	;RECEIVED STATUS
9262	041552	042637	001142		BIC	(SP)+, \$BDDAT	;CLEAR WCE IF ENABLED
9263	041556	001412			BEQ	90\$;BRANCH IF WCE OK
9264	041560	062716	000004		ADD	#4, (SP)	;MOVE SP TO USER'S ERROR CALL
9265	041564	112776	000026	000000	MOVB	#26, 2(SP)	;WRITE ERROR NUMBER
9266	041572	162716	000002		SUB	#2, (SP)	;MOVE SP TO ERROR RETURN
9267	041576	004736			JSR	PC, 2(SP)+	;REPORT ERROR
9268	041600	162716	000010		SUB	#10, (SP)	;RESTORE ERROR
9269	041604						
9270							
9271							
9272	041604	013746	044472				
9273	041610	052716	157777				
9274	041614	013737	001342	001142	MOV	500\$, -(SP)	;GET OPI STATUS ENABLE
9275	041622	042637	001142		BIS	#1COPI, (SP)	;SET ALL OTHER BITS
9276	041626	001412			MOV	RMER1I, \$BDDAT	;GET RECEIVED STATUS
9277	041630	062716	000004		BIC	(SP)+, \$BDDAT	;CLEAR OPI IF ENABLED
9278	041634	112776	000164	000000	BEQ	100\$;BRANCH IF OPI OK
9279	041642	162716	000002		ADD	#4, (SP)	;MOVE SP TO USER'S ERROR CALL
9280	041646	004736			MOVB	#164, 2(SP)	;WRITE ERROR NUMBER IN CALL
9281	041650	162716	000010		SUB	#2, (SP)	;MOVE SP TO ERROR RETURN
9282	041654				JSR	PC, 2(SP)+	;REPORT ERROR
9283					SUB	#10, (SP)	;RESTORE SP
9284							
9285							
9286	041654	013746	044472				
9287	041660	032737	000100	001340	MOV	500\$, -(SP)	;GET IVC STATUS ENABLE
9288	041666	001402			BIT	#VV, RMDSI	;IS VV SET
9289	041670	042716	010000		BEQ	105\$;NO !!
9290	041674	052716	167777		BIC	#IVC, (SP)	;YES - IVC SHOULD BE 0
9291	041700	013737	001370	001142	BIS	#1CIVC, (SP)	;SET ALL OTHER BITS
9292	041706	042637	001142		MOV	RMER2I, \$BDDAT	;GET RECEIVED STATUS
9293	041712	001412			BIC	(SP)+, \$BDDAT	;CLEAR IVC IF ENABLED
9294	041714	062716	000004		BEQ	110\$;BRANCH IF IVC OK
9295	041720	112776	000165	000000	ADD	#4, (SP)	;MOVE SP TO USERS ERROR CALL
9296	041726	162716	000002		MOVB	#165, 2(SP)	;WRITE ERROR NUMBER IN CALL
9297	041732	004736			SUB	#2, (SP)	;MOVE SP TO ERROR RETURN
9298	041734	162716	000010		JSR	PC, 2(SP)+	;REPORT ERROR
9299	041740				SUB	#10, (SP)	;RESTORE SP TO NO ERROR
9300							
9301							
9302							
9303							
9304							
9305							
9306							
9307							
9308							
9309							
9310							
9311	041740	012746	177777				
9312	041744	032737	004000	044472	MOV	#-1, -(SP)	;ASSUME WRITE ERRORS ENABLED
9313	041752	001404			BIT	#WLE, 500\$;ARE WRITE ERRORS ENABLED ??
9314	041754	032737	004000	001340	BEQ	115\$;NO !!
9315	041762	001002			BIT	#WRL, RMDSI	;IS THE DRIVE WRITE PROTECTED ??
					BNE	120\$;YES !!


```

9372
9373 042250 013746 044472
9374 042254 052716 175777
9375 042260 013737 001342 001142
9376 042266 042637 001142
9377 042272 001412
9378 042274 062716 000004
9379 042300 112776 000166 000000
9380 042306 162716 000002
9381 042312 004736
9382 042314 162716 000010
9383 042320
9384
9385
9386
9387
9388
9389
9390
9391
9392
9393
9394
9395
9396
9397
9398
9399
9400
9401 042320 012746 177777
9402 042324 032737 001000 044472
9403 042332 001002
9404 042334 042716 100000
9405 042340 013737 001336 001142 165$:
9406 042346 042637 001142
9407 042352 001412
9408 042354 062716 000004
9409 042360 112776 000032 000000
9410 042366 162716 000002
9411 042372 004736
9412 042374 162716 000010
9413 042400
9414
9415
9416 042400 012746 177777
9417 042404 032737 001000 044472
9418 042412 001002
9419 042414 042716 004000
9420 042420 013737 001336 001142 175$:
9421 042426 042637 001142
9422 042432 001412
9423 042434 062716 000004
9424 042440 112776 000167 000000
9425 042446 162716 000002
9426 042452 004736
9427 042454 162716 000010

```

```

;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
MOV 500$, -(SP) ;GET IAE ENABLE
BIS #1,IAE,(SP) ;SET ALL OTHER BITS
MOV RMER1I,$BDDAT ;GET RECEIVED STATUS
BIC (SP)+,$BDDAT ;CLEAR IAE IF ENABLED
BEQ 160$ ;BRANCH IF IAE IS OK
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOV #166,2(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO ERROR RETURN
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR

160$:
;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
;ALL READ/WRITE ERRORS, I.E.,
; RMCS1 - TRE
; RMCS2 - DLT,NEM,MXF
; RMDS - LBT
; RMER1 - AOE
NOTE:
; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
; CYLINDER REGISTER IS WRITTEN
NOTE:
; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
NOTE:
; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE

;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
MOV #-1,-(SP) ;ASSUME ERRORS ARE ENABLED
BIT #AOE,500$ ;ARE ERRORS ENABLED ??
BNE 165$ ;YES !!
BIC #DLT,(SP) ;RESET DLT ENABLE
MOV RMCS2I,$BDDAT ;GET RECEIVED STATUS
BIC (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
BEQ 170$ ;BRANCH IF DLT IS OK
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOV #32,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO ERROR RETURN
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR

170$:
;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
MOV #-1,-(SP) ;ASSUME ERRORS ARE ENABLED
BIT #AOE,500$ ;ARE ERRORS ENABLED ??
BNE 175$ ;YES !!
BIC #NEM,(SP) ;DISABLE NEM
MOV RMCS2I,$BDDAT ;GET RECEIVED STATUS
BIC (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
BEQ 180$ ;BRANCH IF NEM IS OK
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOV #167,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO ERROR RETURN
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR

```

9428	042460				180\$:		
9429							
9430							
9431	042460	012746	177777				
9432	042464	032737	001000	044472			
9433	042472	001002					
9434	042474	042716	001000				
9435	042500	013737	001336	001142	185\$:		
9436	042506	042637	001142				
9437	042512	001412					
9438	042514	062716	000004				
9439	042520	112776	000033	000000			
9440	042526	162716	000002				
9441	042532	004736					
9442	042534	162716	000010				
9443	042540				190\$:		
9444							
9445							
9446	042540	012746	177777				
9447	042544	032737	001000	044472			
9448	042552	001404					
9449	042554	032737	002000	001340			
9450	042562	001002					
9451	042564	042716	001000		191\$:		
9452	042570	013737	001342	001142	195\$:		
9453	042576	042637	001142				
9454	042602	001412					
9455	042604	062716	000004				
9456	042610	112776	000020	000000			
9457	042616	162716	000002				
9458	042622	004736					
9459	042624	162716	000010				
9460	042630				200\$:		
9461							
9462							
9463							
9464							
9465							
9466							
9467							
9468	042630	032737	002000	001360			
9469	042636	001403					
9470	042640	042737	000200	044472			
9471	042646				201\$:		
9472							
9473							
9474	042646	012746	177777				
9475	042652	032737	000200	044472			
9476	042660	001002					
9477	042662	042716	000400				
9478	042666	013737	001342	001142	205\$:		
9479	042674	042637	001142				
9480	042700	001412					
9481	042702	062716	000004				
9482	042706	112776	000035	000000			
9483	042714	162716	000002				


```

9484 042720 004736          JSR    PC,2(SP)+      ;REPORT ERROR AND RETURN
9485 042722 162716 000010   SUB    #10,(SP)        ;MOVE SP TO NO ERROR
9486 042726                210$:
9487
9488
9489 042726 012746 177777   ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
9490 042732 032737 000200 044472   MOV    #-1,-(SP)      ;ASSUME ERRORS ENABLED
9491 042740 001002          BIT    #HCE,500$      ;ARE ERRORS ENABLED ??
9492 042742 042716 000200          BNE    215$          ;YES !!
9493 042746 013737 001342 001142 215$:   BIC    #HCE,(SP)      ;DISABLE HCE
9494 042754 042637 001142          MOV    RMER1I,$BDDAT  ;GET RECEIVED STATUS
9495 042760 001412          BIC    (SP)+,$BDDAT  ;CLEAR HCE IF ENABLED
9496 042762 062716 000004          BEQ    220$          ;BRANCH IF HCE IS OK
9497 042766 112776 000036 000000   ADD    #4,(SP)        ;MOVE SP TO USERS ERROR CALL
9498 042774 162716 000002          MOVB   #36,2(SP)     ;WRITE ERROR NUMBER IN CALL
9499 043000 004736          SUB    #2,(SP)        ;MOVE SP TO ERROR RETURN
9500 043002 162716 000010   JSR    PC,2(SP)+      ;REPORT ERROR AND RETURN
9501 043006                SUB    #10,(SP)        ;MOVE SP TO NO ERROR
9502
9503
9504 043006 012746 177777   ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
9505 043012 032737 000200 044472   MOV    #-1,-(SP)      ;ASSUME FER IS ENABLED
9506 043020 001002          BIT    #HCE,500$      ;ARE HEADER ERRORS ENABLED ??
9507 043022 042716 000020          BNE    225$          ;YES !!
9508 043026 013737 001342 001142 225$:   BIC    #FER,(SP)     ;DISABLE FER
9509 043034 042637 001142          MOV    RMER1I,$BDDAT  ;GET RECEIVED STATUS
9510 043040 001412          BIC    (SP)+,$BDDAT  ;RESET FER IF ENABLED
9511 043042 062716 000004          BEQ    230$          ;BRANCH IF FER OK
9512 043046 112776 000037 000000   ADD    #4,(SP)        ;MOVE SP TO USERS ERROR CALL
9513 043054 162716 000002          MOVB   #37,2(SP)     ;WRITE ERROR NUMBER IN CALL
9514 043060 004736          SUB    #2,(SP)        ;MOVE SP TO ERROR RETURN
9515 043062 162716 000010   JSR    PC,2(SP)+      ;REPORT ERROR AND RETURN
9516 043066                SUB    #10,(SP)        ;MOVE SP TO NO ERROR
9517
9518
9519 043066 012746 177777   ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
9520 043072 032737 000200 044472   MOV    #-1,-(SP)      ;ASSUME ERRORS ENABLED
9521 043100 001002          BIT    #HCE,500$      ;ARE THEY ENABLED ??
9522 043102 042716 100000          BNE    235$          ;YES !!
9523 043106 013737 001370 001142 235$:   BIC    #BSE,(SP)     ;DISABLE BSE
9524 043114 042637 001142          MOV    RMER2I,$BDDAT  ;GET RECEIVED STATUS
9525 043120 001412          BIC    (SP)+,$BDDAT  ;CLEAR BSE IF ENABLED
9526 043122 062716 000004          BEQ    240$          ;BRANCH IF BSE OK
9527 043126 112776 000354 000000   ADD    #4,(SP)        ;MOVE SP TO USERS ERROR CALL
9528 043134 162716 000002          MOVB   #354,2(SP)    ;WRITE ERROR NUMBER
9529 043140 004736          SUB    #2,(SP)        ;MOVE SP TO ERROR RETURN
9530 043142 162716 000010   JSR    PC,2(SP)+      ;REPORT ERROR AND RETURN
9531 043146                SUB    #10,(SP)        ;MOVE SP TO NO ERROR
9532
9533
9534
9535
9536
9537
9538
9539
;BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
;FIELD ERRORS, I.E.
;
;      RMCS2 - MDPE
;      RMER1 - DCK,ECH
;NOTE:
;      ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
;DCK IS SET.

```

```

9540
9541 ;REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
9542 043146 012746 177777 MOV #-1,-(SP) ;ASSUME ENABLED
9543 043152 032737 000100 044472 BIT #ECH,500$ ;ARE DATA FIELD ERRORS ENABLED ??
9544 043160 001002 BNE 245$ ;YES !!
9545 043162 042716 000400 BIC #MDPE,(SP) ;DISBALE MDPE
9546 043166 013737 001336 001142 245$: MOV RMCS21,$BDDAT ;GET RECEIVED STATUS
9547 043174 042637 001142 BIC (SP)+,$BDDAT ;CLEAR MDPE IF ENABLED
9548 043200 001412 BEQ 250$ ;BRANCH IF MDPE OK
9549 043202 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
9550 043206 112776 000027 000000 MOV#B #27,@(SP) ;WRITE ERROR NUMBER IN CALL
9551 043214 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
9552 043220 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
9553 043222 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
9554 043226 250$:
9555
9556 ;REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
9557 043226 012746 177777 MOV #-1,-(SP) ;ASSUME ENABLED
9558 043232 032737 000100 044472 BIT #ECH,500$ ;ARE THEY ENABLED ??
9559 043240 001002 BNE 255$ ;YES !!
9560 043242 042716 100000 BIC #DCK,(SP) ;DISABLE DCK
9561 043246 013737 001342 001142 255$: MOV RMER11,$BDDAT ;GET RECEIVED STATUS
9562 043254 042637 001142 BIC (SP)+,$BDDAT ;CLEAR DCK IF ENABLED
9563 043260 001412 BEQ 260$ ;BRANCH IF DCK IS OK
9564 043262 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
9565 043266 112776 000030 000000 MOV#B #30,@(SP) ;WRITE ERROR NUMBER IN CALL
9566 043274 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
9567 043300 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
9568 043302 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
9569 043306 260$:
9570
9571 ;REPORT ERROR IF ECH IS SET AND
9572 ;DATA FIELD ERRORS ARE NOT ENABLED, OR
9573 ;ECI IS SET, OR
9574 ;DCK IS NOT SET.
9575 043306 012746 177777 MOV #-1,-(SP) ;ASSUME ENABLED
9576 043312 032737 000100 044472 BIT #ECH,500$ ;ARE ERRORS ENABLED ??
9577 043320 001410 BEQ 265$ ;NO !!
9578 043322 032737 004000 001360 BIT #ECI,RMOFI ;IS ECI SET ??
9579 043330 001004 BNE 265$ ;YES !!
9580 043332 032737 100000 001342 BIT #DCK,RMER11 ;IS DCK ALSO SET ??
9581 043340 001002 BNE 270$ ;YES !!
9582 043342 042716 000100 265$: BIC #ECH,(SP) ;DISABLE ECH
9583 043346 013737 001342 001142 270$: MOV RMER11,$BDDAT ;GET RECEIVED STATUS
9584 043354 042637 001142 BIC (SP)+,$BDDAT ;CLEAR ECH IF ENABLED
9585 043360 001412 BEQ 275$ ;BRANCH IF ECH IS OK
9586 043362 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
9587 043366 112776 000031 000000 MOV#B #31,@(SP) ;WRITE ERROR NUMBER IN CALL
9588 043374 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
9589 043400 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
9590 043402 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
9591 043406 275$:
9592
9593 ;*****
9594 ;PERFORM THE REMAINING ERROR CHECKS ONLY FOR DATA TRANSFER COMMANDS
9595

```

L15

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 193
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0195

9596	043406	022737	000030	044500	CMP	#SEARCH,515\$;WAS DATA TRANSFERRED??
9597	043414	103402			BLO	280\$;YES!!
9598	043416	000137	044444		JMP	355\$	
9599							

```

;THE FOLLOWING STATUS CHECKS ARE FOR DATA TRANSFER COMMANDS ONLY
;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
%00
%01
%02 043422 013737 001330 001142 280$: MOV RMWCI,SBDDAT ;WORD COUNT ZERO??
%03 043430 001421 BEQ 285$ ;YES
%04 043432 032737 040000 001326 BIT #TRE,RMCSI1 ;TRANSFER ERROR DETECTED??
%05 043440 001015 BNE 285$ ;YES!!
%06 043442 062716 000004 ADD #4,(SP)
%07 043446 112776 000015 000000 MOVB #15,2(SP) ;ERROR NUMBER
%08 043454 005037 001140 CLR $GDDAT ;GOOD DADA FOR TYPEOUT
%09 043460 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
%10 043464 004736 JSR PC,2(SP)+ ;REPORT WORD COUNT NOT ZERO
%11 043466 162716 000010 SUB #10,(SP) ;RESTORE (SP)
%12 043472 000240 NOP
;REPORT ERROR IF RMBA IS NOT CORRECT
%13
%14 043474 013737 001330 001140 285$: MOV RMWCI,$GDDAT ;NUMBER OF WORDS TRANSFERRED
%15 043502 153737 001400 001140 SUB RMWCO,$GDDAT
%16 043510 006337 001140 ASL $GDDAT
%17 043514 063737 001402 001140 ADD RMBAO,$GDDAT ;EXPECTED BUS ADDRESS
%18 043522 032737 000010 001336 BIT #BAI,RMCSI2 ;WAS BAI SET ??
%19 043530 001403 BEQ 290$ ;NO !!
%20 043532 013737 001402 001140 MOV RMBAO,$GDDAT ;ADDRESS SHOULD NOT HAVE CHANGED
%21 043540 023737 001140 001332 290$: CMP $GDDAT,RMBAI ;BUS ADDRESS OK??
%22 043546 001416 BEQ 295$ ;YES!!
%23 043550 013737 001332 001142 MOV RMBAI,SBDDAT ;BAD DATA FOR TYPEOUT
%24 043556 062716 000004 ADD #4,(SP)
%25 043562 112776 000016 000000 MOVB #16,2(SP) ;ERROR NUMBER
%26 043570 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
%27 043574 004736 JSR PC,2(SP)+ ;REPORT UNEXPECTED ADDRESS
%28 043576 162716 000010 SUB #10,(SP) ;RESTORE (SP)
%29 043602 000240 NOP
;COMPUTE NUMBER OF SECTORS TRANSFERRED
%30
%31 043604 005046 295$: CLR -(SP) ;NUMBER OF SECTORS TRANSFERRED
%32 043606 013746 001330 MOV RMWCI,-(SP) ;NUMBER OF WORDS TRANSFERRED
%33 043612 163716 001400 SUB RMWCO,(SP)
%34 043616 012746 000400 MOV #256,-(SP) ;NUMBER OF WORDS PER SECTOR
%35 043622 032737 000002 001376 BIT #BIT1,RMCSI0 ;HEADER & DATA COMMAND ??
%36 043630 001402 BEQ 300$ ;NO !!
%37 043632 012716 000402 MOV #258,(SP) ;YES - CHANGE WORDS PER SECTOR
%38 043636 005266 000004 300$: INC 4(SP) ;INCREMENT SECTOR COUNT
%39 043642 161666 000002 SUB (SP),2(SP) ;SUBTRACT ONE SECTOR'S WORTH
%40 043646 003373 BGT 300$ ;CONTINUE IF NOT DONE
%41 043650 022626 CMP (SP)+,(SP)+ ;STRIP 2 FROM STACK
%42
;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM NUMBER OF SECTORS
%43 043652 013737 001404 044476 MOV RMDAO,510$ ;STORE ORIGINAL SECTOR
%44 043660 013737 001404 044474 MOV RMDAO,505$ ;STORE ORIGINAL TRACK
%45 043666 013737 001432 044472 MOV RMDCO,500$ ;STORE ORIGINAL CYLINDER
%46 043674 042737 177740 044476 BIC #C37,510$
%47 043702 000337 044474 SWAB 505$
%48 043706 042737 177770 044474 BIC #C7,505$
%49 043714 062637 044476 ADD (SP)+,510$
%50
%51 043720 023727 044476 000040 305$: CMP 510$,#32. ;SECTOR OVEFLOWED??
%52 043726 103420 BLO 315$ ;NO!!
%53 043730 023727 044476 000240 CMP 510$,#(5*32.) ;CYLINDERS WORTH??
%54 043736 103406 BLO 310$ ;NO!!
%55 043740 005237 044472 INC 500$ ;YES INCREMENT CYLINDER

```

```

9656 043744 162737 000240 044476      SUB      #(<5*32.),510$ ;ADJUST SECTOR
9657 043752 000762                    BR       305$ ;TRY AGAIN
9658 043754 005237 044474 044476 310$: INC    505$ ;INCREMENT TRACK
9659 043760 162737 000040 044476      SUB      #32.,510$ ;ADJUST SECTOR
9660 043766 000754                    BR       305$ ;TRY AGAIN
9661
9662 043770 023727 044474 000005 315$: CMP    505$,#5 ;TRACK OVERFLOWED??
9663 043776 103406                    BLO     320$ ;NO!!
9664 044000 005237 044472                    INC    500$ ;INCREMENT CYLINDER
9665 044004 162737 000005 044474      SUB      #5,505$ ;ADJUST TRACK
9666 044012 000766                    BR       315$ ;TRY AGAIN
9667 ;REPORT ERROR IF "LBT" IS NOT CORRECT
9668 044014 005037 001140 044472 320$: CLR    $GDDAT ;SET GOOD DATA FOR LBT=0
9669 044020 022737 001467                    CMP    #823.,500$ ;SHOULD LBT BE SET??
9670 044026 101007                    BHI     325$ ;NO!!
9671 044030 032737 002000 001342      BIT     #IAE,RMER11 ;WAS IAE SET ??
9672 044036 001003                    BNE     325$ ;YES - LBT SHOULD NOT BE SET
9673 044040 012737 002000 001140      MOV    #LBT,$GDDAT ;SET GOOD DATA FOR LBT=1
9674 044046 013737 001340 001142 325$: MOV    RMD5I,$BDDAT ;BAD DATA FOR TYPEOUT
9675 044054 042737 175777 001142      BIC    #↑CLBT,$BDDAT
9676 044062 023737 001140 001142      CMP    $GDDAT,$BDDAT ;IS LBT CORRECT??
9677 044070 001413                    BEQ    330$ ;YES!!
9678 044072 062716 000004                    ADD    #4,(SP)
9679 044076 112776 000017 000000      MOV    #17,2(SP) ;ERROR NUMBER
9680 044104 162716 000002                    SUB    #2,(SP) ;MOVE SP TO RETURN FOR ERROR
9681 044110 004736                    JSR    PC,2(SP)+ ;REPORT LBT IS WRONG
9682 044112 162716 000010                    SUB    #10,(SP) ;RESTORE (SP)
9683 044116 000240                    NOP
9684 ;REPORT ERROR IF "AOE" IS INCORRECT
9685 044120 005037 001140 001342 330$: CLR    $GDDAT ;SET FOR AOE=0
9686 044124 032737 002000                    BIT     #IAE,RMER11 ;WAS "IAE" DETECTED??
9687 044132 001031                    BNE     340$ ;YES-"AOE" SHOULD BE ZERO
9688 044134 022737 001467 044472      CMP    #823.,500$ ;SHOULD AOE BE SET??
9689 044142 101025                    BHI     340$ ;NO!!
9690 044144 005737 044474                    TST    505$ ;MAYBE
9691 044150 001012                    BNE     335$ ;YES
9692 044152 005737 044476                    TST    510$
9693 044156 001007                    BNE     335$ ;YES !!
9694 044160 032737 000010 044500      BIT     #F2,515$ ;WAS THIS READ OR WRITE CHECK ??
9695 044166 001413                    BEQ    340$ ;NO !!
9696 044170 005737 001330                    TST    RMCIC ;WAS ALL DATA TRANSFERRED ??
9697 044174 001410                    BEQ    340$ ;YES !!
9698 044176 012737 001000 001140 335$: MOV    #AOE,$GDDAT ;SET FOR AOE=1
9699 044204 005037 044474                    CLR    505$ ;CLEAR EXPECTED TRACK
9700 044210 012737 000001 044476      MOV    #1,510$ ;EXPECT SECTOR=1
9701 044216 013737 001342 001142 340$: MOV    RMER11,$BDDAT ;BAD DATA FOR TYPEOUT
9702 044224 042737 176777 001142      BIC    #↑CAOE,$BDDAT
9703 044232 023737 001140 001142      CMP    $GDDAT,$BDDAT ;IS AOE CORRECT??
9704 044240 001413                    BEQ    345$ ;YES!!
9705 044242 062716 000004                    ADD    #4,(SP)
9706 044246 112776 000020 000000      MOV    #20,2(SP) ;ERROR NUMBER
9707 044254 162716 000002                    SUB    #2,(SP) ;MOVE SP TO RETURN FOR ERROR
9708 044260 004736                    JSR    PC,2(SP)+ ;REPORT AOE IS WRONG
9709 044262 162716 000010                    SUB    #10,(SP) ;RESTORE (SP)
9710 044266 000240                    NOP
9711 ;REPORT ERROR IF RMDA IS NOT CORRECT

```

9712	044270	032737	002000	001342	345\$:	BIT	#IAE,RMER11	; WAS THERE AN IAE ERROR ??
9713	044276	001062				BNE	355\$; YES - DONT CHECK RMDA,RMDC
9714	044300	013737	044474	001140		MOV	505\$,SGDDAT	; SETUP EXPECTED DISK ADDRESS
9715	044306	000337	001140			SWAB	SGDDAT	
9716	044312	113737	044476	001140		MOVB	510\$,SGDDAT	
9717	044320	013737	001334	001142		MOV	RMDAT,SBDDAT	; SETUP RECEIVED DISK ADDRESS
9718	044326	023737	001140	001142		CMP	SGDDAT,SBDDAT	; COMPARE EXPECTED & RECEIVED
9719	044334	001413				BEQ	350\$; BRANCH IF EQUAL
9720	044336	062716	000004			ADD	#4,(SP)	
9721	044342	112776	000021	000000		MOVB	#21,@(SP)	; ERROR NUMBER
9722	044350	162716	000002			SUB	#2,(SP)	; MOVE SP TO RETURN FOR ERROR
9723	044354	004736				JSR	PC,@(SP)+	; REPORT BAD DISK ADDRESS
9724	044356	162716	000010			SUB	#10,(SP)	; RESTORE (SP)
9725	044362	000240				NOP		
9726								
9727	044364	013737	044472	001140		MOV	500\$,SGDDAT	; SETUP EXPECTED CYLINDER
9728	044372	042737	176000	001140		BIC	#1C1777,SGDDAT	
9729	044400	013737	001362	001142		MOV	RMDCI,SBDDAT	; SETUP RECEIVED CYLINDER
9730	044406	023737	001140	001142		CMP	SGDDAT,SBDDAT	; COMPARE CYLINDERS
9731	044414	001413				BEQ	355\$; BRANCH IF EQUAL
9732	044416	062716	000004			ADD	#4,(SP)	
9733	044422	112776	000022	000000		MOVB	#22,@(SP)	; ERROR NUMBER
9734	044430	162716	000002			SUB	#2,(SP)	; MOVE SP TO RETURN FOR ERROR
9735	044434	004736				JSR	PC,@(SP)+	; REPORT BAD CYLINDER
9736	044436	162716	000010			SUB	#10,(SP)	; RESTORE (SP)
9737	044442	000240				NOP		

; REPORT ERROR IF RMDC IS INCORRECT

350\$:

9738	044444	062716	000004	355\$:	ADD	#4 (SP)	;MOVE (SP) TO ERROR CALL
9739	044450	105776	000000		TSTB	2(SP)	;WAS ERROR FOUND??
9740	044454	001403			BEQ	360\$	
9741	044456	062716	000004		ADD	#4 (SP)	;MOVE (SP) TO ERROR RETURN
9742	044462	000402			BR	365\$	
9743	044464	162716	000004	360\$:	SUB	#4 (SP)	;MOVE (SP) TO NO ERROR RETURN
9744	044470	000207		365\$:	RTS	PC	
9745							
9746	044472	000000		500\$:	.WORD	0	;CYLINDER
9747	044474	000000		505\$:	.WORD	0	;TRACK
9748	044476	000000		510\$:	.WORD	0	;SECTOR
9749	044500	000000		515\$:	.WORD	0	;FUNCTION CODE
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
 9784
 9785
 9786
 9787
 9788
 9789
 9790
 9791
 9792
 9793
 9794
 9795
 9796
 9797
 9798
 9799
 9800
 9801
 9802
 9803
 9804
 9805
 9806
 9807

044502

044502 062716 000004
 044506 105076 000000
 044512 162716 000004
 044516 013746 000004
 044522 013746 000006
 044526 010046
 044530 010146
 044532 012737 044652 000004
 044540 012737 000300 000006
 044546 013700 001276
 044552 013701 001446
 044556 111160 000010
 044562 016037 000000 001176
 044570 016037 000010 001174
 044576 032737 010000 001174
 044604 001407
 044606 062766 000004 000010
 044614 112776 000111 000010
 044622 000422
 044624 032737 004000 001176
 044632 001021
 044634 062766 000004 000010
 044642 112776 000112 000010
 044650 000407
 044652
 044652 022626
 044654 062766 000004 000010
 044662 112776 000113 000010
 044670 162766 000002 000010
 044676
 044676 012601
 044700 012600

```

.SBTTL  DEVICE SELECT SUBROUTINE
; THIS SUBROUTINE SELECTS THE DEVICE, GETTING THE DEVICE NUMBER FROM THE
; TEST QUEUE.

;CALL:
;(1)  JSR    PC,DEVSEL
;(2)  BR     ??          RETURN IF NO ERROR
;(3)  NOP
;(4)  ERROR          RETURN IF ERROR
                       ERROR DEFINED BY SUBROUTINE

DEVSEL:

;CLEAR USER'S ERROR CALL
      ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR
      CLRB  2(SP)        ;CLEAR LOW ORDER BYTE OF CALL
      SUB   #4,(SP)        ;MOVE SP BACK
;SAVE USER'S INFORMATION AND SETUP REGISTERS
      MOV   ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
      MOV   ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
      MOV   RO,-(SP)     ;;PUSH RO ON STACK
      MOV   R1,-(SP)     ;;PUSH R1 ON STACK
      MOV   #20$,ERRVEC  ;SETUP FOR BUS TIMEOUT
      MOV   #PR6,ERRVEC+2
      MOV   $BASE,RO     ;RO = UNIBUS ADDRESS
      MOV   TSTQUE,R1    ;R1 POINTS TO DEVICE NUMBER

;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
      MOVB  (R1),RMCS2(RO) ;WRITE UNIT SELECT BITS
      MOV   RMCS1(RO),$TMP1 ;GET "DVA" STATUS
      MOV   RMCS2(RO),$TMP0 ;GET "NED" STATUS
      BIT   #NED,$TMP0     ;IS DEVICE NONEXISTENT??
      BEQ   10$           ;NO!!
      ADD   #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
      MOVB  #111,210(SP)  ;WRITE ERROR NUMBER
      BR    30$
      BIT   #DVA,$TMP1    ;IS DEVICE AVAILABLE??
      BNE   35$           ;YES!!
      ADD   #4,10(SP)
      MOVB  #112,210(SP)
      BR    30$

20$:

;HANDLE BUS TIMEOUT
      CMP   (SP)+,(SP)+  ;ADJUST SP
      ADD   #4,10(SP)
      MOVB  #113,210(SP)
      SUB   #2,10(SP)    ;MOVE SP TO RETURN IF ERROR

30$:

35$:

;RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
      MOV   (SP)+,R1    ;;POP STACK INTO R1
      MOV   (SP)+,RO    ;;POP STACK INTO RO
    
```


E16

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 199
DEVICE SELECT SUBROUTINE

SEQ 0201

9808 044702 012637 000006
9809 044706 012637 000004
9810 044712 000207

MOV (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;:POP STACK INTO ERRVEC
RTS PC ;EXIT

9811
9812
9813
9814
9815
9816
9817
9818
9819
9820
9821
9822
9823
9824
9825
9826
9827
9828
9829 044714
9830
9831
9832 044714 000240
9833 044716 062716 000004
9834 044722 105076 000000
9835 044726 162716 000004
9836 044732 005037 046152
9837
9838
9839
9840 044736 032737 000010 001342
9841 044744 001424
9842 044746 032737 000010 001370
9843 044754 001020
9844
9845
9846 044756 005037 001140
9847 044762 013737 001342 001142
9848 044770 062716 000004
9849 044774 112776 000050 000000
9850 045002 162716 000002
9851 045006 004736
9852 045010 162716 000010
9853 045014 000437
9854
9855
9856
9857
9858 045016 012737 002000 001140
9859 045024 052737 040000 046152
9860 045032 023727 001432 001466
9861 045040 101025
9862 045042 042737 040000 046152
9863 045050 123727 001404 000037
9864 045056 101016
9865 045060 123727 001404 000035
9866 045066 101404

```

.SBTTL SEEK STATUS CHECK SUBROUTINE
; THIS SUBROUTINE VERIFIES THE RESULTS OF SEEK TESTS USING STATUS
; STORED IN THE GET BUFFER AND TEST PARAMETERS STORED IN THE PUT BUFFER.

; THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS DETECTED
; AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN THE "ERROR" TRAP
; OF THE CALLING ROUTINE. SEEK STATUS IS CHECKED AS FOLLOWS:

CALL:
(1) JSR PC,SEKSTS
     BR   ???          RETURN HERE IF NO ERROR
     NOP          RETURN HERE TO REPORT AN ERROR
     ERROR        ERROR NUMBER DEFINED BY SUB
     JSR PC,@(SP)+  GO BACK TO SUB FOR MORE ERROR CHECKS
     ???          RETURN HERE IF NO MORE ERRORS

SEKSTS:
; CLEAR USER'S ERROR CALL
NOP
ADD #4,(SP)      ; MOVE (SP) TO ERROR CALL
CLRB @(SP)      ; CLEAR ERROR NUMBER
SUB #4,(SP)     ; MOVE (SP) TO NO ERROR RETURN
CLR 300$       ; CLEAR ERROR FLAGS

; TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
; LOCAL REGISTERS, I.E., "PAR" = 1 AND "DPE" = 0
BIT #PAR,RMER1I ; WAS PARITY ERROR DETECTED??
BEQ 1$         ; NO!!
BIT #DPE,RMER2I ; WAS IT DUE TO CONTROL BUS??
BNE 1$         ; NOT SURE!!

; REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
CLR $GDDAT     ; EXPECTED STATUS
MOV RMER1I,$BDDAT ; RECEIVED STATUS
ADD #4,(SP)    ; MOVE STACK TO USER'S ERROR
MOVB #50,@(SP) ; ERROR #50
SUB #2,(SP)    ; MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+
SUB #10,(SP)   ; RESTORE STACK
BR 3$         ; IAE SHOULD BE ZERO

; DETERMINE THE VALUE OF "IAE" STATUS BASED ON TRACK, SECTOR AND
; CYLINDER ADDRESS USED DURING SEEK OPERATION
1$: MOV #IAE,$GDDAT ; SET UP FOR IAE = 1
    BIS #SKI,300$  ; SET SKI ERROR FLAG
    CMP RMDCO,#822. ; CYLINDER > 822??
    BHI 3$        ; YES!!
    BIC #SKI,300$ ; CLEAR SKI ERROR FLAG
    CMPB RMDAO,#31. ; SECTOR > 31??
    BHI 3$        ; YES!!
    CMPB RMDAO,#29. ; SECTOR > 29 ??
    BLOS 2$      ; NO!!
    
```

```

9867 045070 032737 010000 001430      BIT      #FMT16,RMOFO      ;30 SECTOR FORMAT??
9868 045076 001406 001405 000004      BEQ      3$              ;YES!!
9869 045100 123727 001405 000004      2$: CMPB   RMDAO+1,#4.    ;TRACK >4??
9870 045106 101002 001140 000004      BHI      3$              ;YES!!
9871 045110 005037 001140 000004      CLR      $GDDAT         ;"IAE" SHOULD BE 0
9872
9873                                     ;COMPARE EXPECTED AND RECEIVED "IAE" STATUS
9874 045114 013737 001342 001142      3$: MOV   RMER1I,$BDDAT  ;IS IAE OK??
9875 045122 042737 175777 001142      BIC     #↑CIAE,$BDDAT
9876 045130 023737 001140 001142      CMP     $GDDAT,$BDDAT
9877 045136 001004 000000 000000      BNE     35$             ;IAE IN ERROR
9878 045140 042737 040000 046152      BIC     #SKI,300$      ;CLEAR SKI FLAG
9879 045146 000413 000000 000000      BR      5$              ;GO CHECK NEXT ERROR
9880 045150
9881                                     35$: ;REPORT INCORRECT "IAE" STATUS VIA USER'S ERROR CALL
9882 045150 062716 000004 000000      ADD     #4,(SP)
9883 045154 112776 000051 000000      MOVVB  #51,2(SP)       ;ERROR 51
9884 045162 162716 000002 000000      SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
9885 045166 004736 000000 000000      JSR    PC,2(SP)+      ;REPORT INCORRECT IAE
9886 045170 162716 000010 000000      SUB     #10,(SP)      ;RESTORE (SP)
9887 045174 000240
9888 045176
9889
9890                                     5$: ;REPORT ANY IVC ERROR AS
9891                                     ; IVC ERROR WITH VOLUME VALID ZERO
9892                                     ; ERRONEOUS IVC ERROR, VOLUME VALID IS SET
9893 045176 032737 010000 001370      BIT     #IVC,RMER2I    ;IVC ERROR??
9894 045204 001427 001140 000000      BEQ     52$            ;NO!!
9895 045206 005037 001140 000000      CLR     $GDDAT        ;EXPECTED STATUS
9896 045212 013737 001370 001142      MOV     RMER2I,$BDDAT ;RECEIVED STATUS
9897 045220 062716 000004 000000      ADD     #4,(SP)       ;MOVE SP TO USER'S ERROR
9898 045224 112776 000060 000000      MOVVB  #60,2(SP)     ;ERROR 60 IF VV = 0
9899 045232 032737 000100 001340      BIT     #VV,RMDSI
9900 045240 001403 000000 000000      BEQ     51$
9901 045242 112776 000061 000000      MOVVB  #61,2(SP)     ;ERROR 61 IF VV = 1
9902 045250 162716 000002 000000      51$: SUB     #2,(SP)    ;MOVE SP TO RETURN FOR ERROR
9903 045254 004736 000000 000000      JSR    PC,2(SP)+      ;REPORT ERROR VIA USER
9904 045256 162716 000010 000000      SUB     #10,(SP)     ;RESTORE SP
9905 045262 000240
9906
9907 045264 013737 001370 001142      52$: MOV     RMER2I,$BDDAT ;RECEIVED STATUS
9908 045272 042737 137777 001142      BIC     #↑CSKI,$BDDAT ;CLEAR DONT CARES
9909 045300 013737 046152 001140      MOV     300$,$GDDAT   ;GET EXPECTED SKI STATUS
9910 045306 042737 137777 001140      BIC     #↑CSKI,$GDDAT ;CLEAR DONT CARES
9911 045314 001417 000000 000000      BEQ     53$            ;BRANCH IF 0 EXPECTED
9912
9913                                     ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
9914 045316 032737 040000 001142      BIT     #SKI,$BDDAT   ;WAS SKI DETECTED ??
9915 045324 001032 000000 000000      BNE     54$            ;YES !!
9916 045326 062716 000004 000000      ADD     #4,(SP)       ;MOVE SP TO USERS ERROR CALL
9917 045332 112776 000267 000000      MOVVB  #267,2(SP)    ;WRITE ERROR NUMBER
9918 045340 162716 000002 000000      SUB     #2,(SP)       ;MOVE SP TO ERROR RETURN
9919 045344 004736 000000 000000      JSR    PC,2(SP)+      ;REPORT ERROR AND RETURN
9920 045346 162716 000010 000000      SUB     #10,(SP)     ;MOVE SP TO NO ERROR
9921 045352 000443 000000 000000      BR      6$              ;GO TO NEXT ERROR CHECK
9922 045354

```

```

9923
9924 ;REPORT ERROR IF SKI IS SET
9925 045354 032737 040000 001142 BIT #SKI,$BDDAT ;IS SKI SET ??
9926 045362 001413 BEQ 54$ ;NO - SKI IS OK
9927 045364 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
9928 045370 112776 000054 000000 MOVB #54,2(SP) ;LOAD ERROR NUMBER
9929 045376 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
9930 045402 004736 JSR PC,2(SP)+ ;REPORT SEEK ERROR
9931 045404 162716 000010 SUB #10,(SP) ;RESTORE (SP)
9932 045410 000240 NOP
9933
9934 ;REPORT ANY DEVICE CHECK
9935 045412 032737 000200 001370 54$: BIT #DVC,RMER2I ;WAS THERE DVC DURING SEEK??
9936 045420 001420 BEQ 6$ ;NO!!
9937 045422 005037 001140 CLR $GDDAT ;EXPECTED STATUS
9938 045426 013737 001370 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
9939 045434 062716 000004 ADD #4,(SP)
9940 045440 112776 000055 000000 MOVB #55,2(SP) ;ERROR #55
9941 045446 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
9942 045452 004736 JSR PC,2(SP)+ ;REPORT ERROR VIA USER
9943 045454 162716 000010 SUB #10,(SP) ;RESTORE SP
9944 045460 000240 NOP
9945
9946 ;REPORT ANY "OPI" ERROR AS OPI WITH MOL =0, OR OPI
9947 ;BECAUSE ON CYLINDER LATCH DIDN'T RESET
9948 045462 032737 020000 001342 6$: BIT #OPI,RMER1I ;"OPI" ERROR??
9949 045470 001427 BEQ 8$ ;NO!!
9950 045472 005037 001140 CLR $GDDAT ;EXPECTED STATUS
9951 045476 013737 001342 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
9952 045504 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
9953 045510 112776 000052 000000 MOVB #52,2(SP) ;LOAD ERROR NUMBER
9954 045516 032737 010000 001340 BIT #MOL,RMDSI ;WAS MEDIUM ON LINE??
9955 045524 001403 BEQ 7$ ;NO!!
9956 045526 112776 000053 000000 MOVB #53,2(SP) ;YES - CHANGE ERROR NUMBER
9957 045534 162716 000002 7$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
9958 045540 004736 JSR PC,2(SP)+ ;REPORT "OPI" ERROR
9959 045542 162716 000010 SUB #10,(SP) ;RESTORE (SP)
9960 045546 000240 NOP
9961
9962 ;SEE IF "PIP" IS 0, AND "ATA","MOL"AND"VV" =1
9963 045550 013746 001340 8$: MOV RMDSI, -(SP)
9964 045554 042716 047677 BIC #C<ATA!PIP!MOL!VV>,(SP)
9965 045560 022726 110100 CMP #ATA!MOL!VV,(SP)+
9966 045564 001002 BNE 9$ ;ERROR IN RMD3
9967 045566 000137 046122 JMP 14$ ;RMD3 IS OK
9968
9969 ;REPORT ERROR IF MOL = 0 AND OPI = 0
9970 045572 032737 010000 001340 9$: BIT #MOL,RMDSI ;IS MOL RESET??
9971 045600 001030 BNE 10$ ;NO - MOL IS SET
9972 045602 032737 020000 001342 BIT #OPI,RMER1I ;WAS OPI SET
9973 045610 001024 BNE 10$ ;YES - DONT REPORT ERROR
9974 045612 013737 001340 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
9975 045620 052737 010000 001140 BIS #MOL,$GDDAT
9976 045626 013737 001340 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
9977 045634 062716 000004 ADD #4,(SP)
9978 045640 112776 000062 000000 MOVB #62,2(SP)

```

9979	045646	162716	000002		SUB	#2,(SP)		;MOVE SP TO RETURN FOR ERROR
9980	045652	004736			JSR	PC,@(SP)+		;REPORT ERROR VIA USER
9981	045654	162716	000010		SUB	#10,(SP)		
9982	045660	000240			NOP			
9983								
9984								;REPORT AN ERROR IF "PIP" IS STIL SET AND SKI NOT SET
9985	045662	032737	020000	001340	10\$:	BIT	#PIP,RMSI	;IS "PIP" STILL SET??
9986	045670	001430			BEQ	11\$;NO!!
9987	045672	032737	040000	001370	BIT	#SKI,RMER2I		;WAS "SKI"SET??
9988	045700	001024			BNE	11\$;YES-DONT REPORT PIP
9989	045702	013737	001340	001140	MOV	RMSI,\$GDDAT		;EXPECTED STATUS
9990	045710	042737	020000	001142	BIC	#PIP,\$BDDAT		
9991	045716	013737	001340	001142	MOV	RMSI,\$BDDAT		;RECEIVED STATUS
9992	045724	062716	000004		ADD	#4,(SP)		;MOVE (SP) TO ERROR
9993	045730	112776	000056	000000	MOVB	#56,@(SP)		;LOAD ERROR NUMBER
9994	045736	162716	000002		SUB	#2,(SP)		;MOVE SP TO RETURN FOR ERROR
9995	045742	004736			JSR	PC,@(SP)+		;REPORT "PIP" SET AFTER SEEK
9996	045744	162716	000010		SUB	#10,(SP)		;RESTORE (SP)
9997	045750	000240			NOP			
9998								
9999								;REPORT AN ERROR IF "ATA" IS NOT SET
10000	045752	032737	100000	001340	11\$:	BIT	#ATA,RMSI	;WAS "ATA" SET ??
10001	045760	001024			BNE	13\$;YES!!
10002	045762	013737	001340	001140	MOV	RMSI,\$GDDAT		;EXPECTED STATUS
10003	045770	052737	110600	001140	BIS	#ATA!MOL!DPR!DRY,\$GDDAT		
10004	045776	013737	001340	001142	MOV	RMSI,\$BDDAT		;RECEIVED STATUS
10005	046004	062716	000004		ADD	#4,(SP)		;MOVE (SP) TO ERROR
10006	046010	112776	000057	000000	MOVB	#57,@(SP)		;LOAD ERROR NUMBER
10007	046016	162716	000002		SUB	#2,(SP)		;MOVE SP TO RETURN FOR ERROR
10008	046022	004736			JSR	PC,@(SP)+		;REPORT ATTENTION NOT SET DURING
10009								;SEEK TEST
10010	046024	162716	000010		SUB	#10,(SP)		;RESTORE (SP)
10011	046030	000240			NOP			
10012								
10013								;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
10014	046032	032737	000100	001340	13\$:	BIT	#VV,RMSI	;IS VV = 0 ??
10015	046040	001030			BNE	14\$;NO!!
10016	046042	032737	010000	001370	BIT	#IVC,RMER2I		;IS IVC ALSO 0 ??
10017	046050	001024			BNE	14\$;NO - IVC IS SET
10018	046052	013737	001340	001140	MOV	RMSI,\$GDDAT		;EXPECTED STATUS
10019	046060	052737	000100	001140	BIS	#VV,\$GDDAT		
10020	046066	013737	001340	001142	MOV	RMSI,\$BDDAT		;RECEIVED STATUS
10021	046074	062716	000004		ADD	#4,(SP)		
10022	046100	112776	000064	000000	MOVB	#64,@(SP)		;ERROR #64
10023	046106	162716	000002		SUB	#2,(SP)		;MOVE SP TO RETURN FOR ERROR
10024	046112	004736			JSR	PC,@(SP)+		
10025	046114	162716	000010		SUB	#10,(SP)		
10026	046120	000240			NOP			
10027	046122							14\$:
10028								
10029								;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
10030	046122	000240			NOP			
10031	046124	062716	000004		ADD	#4,(SP)		;MOVE (SP) TO ERROR CALL
10032	046130	105776	000000		TSTB	@(SP)		;WAS ERROR CALLED??
10033	046134	001403			BEQ	15\$;NO!!
10034	046136	062716	000004		ADD	#4,(SP)		;MOVE TO ERROR RETURN

J16

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 204
SEEK STATUS CHECK SUBROUTINE

SEQ 0206

10035	046142	000402	BR	16\$	
10036					
10037	046144	162716	000004	15\$: SUB #4, (SP)	; MOVE (SP) TO NO ERROR RETURN
10038	046150	000207		16\$: RTS PC	; RETURN
10039					
10040	046152	000000		300\$: .WORD 0	; ERROR FLAGS
10041					

10042
 10043
 10044
 10045
 10046
 10047
 10048
 10049
 10050
 10051
 10052
 10053
 10054
 10055
 10056
 10057
 10058
 10059
 10060
 10061
 10062
 10063
 10064
 10065
 10066
 10067
 10068
 10069
 10070
 10071
 10072
 10073
 10074
 10075

046154				
046154	010046			
046156	010146			
046160	013746	000004		
046164	013746	000006		
046170	012737	046230	000004	
046176	012737	000300	000006	
046204	013700	001276		
046210	012760	000040	000010	
046216	013701	001446		
046222	111160	000010		
046226	000412			
046230	022626			
046232	062766	000004	000010	
046240	112776	000007	000010	
046246	162766	000002	000010	
046254				
046254	012637	000006		
046260	012637	000004		
046264	012601			
046266	012600			
046270	000207			

.SBTTL CONTROLLER CLEAR SUBROUTINE

; THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
 ; AND DRIVES, THEN SELECTS THE DRIVE.

```

CALL: JSR    PC,CNTCLR
      BR     ???
      NOP
      ERROR
      ???
  
```

RETURN HERE IF NO ERROR FOUND
 RETURN HERE IF ANY ERROR FOUND
 SUB DEFINES ERROR NUMBER

```

CNTCLR:
MOV    RO,-(SP)      ;; PUSH RO ON STACK
MOV    R1,-(SP)      ;; PUSH R1 ON STACK
MOV    ERRVEC,-(SP)  ;; PUSH ERRVEC ON STACK
MOV    ERRVEC+2,-(SP) ;; PUSH ERRVEC+2 ON STACK
MOV    #10$,ERRVEC   ; SETUP FOR BUS TIMEOUT
MOV    #PR6,ERRVEC+2
MOV    $BASE,RO      ; RO=UNIBUS ADDRESS
MOV    #CLR,AMCS2(RO) ; CLEAR MASSBUS
MOV    TSTQUE,R1
MOVB   (R1),AMCS2(RO) ; SELECT DEVICE
BR     20$
10$:  CMP    (SP)+,(SP)+ ; ADJUST STACK
      ADD   #4,10(SP)   ; MOVE SP TO USER'S ERROR CALL
      MOVB #7,210(SP)  ; WRITE THE ERROR NUMBER
      SUB   #2,10(SP)
20$:  MOV    (SP)+,ERRVEC+2 ;; POP STACK INTO ERRVEC+2
      MOV    (SP)+,ERRVEC  ;; POP STACK INTO ERRVEC
      MOV    (SP)+,R1      ;; POP STACK INTO R1
      MOV    (SP)+,RO      ;; POP STACK INTO RO
      RTS   PC
  
```

```

10076
10077
10078
10079
10080
10081
10082
10083
10084
10085
10086
10087
10088
10089
10090
10091
10092
10093
10094
10095
10096
10097 046272
10098
10099
10100 046272 062716 000004
10101 046276 105076 000000
10102 046302 162716 000004
10103
10104 046306 013737 001326 001142
10105 046314 042737 100000 001142
10106 046322 012737 004200 001140
10107 046330 023737 001140 001142
10108 046336 001413
10109 046340 062716 000004
10110 046344 112776 000126 000000
10111 046352 162716 000002
10112 046356 004736
10113 046360 162716 000010
10114 046364 000240
10115
10116 046366 005037 001140
10117 046372 013737 001332 001142
10118 046400 001413
10119 046402 062716 000004
10120 046406 112776 000127 000000
10121 046414 162716 000002
10122 046420 004736
10123 046422 162716 000010
10124 046426 000240
10125
10126 046430 013737 001336 001142
10127 046436 010146
10128 046440 005046
10129 046442 013701 001446
10130 046446 111116
10131 046450 052716 000100

```

```

.SBTTL CONTROLLER CLEAR STATUS CHECK SUBROUTINE

; THIS SUBROUTINE VERIFIES THAT THE RMO3 SUBSYSTEM IS INITIALIZED BASED ON
; STATUS STORED IN THE GET BUFFER. THIS SUBROUTINE SHOULD ONLY BE
; USED FOLLOWING A CONTROLLER CLEAR OPERATION, I.E., WRITING A 1 IN BIT
; 5 OF RMCS2, BECAUSE THE ERROR MESSAGES ARE BASED ON THAT CONDITION.

; STATUS PERTINENT TO THE DEVICE IS NOT CHECKED. IN PARTICULAR, THE
; FOLLOWING STATUS BITS ARE NOT CHECKED:
;
; ATA,ERR,PIP,MOL,WRL,LBT,PGM,VV,OM,UNS,SKI,DVC

; CALL:
; (1) JSR PC,CLRSTS
; BR ??? RETURN HERE IF NO ERROR
; NOP RETURN HERE TO REPORT AN ERROR
; ERROR ERROR NUMBER DEFINED BY SUB
; JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
; ??? RETURN HERE IF NO MORE ERRORS

CLRSTS:
; CLEAR USER'S ERROR CALL
; MOVE SP TO ERROR
; CLEAR ERROR NUMBER
; MOVE SP BACK TO NO ERROR
; REPORT ERROR IF RMCS1 NOT INITIALIZED
4$: MOV RMCS1I,$BDDAT ; VERIFY RMCS1
; IGNORE SPECIAL CONDITION
; EXPECT DVA & RDY
; COMPARE EXPECTED, RECEIVED
; BRANCH IF EQUAL
; MOVE SP TO USER'S ERROR CALL
; WRITE ERROR NUMBER IN CALL
; MOVE SP TO RETURN FOR ERROR
; REPORT ERROR VIA USER
; MOVE SP BACK TO NO ERROR
; REPORT ERROR IF RMBA NOT RESET
5$: CLR $GDDAT ; VERIFY RMBA IS ZERO
; BRANCH IF ZERO
; MOVE SP TO USER'S ERROR CALL
; WRITE ERROR NUMBER IN CALL
; MOVE SP TO RETURN FOR ERROR
; REPORT ERROR VIA USER
; MOVE SP BACK TO NO ERROR
; REPORT ERROR IF RMCS2 NOT INITIALIZED
7$: MOV RMCS2I,$BDDAT ; VERIFY RMCS2
; PUSH R1 ON STACK
; EXPECT IR & UNIT NUMBER
; R1 = ADDRESS OF TEST QUE
MOV R1,-(SP)
CLR -(SP)
MOV TSTQUE,R1
MOVB (R1),(SP)
BIS #IR,(SP)

```



```

10132 046454 012637 001140      MOV      (SP)+,SGDDAT
10133 046460 012601      MOV      (SP)+,R1          ;;POP STACK INTO R1
10134 046462 023737 001140 001142  CMP      SGDDAT,SBDDAT    ;;COMPARE EXPECTED & RECEIVED
10135 046470 001413      BEQ      9$              ;;BRANCH IF EQUAL
10136 046472 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
10137 046476 112776 000130 000000  MOVVB   #130,a(SP)       ;;WRITE ERROR NUMBER IN CALL
10138 046504 162716 000002      SUB      #2,(SP)        ;;MOVE SP TO RETURN FOR ERROR
10139 046510 004736      JSR      PC,a(SP)+       ;;REPORT ERROR VIA USER
10140 046512 162716 000010      SUB      #10,(SP)       ;;MOVE SP BACK TO NO ERROR
10141 046516 000240      NOP
10142      ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
10143 046520 005037 001140 9$:      CLR      SGDDAT          ;VERIFY RMER1
10144 046524 013737 001342 001142  MOV      RMER1I,SBDDAT
10145 046532 042737 040000 001142  BIC      #UNS,SBDDAT     ;IGNORE UNSAFE
10146 046540 001413      BEQ      13$            ;;BRANCH IF ZERO
10147 046542 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
10148 046546 112776 000131 000000  MOVVB   #131,a(SP)       ;;WRITE ERROR NUMBER IN CALL
10149 046554 162716 000002      SUB      #2,(SP)        ;;MOVE SP TO RETURN FOR ERROR
10150 046560 004736      JSR      PC,a(SP)+       ;;REPORT ERROR VIA USER
10151 046562 162716 000010      SUB      #10,(SP)       ;;MOVE SP BACK TO NO ERROR
10152 046566 000240      NOP
10153      ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
10154 046570 013737 001352 001142 13$:   MOV      RMMR1I,SBDDAT   ;VERIFY RMMR
10155 046576 042737 000046 001142  BIC      #WC!LS!LST,SBDDAT ;IGNORE WORD CLOCK, SCT, TRK
10156 046604 012737 000010 001140  MOV      #MMD,SGDDAT     ;EXPECT WRITE DATA BIT
10157 046612 023737 001140 001142  CMP      SGDDAT,SBDDAT   ;;COMPARE EXPECTED AND RECEIVED
10158 046620 001413      BEQ      17$            ;;BRANCH IF 0
10159 046622 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
10160 046626 112776 000133 000000  MOVVB   #133,a(SP)       ;;WRITE ERROR NUMBER IN CALL
10161 046634 162716 000002      SUB      #2,(SP)        ;;MOVE SP TO RETURN FOR ERROR
10162 046640 004736      JSR      PC,a(SP)+       ;;REPORT ERROR VIA USER
10163 046642 162716 000010      SUB      #10,(SP)       ;;MOVE SP BACK TO NO ERROR
10164 046646 000240      NOP
10165      ;REPORT AN ERROR IF RMEC2 IS NOT RESET
10166 046650 005037 001140 17$:   CLR      SGDDAT          ;EXPECT ZEROS
10167 046654 013737 001374 001142  MOV      RMEC2I,SBDDAT   ;VERIFY RMEC2=0
10168 046662 001413      BEQ      19$
10169 046664 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
10170 046670 112776 000135 000000  MOVVB   #135,a(SP)       ;;WRITE ERROR NUMBER IN CALL
10171 046676 162716 000002      SUB      #2,(SP)        ;;MOVE SP TO RETURN FOR ERROR
10172 046702 004736      JSR      PC,a(SP)+       ;;REPORT ERROR VIA USER
10173 046704 162716 000010      SUB      #10,(SP)       ;;MOVE SP BACK TO NO ERROR
10174 046710 000240      NOP
10175      ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
10176 046712 013737 001366 001142 19$:   MOV      RMMR2I,SBDDAT   ;VERIFY RMMR2
10177 046720 042737 140000 001142  BIC      #RQA!RQB,SBDDAT
10178 046726 012737 011777 001140  MOV      #TST!1777,SGDDAT ;EXPECT TEST BIT ON
10179 046734 023737 001140 001142  CMP      SGDDAT,SBDDAT
10180 046742 001413      BEQ      21$
10181 046744 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
10182 046750 112776 000136 000000  MOVVB   #136,a(SP)       ;;WRITE ERROR NUMBER IN CALL
10183 046756 162716 000002      SUB      #2,(SP)        ;;MOVE SP TO RETURN FOR ERROR
10184 046762 004736      JSR      PC,a(SP)+       ;;REPORT ERROR VIA USER
10185 046764 162716 000010      SUB      #10,(SP)       ;;MOVE SP BACK TO NO ERROR
10186 046770 000240      NOP
10187      ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC

```

10188	046772	005037	001140		215:	CLR	SGDAT	: EXPECT ALL ZEROS
10189	046776	013737	001370	001142		MOV	RMR2I, SBDDAT	: VERIFY RMR2
10190	047004	042737	040200	001142		BIC	#SKI!DVC, SBDDAT	: IGNORE DEVICE ERRORS
10191	047012	001413				BEQ	215\$: BRANCH IF OTHER BITS 0
10192	047014	062716	000004			ADD	#4, (SP)	: MOVE SP TO USER'S ERROR CALL
10193	047020	112776	000174	000000		MOVB	#174, 2(SP)	: WRITE ERROR NUMBER IN CALL
10194	047026	162716	000002			SUB	#2, (SP)	: MOVE SP TO RETURN FOR ERROR
10195	047032	004736				JSR	PC, 2(SP)+	: REPORT ERROR VIA USER
10196	047034	162716	000010			SUB	#10, (SP)	: MOVE SP BACK TO NO ERROR
10197	047040	000240				NOP		
10198								: REPORT ERROR IF RMD3 NOT INITIALIZED
10199	047042	013737	001340	001142	215\$:	MOV	RMDSI, SBDDAT	: TEST DRIVE STATUS REGISTER
10200	047050	042737	177177	001142		BIC	#C<DRY!DPR>, SBDDAT	
10201	047056	012737	000600	001140		MOV	#DPR!DRY, SGDAT	: EXPECTED DRIVE STATUS
10202	047064	023737	001140	001142		CMP	SGDAT, SBDDAT	: COMPARE EXPECTED & RECEIVED
10203	047072	001413				BEQ	22\$: BRANCH IF EQUAL
10204	047074	062716	000004			ADD	#4, (SP)	: MOVE SP TO USER'S ERROR CALL
10205	047100	112776	000134	000000		MOVB	#134, 2(SP)	: WRITE ERROR NUMBER
10206	047106	162716	000002			SUB	#2, (SP)	: MOVE SP TO RETURN FOR ERROR
10207	047112	004736				JSR	PC, 2(SP)+	: REPORT ERROR TO USER
10208	047114	162716	000010			SUB	#10, (SP)	: MOVE SP BACK TO NO ERROR
10209	047120	000240				NOP		
10210	047122	062716	000004		22\$:	ADD	#4, (SP)	: MOVE SP TO ERROR CALL
10211	047126	105776	000000			TSTB	2(SP)	: WAS AN ERROR DETECTED??
10212	047132	001403				BEQ	23\$: NO!!
10213	047134	062716	000004			ADD	#4, (SP)	: YES - MOVE TO ERROR RETURN
10214	047140	000402				BR	24\$	
10215	047142	162716	000004		23\$:	SUB	#4, (SP)	: MOVE SP TO NO ERROR RETURN
10216	047146	000240			24\$:	NOP		
10217	047150	000207				RTS	PC	

```

10218
10219
10220
10221
10222
10223
10224
10225
10226
10227
10228
10229
10230
10231
10232 047152
10233
10234
10235 047152 062716 000004
10236 047156 105076 000000
10237 047162 162716 000004
10238
10239
10240 047166 032737 000100 001340
10241 047174 001024
10242 047176 013737 001340 001140
10243 047204 052737 000100 001140
10244 047212 013737 001340 001142
10245 047220 062716 000004
10246 047224 112776 000155 000000
10247 047232 162716 000002
10248 047236 004736
10249 047240 162716 000010
10250 047244 000240
10251 047246
10252
10253
10254 047246 032737 010000 001340
10255 047254 001024
10256 047256 013737 001340 001140
10257 047264 052737 010000 001140
10258 047272 013737 001340 001142
10259 047300 062716 000004
10260 047304 112776 000041 000000
10261 047312 162716 000002
10262 047316 004736
10263 047320 162716 000010
10264 047324 000240
10265 047326
10266
10267
10268 047326 032737 060007 001342
10269 047334 001570
10270
10271
10272 047336 032737 040000 001342
10273 047344 001424

```

```

.SBTTL PACK ACKNOWLEDGE STATUS CHECK

;THIS SUBROUTINE CHECKS THE RESULTS OF A PACK ACKNOWLEDGE
;COMMAND USING THE STATUS STORED IN THE GET BUFFER. ERRORS ARE
;REPORTED TO THE USER VIA THE USER'S ERROR CALL.

:CALL:
:(1) JSR PC,ACKSTS
      BR ???
      NOP
      ERROR
      JSR PC,2(SP)+
      ???

RETURN HERE IF NO ERROR
RETURN HERE TO REPORT AN ERROR
ERROR NUMBER DEFINED BY SUB
GO BACK TO SUB FOR MORE ERROR CHECKS
RETURN HERE IF NO MORE ERRORS

ACKSTS:

;CLEAR USER'S ERROR CALL
ADD #4,(SP)
CLR #2,(SP)
SUB #4,(SP)

;MOVE SP TO ERROR CALL
;CLEAR LOW ORDER BYTE
;MOVE SP BACK

;REPORT AN ERROR IF "VV" IS 0
BIT #VV,RMSI
BNE IS
MOV RMSI,$GDDAT
BIS #VV,$GDDAT
MOV RMSI,$BDDAT
ADD #4,(SP)
MOVB #155,2(SP)
SUB #2,(SP)
JSR PC,2(SP)+
SUB #10,(SP)
NOP

;IS VOLUME VALID SET??
;YES!!
;EXPECTED STATUS
;RECEIVED STATUS
;MOVE SP TO ERROR CALL
;WRITE NUMBER IN ERROR CALL
;MOVE SP TO RETURN FOR ERROR
;REPORT THE ERROR
;MOVE SP BACK TO BRANCH

IS:
;REPORT AN ERROR IF "MOL" IS 0
BIT #MOL,RMSI
BNE 2$
MOV RMSI,$GDDAT
BIS #MOL,$GDDAT
MOV RMSI,$BDDAT
ADD #4,(SP)
MOVB #41,2(SP)
SUB #2,(SP)
JSR PC,2(SP)+
SUB #10,(SP)
NOP

;IS MOL SET??
;YES!!
;EXPECTED STATUS
;RECEIVED STATUS
;MOVE SP TO ERROR CALL
;WRITE NUMBER OF ERROR IN CALL
;MOVE SP TO RETURN FOR ERROR
;REPORT TH ERROR
;MOVE SP TO BRANCH

2$:
;SEE IF "UNS","OPI","RMR","ILR" OR "ILF" IS SET
BIT #UNS!OPI!RMR!ILR!ILF,RMER1I
BEQ 7$

;REPORT AN ERROR IF "UNS" IS SET
BIT #UNS,RMER1I
BEQ 3$

;WAS UNS SET??
;NO!!

```

10274	047346	013737	001342	001142	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
10275	047354	013737	001342	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
10276	047362	042737	040000	001140	BIC	#UNS,\$GDDAT	
10277	047370	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
10278	047374	112776	000042	000000	MOVB	#42,2(SP)	;WRITE NUMBER OF ERROR IN CALL
10279	047402	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
10280	047406	004736			JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
10281	047410	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
10282	047414	000240			NOP		
10283	047416						
10284							
10285							
10286	047416	032737	020000	001342			
10287	047424	001424					
10288	047426	013737	001342	001142	BIT	#OPI,RMER11	;WAS OPI SET ??
10289	047434	013737	001342	001140	BEQ	4\$;NO!!
10290	047442	042737	020000	001140	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
10291	047450	062716	000004		MOV	RMER11,\$GDDAT	;EXPECTED STATUS
10292	047454	112776	000043	000000	BIC	#OPI,\$GDDAT	
10293	047462	162716	000002		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
10294	047466	004736			MOVB	#43,2(SP)	;WRITE NUMBER OF ERROR IN CALL
10295	047470	162716	000010		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
10296	047474	000240			JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
10297	047476				SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
10298					NOP		
10299							
10300	047476	032737	000004	001342			
10301	047504	001424					
10302	047506	013737	001342	001142	BIT	#RMR,RMER11	;WAS RMR SET??
10303	047514	013737	001342	001140	BEQ	5\$;NO!!
10304	047522	042737	000004	001140	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
10305	047530	062716	000004		MOV	RMER11,\$GDDAT	;EXPECTED STATUS
10306	047534	112776	000044	000000	BIC	#RMR,\$GDDAT	
10307	047542	162716	000002		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
10308	047546	004736			MOVB	#44,2(SP)	;WRITE NUMBER OF ERROR IN CALL
10309	047550	162716	000010		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
10310	047554	000240			JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
10311	047556				SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
10312					NOP		
10313							
10314	047556	032737	000002	001342			
10315	047564	001424					
10316	047566	013737	001342	001142			
10317	047574	013737	001342	001140			
10318	047602	042737	000002	001140			
10319	047610	062716	000004				
10320	047614	112776	000045	000000			
10321	047622	162716	000002				
10322	047626	004736					
10323	047630	162716	000010				
10324	047634	000240					
10325	047636						
10326							
10327							
10328	047636	032737	000001	001342			
10329	047644	001424					

E01

DZRM0A - RM03 FUNCTIONAL TEST, PART 2
DZRM0A.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 211
PACK ACKNOWLEDGE STATUS CHECK

SEQ 0213

10330	047646	013737	001342	001142	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
10331	047654	013737	001342	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
10332	047662	042737	000001	001140	BIC	#1LF,\$GDDAT	
10333	047670	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
10334	047674	112776	000046	000000	MOVB	#46,2(SP)	;WRITE NUMBER OF ERROR IN CALL
10335	047702	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
10336	047706	004736			JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
10337	047710	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
10338	047714	000240			NOP		
10339	047716						
10340							
10341							
10342	047716	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
10343	047722	105776	000000		TSTB	2(SP)	;WAS ERROR FOUND??
10344	047726	001403			BEQ	8\$;NO!!
10345	047730	062716	000004		ADD	#4,(SP)	;YES - MOVE TO ERROR RETURN
10346	047734	000402			BR	9\$	
10347	047736	162716	000004		SUB	#4,(SP)	;MOVE SP TO NO ERROR RETURN
10348	047742	000240			NOP		
10349	047744	000207			RTS	PC	
10350							

7\$:

;AUGMENT RETURN ADDRESS IF ERROR WAS FOUND

8\$:

9\$:

10351
 10352
 10353
 10354
 10355
 10356
 10357
 10358
 10359
 10360
 10361
 10362
 10363
 10364
 10365 047746
 10366
 10367
 10368 047746 062716 000004
 10369 047752 105076 000000
 10370 047756 162716 000004
 10371
 10372
 10373
 10374 047762 032737 022011 001342
 10375 047770 001553
 10376
 10377
 10378
 10379 047772 032737 000010 001342
 10380 050000 001430
 10381 050002 032737 000010 001370
 10382 050010 001024
 10383 050012 013737 001342 001140
 10384 050020 042737 000010 001140
 10385 050026 032737 001342 001142
 10386 050034 062716 000004
 10387 050040 112776 000050 000000
 10388 050046 162716 000002
 10389 050052 004736
 10390 050054 162716 000010
 10391 050060 000240
 10392 050062
 10393
 10394
 10395 050062 032737 000001 001342
 10396 050070 001424
 10397 050072 013737 001342 001140
 10398 050100 042737 000001 001140
 10399 050106 013737 001342 001142
 10400 050114 062716 000004
 10401 050120 112776 000071 000000
 10402 050126 162716 000002
 10403 050132 004736
 10404 050134 162716 000010
 10405 050140 000240
 10406 050142

```

.SBTTL RECALIBRATE STATUS CHECK SUBROUTINE

;THIS SUBROUTINE CHECKS THE RESULTS OF A RECALIBRATE OPERATION
;USING THE STATUS STORED IN THE GET BUFFER.

;CALL:
(1) JSR PC,RCLSTS ;CALL SUBROUTINE
BR ??? ;RETURN HERE IF NO ERROR
NOP ;RETURN HERE TO REPORT AN ERROR
ERROR ;ERROR NUMBER DEFINED BY SUB
JSR PC,@(SP)+ ;GO BACK TO SUB FOR MORE ERROR CHECKS
??? ;RETURN HERE IF NO MORE ERRORS

RCLSTS:

;CLEAR USER'S ERROR NUMBER
ADD #4,(SP)
CLRB @(SP) ;CLEAR USER'S ERROR CALL
SUB #4,(SP) ;MOVE SP BACK TO BRANCH

;SEE IF "PAR" OR "ILF" OR "OPI" OR "IAE" IS SET
BIT #OPI!PAR!ILF!IAE,RMER1I
BEQ 4$ ;NONE ARE SET - GO TO NEXT CHECK

;REPORT ANY MASSBUS CONTROL BUS PARITY ERROR, I.E.,
;"PAR" = 1 AND "DPE" = 0
BIT #PAR,RMER1I ;WAS "PAR" SET??
BEQ 1$ ;NO!!
BIT #DPE,RMER2I ;WAS "DPE" SET??
BNE 1$ ;YES - NOT A REGISTER ERROR
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #PAR,$GDDAT
MOV RMER1I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #50,@(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;GO REPORT ERROR
SUB #10,(SP) ;MOVE SP BACK TO BRANCH
NOP

1$:

;REPORT ANY "ILF" ERROR
BIT #ILF,RMER1I ;WAS "ILF" SET??
BEQ 2$ ;NO!!
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #ILF,$GDDAT
MOV RMER1I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #71,@(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO BRANCH
NOP

2$:
    
```

10407						
10408						
10409						
10410						
10411	050142	032737	020000	001342		
10412	050150	001433				
10413	050152	013737	001342	001140		
10414	050160	042737	020000	001140		
10415	050166	013737	001342	001142		
10416	050174	062716	000004			
10417	050200	112776	000072	000000		
10418	050206	032737	010000	001340		
10419	050214	001403				
10420	050216	112776	000073	000000		
10421	050224	162716	000002		3\$:	
10422	050230	004736				
10423	050232	162716	000010			
10424	050236	000240				
10425	050240					
10426						
10427						
10428	050240	032737	002000	001342		
10429	050246	001424				
10430	050250	013737	001342	001140		
10431	050256	042737	002000	001140		
10432	050264	013737	001342	001142		
10433	050272	062716	000004			
10434	050276	112776	000070	000000		
10435	050304	162716	000002			
10436	050310	004736				
10437	050312	162716	000010			
10438	050316	000240				
10439	050320					
10440						
10441						
10442	050320	032737	050200	001370		
10443	050326	001517				
10444						
10445						
10446						
10447						
10448						
10449	050330	032737	010000	001370		
10450	050336	001433				
10451	050340	013737	001370	001140		
10452	050346	042737	010000	001140		
10453	050354	013737	001370	001142		
10454	050362	062716	000004			
10455	050366	112776	000074	000000		
10456	050374	032737	000100	001340		
10457	050402	001403				
10458	050404	112776	000075	000000		
10459	050412	162716	000002		5\$:	
10460	050416	004736				
10461	050420	162716	000010			
10462	050424	000240				

```

;REPORT ANY "OPI" ERROR AS
; . OPI DUE TO "MOL" = 0
; . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
;
BIT #OPI,RMER1I ;WAS OPI SET??
BEQ 31$ ;NO!!
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #OPI,$GDDAT
MOV RMER1I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #72,2(SP) ;WRITE ERROR NUMBER IN USER'S CALL
BIT #MOL,RMDSI ;WAS "MOL" = 0??
BEQ 3$ ;YES!!
MOVB #73,2(SP) ;NO - CHANGE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO BRANCH
NOP

31$:
;REPORT AN ERROR IF "IAE" IS SET
BIT #IAE,RMER1I ;IS "IAE" SET??
BEQ 4$ ;NO!!
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #IAE,$GDDAT
MOV RMER1I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO ERROR CALL
MOVB #70,2(SP) ;WRITE ERROR NUMBER IN USER'S CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR
SUB #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN
NOP

4$:
;SEE IF "SKI" OR "IVC" OR "DVC" IS SET
BIT #SKI!IVC!DVC,RMER2I
BEQ 8$ ;NONE OF THE BITS ARE SET

;REPORT ANY "IVC" ERROR AS
; . IVC WITH VV = 0
; . ERRONEOUS IVC ERROR
;
BIT #IVC,RMER2I ;WAS IVC SET??
BEQ 6$ ;NO!!
MOV RMER2I,$GDDAT ;EXPECTED STATUS
BIC #IVC,$GDDAT
MOV RMER2I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #74,2(SP) ;WRITE ERROR NUMBER IN CALL
BIT #VV,RMDSI ;WAS VV = 0??
BEQ 5$ ;YES!!
MOVB #75,2(SP) ;NO - CHANGE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO BRANCH
NOP
5$:

```

```

10463 050426
10464
10465
10466 050426 032737 040000 001370
10467 050434 001424
10468 050436 013737 001370 001140
10469 050444 042737 040000 001140
10470 050452 013737 001370 001142
10471 050460 062716 000004
10472 050464 112776 000076 000000
10473 050472 162716 000002
10474 050476 004736
10475 050500 162716 000010
10476 050504 000240
10477 050506
10478
10479
10480 050506 032737 000200 001370
10481 050514 001424
10482 050516 013737 001370 001140
10483 050524 042737 000200 001140
10484 050532 013737 001370 001142
10485 050540 062716 000004
10486 050544 112776 000077 000000
10487 050552 162716 000002
10488 050556 004736
10489 050560 162716 000010
10490 050564 000240
10491 050566
10492
10493
10494 050566 013746 001340
10495 050572 042716 047676
10496 050576 022726 110100
10497 050602 001002
10498 050604 000137 051220
10499 050610
10500
10501
10502
10503 050610 032737 010000 001340
10504 050616 001030
10505 050620 032737 020000 001342
10506 050626 001024
10507 050630 013737 001340 001140
10508 050636 052737 010000 001140
10509 050644 013737 001340 001142
10510 050652 062716 000004
10511 050656 112776 000100 000000
10512 050664 162716 000002
10513 050670 004736
10514 050672 162716 000010
10515 050676 000240
10516 050700
10517
10518

```

```

6$:
;REPORT ANY "SKI" ERROR
BIT #SKI,RMER2I ;WAS SKI SET??
BEQ 7$ ;NO!!
MOV RMER2I,$GDDAT ;EXPECTED STATUS
BIC #SKI,$GDDAT
MOV RMER2I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #76,2(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP TO BRANCH
NOP

7$:
;REPORT ANY "DVC" ERROR
BIT #DVC,RMER2I ;WAS "DVC" SET??
BEQ 8$ ;NO!!
MOV RMER2I,$GDDAT ;EXPECTED STATUS
BIC #DVC,$GDDAT
MOV RMER2I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #77,2(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
NOP

8$:
;SEE IF "PIP" AND "OM" ARE 0, AND "ATA" "MOL" AND "VV" ARE 1
MOV RMDSI,-(SP) ;PUT RMD5 ON STACK
BIC #1C<PIP!MOL!VV!OM!ATA>,(SP)
CMP #ATA!MOL!VV,(SP)+
BNE 85$
JMP 13$

85$:
;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
;LINE AFTER RECALIBRATE WAS INITIATED
BIT #MOL,RMDSI ;DID MOL DROP??
BNE 9$ ;NO!!
BIT #OPI,RMER1I ;WAS OPI ERROR REPORTED??
BNE 9$ ;YES - DON'T REPORT MOL=0
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIS #MOL,$GDDAT
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #100,2(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
NOP

9$:
;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0

```



```

10519 050700 032737 000100 001340      BIT      #VV,RMDSI      ;DID "VV" DROP??
10520 050706 001030                    BNE      10$         ;NO!!
10521 050710 032737 010000 001370      BIT      #IVC,RMER2I ;WAS THERE A IVC ERROR??
10522 050716 001024                    BNE      10$         ;YES - DONT REPORT VV=0
10523 050720 013737 001340 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
10524 050726 013737 001340 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
10525 050734 052737 000100 001140      BIS      #VV,$GDDAT
10526 050742 062716 000004                    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
10527 050746 112776 000101 000000      MOV      #101,2(SP) ;WRITE ERROR NUMBER IN CALL
10528 050754 162716 000002                    SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
10529 050760 004736                    JSR      PC,2(SP)+
10530 050762 162716 000010                    SUB      #10,(SP)    ;MOVE SP BACK TO USER'S BRANCH
10531 050766 000240                    NOP
10532 050770
10533
10534 ;REPORT AN ERROR IF ATA IS NOT SET
10535 050770 032737 100000 001340      BIT      #ATA,RMDSI  ;WAS ATA SET DURING RECALIBRATE??
10536 050776 001024                    BNE      11$         ;YES!!
10537 051000 013737 001340 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
10538 051006 052737 100000 001140      BIS      #ATA,$GDDAT
10539 051014 013737 001340 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
10540 051022 062716 000004                    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
10541 051026 112776 000102 000000      MOV      #102,2(SP) ;WRITE ERROR NUMBER IN CALL
10542 051034 162716 000002                    SUB      #2,(SP)
10543 051040 004736                    JSR      PC,2(SP)+
10544 051042 162716 000010                    SUB      #10,(SP)    ;MOVE SP TO USER'S BRANCH
10545 051046 000240                    NOP
10546
10547 051050
10548
10549 ;REPORT AN ERROR IF "OM" IS NOT ZERO BECAUSE RECALIBRATE SHOULD
10550 ;ALWAYS CLEAR OFFSET MODE
10551 051050 032737 000001 001340      BIT      #OM,RMDSI   ;WAS "OM" RESET??
10552 051056 001424                    BEQ      12$         ;YES!!
10553 051060 013737 001340 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
10554 051066 042737 000001 001140      BIC      #OM,$GDDAT
10555 051074 013737 001340 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
10556 051102 062716 000004                    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
10557 051106 112776 000103 000000      MOV      #103,2(SP) ;WRITE ERROR NUMBER
10558 051114 162716 000002                    SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
10559 051120 004736                    JSR      PC,2(SP)+   ;REPORT ERROR VIA USER
10560 051122 162716 000010                    SUB      #10,(SP)    ;MOVE SP TO USER'S BRANCH
10561 051126 000240                    NOP
10562 051130
10563
10564 ;REPORT AN ERROR IF "PIP" IS STIL ON, I.E., DRIVE NOT ON
10565 ;CYLINDER
10566 051130 032737 020000 001340      BIT      #PIP,RMDSI  ;IS DRIVE OFF CYLINDER??
10567 051136 001430                    BEQ      13$         ;NO!!
10568 051140 032737 040000 001370      BIT      #SKI,RMER2I ;WAS "SKI" DETECTED??
10569 051146 001024                    BNE      13$         ;YES-DONT REPORT "PIP"
10570 051150 013737 001340 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
10571 051156 042737 020000 001140      BIC      #PIP,$GDDAT
10572 051164 013737 001340 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
10573 051172 062716 000004                    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
10574 051176 112776 000104 000000      MOV      #104,2(SP) ;WRITE ERROR NUMBER

```


K01

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 217
RECALIBRATE STATUS CHECK SUBROUTINE

SEQ 0219

10631 051464 105776 000000
10632 051470 001403
10633 051472 062716 000004
10634 051476 000402
10635 051500 162716 000004
10636 051504 000240
10637 051506 000207
10638

TSTB 2(SP) ; WAS AN ERROR REPORTED??
BEQ 17\$; NO!!
ADD #4, (SP) ; YES - AUGMENT SP RETURN
BR 18\$
17\$: SUB #4, (SP) ; NO ERROR - RETURN SP TO BRANCH
18\$: NOP
RTS PC ; STATUS CECK IS COMPLETE

```

10639
10640
10641
10642
10643
10644
10645
10646 051510
10647
10648
10649 051510 062716 000004
10650 051514 105076 000000
10651 051520 162716 000004
10652
10653 051524 013737 001326 001142
10654 051532 042737 173700 001142
10655 051540 012737 004010 001140
10656 051546 023737 001140 001142
10657 051554 001443
10658 051556 062716 000004
10659 051562 112776 000141 000000
10660 051570 162716 000002
10661 051574 004736
10662 051576 162716 000010
10663 051602 000240
10664
10665 051604 013737 001340 001142
10666 051612 042737 021101 001142
10667 051620 012737 010600 001140
10668 051626 023737 001140 001142
10669 051634 001413
10670 051636 062716 000004
10671 051642 112776 000142 000000
10672 051650 162716 000002
10673 051654 004736
10674 051656 162716 000010
10675 051662 000240
10676
10677 051664 005037 001140
10678 051670 013737 001342 001142
10679 051676 001413
10680 051700 062716 000004
10681 051704 112776 000143 000000
10682 051712 162716 000002
10683 051716 004736
10684 051720 162716 000010
10685 051724 000240
10686
10687 051726 013737 001344 001142
10688 051734 010146
10689 051736 010246
10690 051740 013701 001446
10691 051744 116102 000001
10692 051750 042702 177400
10693 051754 005102
10694 051756 040237 001142

```

```

.SBTTL DRIVE CLEAR STATUS CHECK SUBROUTINE
: BR      ???      RETURN HERE IF NO ERROR
: NOP     RETURN HERE TO REPORT AN ERROR
: ERROR   ERROR NUMBER DEFINED BY SUB
: JSR    PC,2(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
: ???    RETURN HERE IF NO MORE ERRORS

DRVSTS:

;CLEAR USER'S ERROR CALL
ADD     #4,(SP)      ;MOVE SP TO ERROR CALL
CLR     2(SP)       ;CLEAR ERROR CALL
SUB     #4,(SP)     ;MOVE SP TO USER'S BRANCH

;REPORT ERROR IF RMCS1 NOT INITIALIZED
4S: MOV     RMCS1I,$BDDAT ;CHECK RMCS1
BIC     #1C<DVA!FNCSK>,$BDDAT ;CLEAR DONT CARES
MOV     #DVA!DRVCLR,$GDDAT ;EXPECT DVA
CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
BEQ     6S          ;BRANCH IF EQUAL
ADD     #4,(SP)     ;MOVE SP TO ERROR CALL
MOVB   #141,2(SP)  ;WRITE NUMBER OF ERROR IN CALL
SUB     #2,(SP)    ;MOVE SP TO RETURN FOR ERROR
JSR    PC,2(SP)+  ;REPORT THE ERROR VIA USER
SUB     #10,(SP)   ;MOVE SP TO NO ERROR RETURN
NOP

;REPORT ERROR IF RMDS NOT INITIALIZED
5S: MOV     RMDSI,$BDDAT ;CHECK RMDS
BIC     #PGM!OM!VV!PIP,$BDDAT ;CLEAR DONT CARES
MOV     #MOL!DPR!DRY,$GDDAT ;EXPECT DRY & DPR
CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
BEQ     6S          ;BRANCH IF EQUAL
ADD     #4,(SP)     ;MOVE SP TO ERROR CALL
MOVB   #142,2(SP)  ;WRITE NUMBER OF ERROR IN CALL
SUB     #2,(SP)    ;MOVE SP TO RETURN FOR ERROR
JSR    PC,2(SP)+  ;REPORT THE ERROR VIA USER
SUB     #10,(SP)   ;MOVE SP TO NO ERROR RETURN
NOP

;REPORT ERROR IF RMER1 NOT INITIALIZED
6S: CLR     $GDDAT ;EXPECT 0'S
MOV     RMER1I,$BDDAT ;CHECK RMER1
BEQ     8S          ;BRANCH IF EQUAL
ADD     #4,(SP)     ;MOVE SP TO ERROR CALL
MOVB   #143,2(SP)  ;WRITE NUMBER OF ERROR IN CALL
SUB     #2,(SP)    ;MOVE SP TO RETURN FOR ERROR
JSR    PC,2(SP)+  ;REPORT THE ERROR VIA USER
SUB     #10,(SP)   ;MOVE SP TO NO ERROR RETURN
NOP

;REPORT ERROR IF ATA NOT INITIALIZED
8S: MOV     RMASI,$BDDAT ;CHECK ATTENTION BIT
MOV     R1,-(SP)    ;;PUSH R1 ON STACK
MOV     R2,-(SP)    ;;PUSH R2 ON STACK
MOV     TSTQUE,R1
MOVB   1(R1),R2
BIC     #1CATNMSK,R2
COM     R2
BIC     R2,$BDDAT

```

10695	051762	012602			MOV	(SP)+,R2	::POP STACK INTO R2
10696	051764	012601			MOV	(SP)+,R1	::POP STACK INTO R1
10697	051766	005737	001142		TST	\$BDDAT	::IS ATTENTION CLEARED??
10698	051772	001413			BEQ	9\$::BRANCH IF ATTENTION CLEARED
10699	051774	062716	000004		ADD	#4,(SP)	::MOVE SP TO ERROR CALL
10700	052000	112776	000144	000000	MOVB	#144,2(SP)	::WRITE NUMBER OF ERROR IN CALL
10701	052006	162716	000002		SUB	#2,(SP)	::MOVE SP TO RETURN FOR ERROR
10702	052012	004736			JSR	PC,2(SP)+	::REPORT THE ERROR VIA USER
10703	052014	162716	000010		SUB	#10,(SP)	::MOVE SP TO NO ERROR RETURN
10704	052020	000240			NOP		
10705							
10706	052022	013737	001352	001142	9\$:	MOV RMMR1I,\$BDDAT	::CHECK RMMR1
10707	052030	042737	000046	001142	BIC	#WC!LS!LST,\$BDDAT	::CLEAR DONT CARES
10708	052036	012737	000010	001140	MOV	#MMD,\$GDDAT	::EXPECT WRITE DATA ON
10709	052044	023737	001140	001142	CMP	\$GDDAT,\$BDDAT	::COMPARE EXPECTED AND RECEIVED
10710	052052	001413			BEQ	11\$::BRANCH IF ZERO
10711	052054	062716	000004		ADD	#4,(SP)	::MOVE SP TO ERROR CALL
10712	052060	112776	000145	000000	MOVB	#145,2(SP)	::WRITE NUMBER OF ERROR IN CALL
10713	052066	162716	000002		SUB	#2,(SP)	::MOVE SP TO RETURN FOR ERROR
10714	052072	004736			JSR	PC,2(SP)+	::REPORT THE ERROR VIA USER
10715	052074	162716	000010		SUB	#10,(SP)	::MOVE SP TO NO ERROR RETURN
10716	052100	000240			NOP		
10717							
10718	052102	013737	001366	001142	11\$:	MOV RMMR2I,\$BDDAT	::CHECK RMMR2
10719	052110	042737	140000	001142	BIC	#RQA!RQB,\$BDDAT	::CLEAR REQA, REQB
10720	052116	012737	011777	001140	MOV	#TST!1777,\$GDDAT	::EXPECT TEST BIT ON
10721	052124	023737	001140	001142	CMP	\$GDDAT,\$BDDAT	::COMPARE EXPECTED & RECEIVED
10722	052132	001413			BEQ	15\$::BRANCH IF EQUAL
10723	052134	062716	000004		ADD	#4,(SP)	::MOVE SP TO ERROR CALL
10724	052140	112776	000146	000000	MOVB	#146,2(SP)	::WRITE NUMBER OF ERROR IN CALL
10725	052146	162716	000002		SUB	#2,(SP)	::MOVE SP TO RETURN FOR ERROR
10726	052152	004736			JSR	PC,2(SP)+	::REPORT THE ERROR VIA USER
10727	052154	162716	000010		SUB	#10,(SP)	::MOVE SP TO NO ERROR RETURN
10728	052160	000240			NOP		
10729	052162	005037	001140		15\$:	CLR \$GDDAT	::EXPECT ZEROS
10730							
10731	052166	013737	001374	001142	17\$:	MOV RMEC2I,\$BDDAT	::CHECK RMEC2
10732	052174	001413			BEQ	17\$::BRANCH IF 0
10733	052176	062716	000004		ADD	#4,(SP)	::MOVE SP TO ERROR CALL
10734	052202	112776	000150	000000	MOVB	#150,2(SP)	::WRITE NUMBER OF ERROR IN CALL
10735	052210	162716	000002		SUB	#2,(SP)	::MOVE SP TO RETURN FOR ERROR
10736	052214	004736			JSR	PC,2(SP)+	::REPORT THE ERROR VIA USER
10737	052216	162716	000010		SUB	#10,(SP)	::MOVE SP TO NO ERROR RETURN
10738	052222	000240			NOP		
10739							
10740	052224	013737	001370	001142	18\$:	MOV RMR2I,\$BDDAT	::CHECK RMR2
10741	052232	001413			BEQ	18\$::BRANCH IF NO ERROR
10742	052234	062716	000004		ADD	#4,(SP)	::MOVE SP TO ERROR CALL
10743	052240	112776	000147	000000	MOVB	#147,2(SP)	::WRITE NUMBER OF ERROR IN CALL
10744	052246	162716	000002		SUB	#2,(SP)	::MOVE SP TO RETURN FOR ERROR
10745	052252	004736			JSR	PC,2(SP)+	::REPORT THE ERROR VIA USER
10746	052254	162716	000010		SUB	#10,(SP)	::MOVE SP TO NO ERROR RETURN
10747	052260	000240			NOP		
10748	052262				18\$:		
10749							
10750	052262				19\$:		

10751			
10752			
10753	052262	062716	000004
10754	052266	105776	000000
10755	052272	001403	
10756	052274	062716	000004
10757	052300	000402	
10758	052302	162716	000004
10759	052306	000240	
10760	052310	000207	

```

;AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND
      ADD    #4,(SP)      ;MOVE SP TO ERROR CALL
      TSTB   @2(SP)      ;WAS AN ERROR DETECTED??
      BEQ    21$         ;NO!!
      ADD    #4,(SP)      ;YES - MOVE SP TO ERROR RETURN
      BR     23$
21$:   SUB    #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
23$:   NOP
      RTS    PC          ;RETURN TO USER

```

```

10761
10762
10763
10764
10765
10766
10767
10768
10769
10770
10771
10772
10773
10774
10775
10776
10777 052312
10778
10779
10780 052312 062716 000004
10781 052316 105076 000000
10782 052322 162716 000004
10783 052326 005037 055706
10784
10785
10786 052332 032737 020000 001326
10787 052340 001422
10788 052342 013737 001326 001140
10789 052350 042737 020000 001140
10790 052356 013737 001326 001142
10791 052364 062716 000004
10792 052370 112776 000013 000000
10793 052376 162716 000002
10794 052402 004736
10795 052404 000466
10796 052406
10797
10798
10799
10800 052406 032737 000010 001342
10801 052414 001435
10802 052416 032737 000010 001370
10803 052424 001031
10804 052426 013737 001342 001140
10805 052434 042737 000010 001140
10806 052442 013737 001342 001142
10807 052450 062716 000004
10808 052454 112776 000050 000000
10809 052462 032737 001000 001326
10810 052470 001003
10811 052472 112776 000274 000000
10812 052500 162716 000002
10813 052504 004736
10814 052506 000425
10815
10816 052510

```

```

.SBTTL DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

; THIS SUBROUTINE VERIFIES THE RESULTS OF ALL DATA TRANSFER COMMANDS
; USING STATUS STORED IN THE GET BUFFER AND TEST PARAMETERS
; STORED IN THE PUT BUFFER. ERRORS ARE REPORTED BY WRITING
; THE ERROR NUMBER IN THE USERS ERROR CALL.

; USER'S SUBROUTINE CALL:
;(1) JSR PC,DTASTS
;(2) BR ??
;(3) NOP
;(4) ERROR
;(5) JSR PC,@(SP)+
;(6) ??

RETURN HERE IF NO DATA ERRORS
RETURN HERE TO REPORT AN ERROR
SUB WRITES ERROR NUMBER
USER RETURNS FOR MORE CHECKS
SUB RETURNS HERE AFTER ALL
ERRORS ARE REPORTED

DTASTS:
; CLEAR USER'S ERROR CALL AND ERROR FLAGS
ADD #4,(SP) ; MOVE SP TO USER'S ERROR
CLRB @(SP) ; CLEAR LOW ORDER BYTE OF TRAP
SUB #4,(SP) ; RESTORE SP TO NO ERROR
CLR 500$ ; CLEAR ERROR FLAGS

; REPORT ANY CONTROL BUS PARITY ERROR WHILE READING REMOTE REGISTERS, I.E., MCPE = 1
BIT #MCPE,RMCS1I ; WAS THERE A PARITY ERROR??
BEQ 10$ ; NO!!
MOV RMCS1I,$GDDAT ; EXPECTED STATUS
BIC #MCPE,$GDDAT
MOV RMCS1I,$BDDAT ; RECEIVED STATUS
ADD #4,(SP) ; MOVE SP TO USER'S ERROR CALL
MOVB #13,@(SP) ; WRITE ERROR NUMBER
SUB #2,(SP) ; MOVE SP TO RETURN IF ERROR
JSR PC,@(SP)+ ; REPORT ERROR AND RETURN
BR 30$

10$:
; REPORT ANY CONTROL BUS PARITY ERROR WHILE WRING REMOTE REGISTERS, I.E.,
; PAR=1 AND DPE = 0
BIT #PAR,RMER1I ; WAS THERE A PARITY ERROR??
BEQ 20$ ; NO!!
BIT #DPE,RMER2I ; WAS IT DUE TO CONTROL BUS??
BNE 20$ ; NO!!
MOV RMER1I,$GDDAT ; EXPECTED STATUS
BIC #PAR,$GDDAT
MOV RMER1I,$BDDAT ; RECEIVED STATUS
ADD #4,(SP) ; MOVE SP TO USER'S ERROR
MOVB #50,@(SP) ; WRITE ERROR NUMBER
BIT #MXF,RMCS1I ; DID MXF GET SET??
BNE 15$ ; YES!!
MOVB #274,@(SP) ; NO - CHANGE ERROR NUMBER
SUB #2,(SP) ; MOVE SP TO RETURN IF ERROR
JSR PC,@(SP)+ ; REPORT ERROR AND RETURN
BR 30$

15$:
20$:

```

```

10817
10818
10819
10820
10821
10822
10823 052510 032737 001000 001336
10824 052516 001425
10825 052520 013737 001336 001140
10826 052526 042737 001000 001140
10827 052534 013737 001336 001142
10828 052542 062716 000004
10829 052546 112776 000275 000000
10830 052554 162716 000002
10831 052560 004736
10832 052562
10833
10834
10835 052562 162716 000010
10836 052566 000137 055660
10837
10838 052572
10839
10840
10841
10842
10843 052572 032737 020000 001342
10844 052600 001447
10845 052602 013737 001342 001140
10846 052610 042737 020000 001140
10847 052616 013737 001342 001142
10848 052624 032737 010000 001340
10849 052632 001404
10850 052634 032737 000100 001340
10851 052642 001013
10852 052644 062716 000004
10853 052650 112776 000276 000000
10854 052656 162716 000002
10855 052662 004736
10856 052664 162716 000010
10857 052670 000413
10858 052672
10859
10860
10861
10862 052672 062716 000004
10863 052676 112776 000277 000000
10864 052704 162716 000002
10865 052710 004736
10866 052712 162716 000010
10867 052716 000240
10868 052720
10869
10870
10871 052720 032737 010000 001370
10872 052726 001432

```

```

;LOOK FOR ANY ERRORS WHICH MAY HAVE OCCURRED DURING COMMAND INITIATION OR
;MECHANICAL POSITIONING

;FIRST TEST MXF WHICH WOULD INDICATE COMPOSITE ERROR SET WHEN FUNCTION
;CODE AND GO BIT WERE LOADED
        BIT      #MXF,RMCS2I      ;WAS "MISSED TRANSFER" SET??
        BEQ      40$              ;NO!!
        MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
        BIC      #MXF,$GDDAT
        MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
        ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
        MOVB     #275,$2(SP)      ;WRITE ERROR NUMBER
        SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
        JSR      PC,$2(SP)+       ;REPORT ERROR AND RETURN

30$:
;RESTORE SP TO NO ERROR RETURN AND BYPASS FURTHER STATUS CHECKING
        SUB      #10,(SP)         ;MOVE SP TO NO ERROR
        JMP      380$             ;SKIP TO END OF SUB

40$:
;REPORT AN ERROR IF "OPI" ERROR OCCURRED DUE TO "MOL" = 0, OR IF "OPI"
;AND "MOL" ARE SET, BUT "VV" IS RESET, INDICATING AN INTERMITTENT
;"MOL"
        BIT      #OPI,RMER1I     ;IS "OPI" SET??
        BEQ      60$              ;NO!!
        MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
        BIC      #OPI,$GDDAT
        MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
        BIT      #MOL,RMDSI      ;WAS MEDIUM OFF LINE??
        BEQ      45$              ;YES!!
        BIT      #VV,RMDSI       ;WAS "MOL" INTERMITTENT??
        BNE      50$              ;NO!!
45$:
        ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
        MOVB     #276,$2(SP)      ;WRITE ERROR NUMBER IN CALL
        SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
        JSR      PC,$2(SP)+       ;REPORT ERROR AND RETURN
        SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
        BR       60$

50$:
;REPORT "OPI" ERROR, WHICH IS DUE TO "ON CYLINDER" NOT DROPPING OR
;"RUN" TIMEOUT (20 MS) OR SEARCH TIMEOUT (50 MS)
        ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
        MOVB     #277,$2(SP)      ;WRITE ERROR NUMBER IN CALL
        SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
        JSR      PC,$2(SP)+       ;REPORT ERROR AND RETURN
        SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
        NOP

60$:
;LOOK FOR "IVC" ERROR DURING COMMAND INITIATION
        BIT      #IVC,RMER2I     ;WAS THERE AN "IVC" ERROR??
        BEQ      70$              ;NO!!

```


10873
10874 052730 013737 001370 001140
10875 052736 042737 010000 001140
10876 052744 013737 001370 001142
10877 052752 062716 000004
10878 052756 112776 000300 000000
10879 052764 032737 000100 001340
10880 052772 001403
10881 052774 112776 000301 000000
10882 053002 162716 000002
10883 053006 004736
10884 053010 162716 000010
10885 053014
10886
10887
10888 053014 032737 000007 001342
10889 053022 001510
10890
10891 053024 032737 000002 001342
10892 053032 001424
10893 053034 013737 001342 001140
10894 053042 042737 000002 001140
10895 053050 013737 001342 001142
10896 053056 062716 000004
10897 053062 112776 000302 000000
10898 053070 162716 000002
10899 053074 004736
10900 053076 162716 000010
10901 053102 000240
10902 053104
10903
10904
10905 053104 032737 000001 001342
10906 053112 001424
10907 053114 013737 001342 001140
10908 053122 042737 000001 001140
10909 053130 013737 001342 001142
10910 053136 062716 000004
10911 053142 112776 000303 000000
10912 053150 162716 000002
10913 053154 004736
10914 053156 162716 000010
10915 053162 000240
10916 053164
10917
10918
10919 053164 032737 000004 001342
10920 053172 001424
10921 053174 013737 001342 001140
10922 053202 042737 000004 001140
10923 053210 013737 001342 001142
10924 053216 062716 000004
10925 053222 112776 000304 000000
10926 053230 162716 000002
10927 053234 004736
10928 053236 162716 000010

```

;REPORT "IVC" ERROR DUE TO "VV" = 0, OR REPORT ERRONEOUS "IVC" ERROR
MOV RMER2I,SGDDAT ;EXPECTED STATUS
BIC #IVC,SGDDAT
MOV RMER2I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR
MOVB #300,2(SP) ;WRITE ERROR NUMBER IN CALL
BIT #VV,RMSDI ;WAS VOLUME VALID??
BEQ 65$ ;NO!!
MOVB #301,2(SP) ;CHANGE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,2(SP)+ ;REPORT "IVC" ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR

70$:

;SEE IF "ILF" OR "RMR" IS SET
BIT #ILR!ILF!RMR,RMER1I
BEQ 100$ ;NO ERRORS DETECTED

;REPORT AN ERROR IF "ILR" IS SET
BIT #ILR,RMER1I ;WAS "ILR" DETECTED??
BEQ 80$ ;NO!!
MOV RMER1I,SGDDAT ;EXPECTED STATUS
BIC #ILR,SGDDAT
MOV RMER1I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #302,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR

80$:

;REPORT AN ERROR IF "ILF" IS SET
BIT #ILF,RMER1I ;WAS "ILF" DETECTED??
BEQ 90$ ;NO!!
MOV RMER1I,SGDDAT ;EXPECTED STATUS
BIC #ILF,SGDDAT
MOV RMER1I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #303,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR

90$:

;REPORT AN ERROR IF "RMR" IS SET
BIT #RMR,RMER1I ;WAS "RMR" DETECTED??
BEQ 100$ ;NO!!
MOV RMER1I,SGDDAT ;EXPECTED STATUS
BIC #RMR,SGDDAT
MOV RMER1I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #304,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR

```



```

10985 053554 000240
10986 053556
10987
10988
10989 053556 032737 006200 001370
10990 053564 001512
10991
10992
10993 053566 032737 000200 001370
10994 053574 001424
10995 053576 013737 001370 001140
10996 053604 042737 000200 001140
10997 053612 013737 001370 001142
10998 053620 062716 000004
10999 053624 112776 000310 000000
11000 053632 162716 000002
11001 053636 004736
11002 053640 162716 000010
11003 053644 000240
11004 053646
11005
11006
11007 053646 032737 002000 001370
11008 053654 001430
11009 053656 032737 010000 001340
11010 053664 001424
11011 053666 013737 001370 001140
11012 053674 042737 002000 001140
11013 053702 013737 001370 001142
11014 053710 062716 000004
11015 053714 112776 000311 000000
11016 053722 162716 000002
11017 053726 004736
11018 053730 162716 000010
11019 053734 000240
11020 053736
11021
11022
11023 053736 032737 004000 001370
11024 053744 001422
11025 053746 013737 001370 001140
11026 053754 042737 004000 001140
11027 053762 013737 001370 001142
11028 053770 062716 000004
11029 053774 112776 000312 000000
11030 054002 004736
11031 054004 162716 000010
11032 054010 000240
11033 054012
11034
11035
11036 054012 032737 054000 001342
11037 054020 001527
11038
11039 054022 032737 040000 001342
11040 054030 001427

NOP
150$:
;LOOK FOR "LSC" OR "LBC" OR "DVC" IN ERROR REGISTER #2
BIT #LSC!LBC!DVC,RMER2I
BEQ 180$ ;NO ERRORS SET

;REPORT ANY DEVICE FAULT, I.E., "DVC" = 1
BIT #DVC,RMER2I ;IS "DVC" = 1??
BEQ 160$ ;NO!!
MOV RMER2I,SGDDAT ;EXPECTED STATUS
BIC #DVC,SGDDAT
MOV RMER2I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USERS ERROR
MOVB #310,a(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR
NOP

160$:
;REPORT LOSS OF BIT CLOCK, I.E.; "LBC" = 1 IF "MOL" = 1
BIT #LBC,RMER2I ;IS LBC SET??
BEQ 170$ ;NO!!
BIT #MOL,RMDSI ;WAS LBC ERROR BY MOL = 0
BEQ 170$ ;YES!!
MOV RMER2I,SGDDAT ;EXPECTED STATUS
BIC #LBC,SGDDAT
MOV RMER2I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #311,a(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR
NOP

170$:
;REPORT LOS OF SYSTEM CLOCK, I.E. "LSC" = 1
BIT #LSC,RMER2I ;IS "LSC" = 1??
BEQ 180$ ;NO!!
MOV RMER2I,SGDDAT ;EXPECTED STATUS
BIC #LSC,SGDDAT
MOV RMER2I,SBDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #312,a(SP) ;WRITE ERROR NUMBER
JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;RESTORE SP TO NO ERROR
NOP

180$:
;LOOK FOR "UNS" OR "DTE" OR "WLE" IN ERROR REGISTER #1
BIT #UNS!DTE!WLE,RMER1I
BEQ 220$ ;NO BITS SET

;REPORT "UNS" ERROR IF "DVC" = 0
BIT #UNS,RMER1I ;IS "UNS" SET??
BEQ 190$ ;NO!!

```

```

11041 054032 032737 000200 001370      BIT      #DVC,RMER2I      ;WAS "UNS" CAUSED BY "DVC"??
11042 054040 001023                    BNE      190$          ;YES!!
11043 054042 013737 001342 001140      MOV      RMER1I,$GDDAT ;EXPECTED STATUS
11044 054050 042737 040000 001140      BIC      #UNS,$GDDAT
11045 054056 013737 001342 001142      MOV      RMER1I,$BDDAT ;RECEIVED STATUS
11046 054064 062716 000004                    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
11047 054070 112776 000313 000000      MOV      #313,$(SP)   ;WRITE ERROR NUMBER
11048 054076 162716 000002                    SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
11049 054102 004736                    JSR      PC,$(SP)+    ;REPORT ERROR AND RETURN
11050 054104 162716 000010                    SUB      #10,(SP)     ;RESTORE SP TO NO ERROR
11051 054110
11052
11053
11054 054110 032737 010000 001342      ;REPORT ANY DRIVE TIMING ERROR, I.E. "DTE" = 1
11055 054116 001423                    BIT      #DTE,RMER1I  ;IS DTE SET??
11056 054120 013737 001342 001140      BEQ      200$          ;NO!!
11057 054126 042737 010000 001140      MOV      RMER1I,$GDDAT ;EXPECTED STATUS
11058 054134 013737 001342 001142      BIC      #DTE,$GDDAT
11059 054142 062716 000004                    MOV      RMER1I,$BDDAT ;RECEIVED STATUS
11060 054146 112776 000314 000000      ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
11061 054154 162716 000002                    MOV      #314,$(SP)   ;WRITE ERROR NUMBER IN CALL
11062 054160 004736                    SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
11063 054162 162716 000010                    JSR      PC,$(SP)+    ;REPORT ERROR AND RETURN
11064 054166
11065
11066
11067
11068 054166 032737 004000 001342      ;REPORT AN ERROR IF WRITE LOCK ERROR IS SET. SEE IF DRIVE IS NOT
11069 054174 001441                    ;WRITE PROTECTED, OR IF FUNCTION WAS NOT A WRITE
11070 054176 013737 001342 001142      BIT      #MLE,RMER1I  ;WAS "MLE" SET??
11071 054204 013737 001342 001140      BEQ      220$          ;NO!!
11072 054212 052737 004000 001140      MOV      RMER1I,$BDDAT ;RECEIVED STATUS
11073 054220 062716 000004                    MOV      RMER1I,$GDDAT ;EXPECTED STATUS
11074 054224 112776 000315 000000      BIS      #MLE,$GDDAT
11075 054232 032737 004000 001340      ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
11076 054240 001404                    MOV      #315,$(SP)   ;WRITE ERROR NUMBER IN CALL
11077 054242 032737 000010 001376      BIT      #WRL,RMDSI   ;WAS DRIVE WRITE PROTECTED??
11078 054250 001406                    BEQ      205$          ;NO!!
11079 054252 112776 000316 000000      BIT      #BIT3,RMCS10 ;WAS COMMAND A WRITE??
11080 054260 042737 004000 001140      BEQ      210$          ;YES!!
11081 054266 162716 000002                    MOV      #316,$(SP)   ;CHANGE ERROR NUMBER
11082 054272 004736                    SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
11083 054274 162716 000010                    JSR      PC,$(SP)+    ;REPORT ERROR AND RETURN
11084
11085 054300
11086
11087
11088 054300 062716 000004                    ;OMIT DATA ERROR CHECKS IF ANY PREVIOUS ERRORS HAVE BEEN DETECTED
11089 054304 105776 000000                    ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR
11090 054310 001404                    TSTB    $(SP)         ;WAS ERROR DETECTED??
11091 054312 162716 000004                    BEQ      225$          ;NO - DO DATA CHECKS
11092 054316 000137 055320                    SUB      #4,(SP)       ;RESTORE SP
11093 054322 162716 000004                    JMP      340$          ;SKIP DATA CHECKS
11094
11095
11096
205$:  MOV      #316,$(SP)   ;CHANGE ERROR NUMBER
210$:  SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
        JSR      PC,$(SP)+ ;REPORT ERROR AND RETURN
        SUB      #10,(SP)  ;MOVE SP TO NO ERROR
220$:
225$:  SUB      #4,(SP)       ;RESTORE SP
;CHECK HEADER ERRORS IF FUNCTION WAS NOT WRITE HEADER AND DATA, AND
;IF HEADER COMPARE IS NOT INHIBITED

```

```

11097 054326 013737 001376 055710      MOV      RMCS10,510$      ;STRIP AND STORE FUNCTION CODE
11098 054334 042737 177700 055710      BIC      #+CFNCRASK,510$
11099 054342 022737 000063 055710      CMP      #MH!GO,510$     ;WAS FUNCTION WRITE HEADER & DATA??
11100 054350 001512                BEQ      250$             ;YES - SKIP HEADER CHECKS
11101 054352 032737 002000 001360      BIT      #HCI,RMOFI      ;WAS HCI SET??
11102 054360 001106                BNE      250$             ;YES - SKIP HEADER CHECKS
11103
11104                                     ;SEE IF ANY HEADER ERRORS ARE SET, I.E., "FER" OR "HCRC" OR "HCE"
11105 054362 032737 000620 001342      BIT      #HCRC!FER!HCE,RMER11
11106 054370 001533                BEQ      270$             ;NO ERRORS SET
11107
11108                                     ;REPORT HEADER CRC ERROR IF SET
11109 054372 032737 000400 001342      BIT      #HCRC,RMER11    ;WAS HCRC SET??
11110 054400 001422                BEQ      230$             ;NO!!
11111 054402 013737 001342 001140      MOV      RMER11,$GDDAT   ;EXPECTED STATUS
11112 054410 042737 000400 001140      BIC      #HCRC,$GDDAT
11113 054416 013737 001342 001142      MOV      RMER11,$BDDAT   ;RECEIVED STATUS
11114 054424 062716 000004                ADD      #4,(SP)         ;MOVE SP TO USERS ERROR
11115 054430 112776 000317 000000      MOVB    #317,2(SP)      ;WRITE ERROR NUMBER
11116 054436 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
11117 054442 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
11118 054444 000501
11119 054446
11120
11121                                     ;REPORT FORMAT ERROR IF SET
11122 054446 032737 000020 001342      BIT      #FER,RMER11    ;WAS "FER" SET??
11123 054454 001422                BEQ      240$             ;NO!!
11124 054456 013737 001342 001140      MOV      RMER11,$GDDAT   ;EXPECTED STATUS
11125 054464 042737 000020 001140      BIC      #FER,$GDDAT
11126 054472 013737 001342 001142      MOV      RMER11,$BDDAT   ;RECEIVED STATUS
11127 054500 062716 000004                ADD      #4,(SP)        ;MOVE SP TO USERS ERROR
11128 054504 112776 000320 000000      MOVB    #320,2(SP)      ;WRITE ERROR NUMBER
11129 054512 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
11130 054516 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
11131 054520 000453
11132 054522
11133
11134                                     ;REPORT HEADER COMPARE ERROR IF SET
11135 054522 032737 000200 001342      BIT      #HCE,RMER11    ;WAS "HCE" SET??
11136 054530 001453                BEQ      270$             ;NO!!
11137 054532 013737 001342 001140      MOV      RMER11,$GDDAT   ;EXPECTED STATUS
11138 054540 042737 000200 001140      BIC      #HCE,$GDDAT
11139 054546 013737 001342 001142      MOV      RMER11,$BDDAT   ;RECEIVED STATUS
11140 054554 062716 000004                ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR
11141 054560 112776 000321 000000      MOVB    #321,2(SP)      ;WRITE ERROR NUMBER
11142 054566 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
11143 054572 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
11144 054574 000425
11145
11146                                     ;THERE SHOULD BE NO HEADER ERRORS BECAUSE
11147                                     ;.COMMAND WAS WRITE HEADER AND DATA, OR
11148                                     ;.HEADER COMPARE INHIBIT WAS SET
11148 054576 032737 000620 001342      250$: BIT      #HCE!FER!HCRC,RMER11
11149 054604 001425                BEQ      270$             ;NO ERRORS WERE SET
11150 054606 013737 001342 001140      MOV      RMER11,$GDDAT   ;EXPECTED STATUS
11151 054614 042737 000620 001140      BIC      #HCE!FER!HCRC,$GDDAT
11152 054622 013737 001342 001142      MOV      RMER11,$BDDAT   ;RECEIVED STATUS

```

11153	054630	062716	000004		ADD	#4,(SP)	;	MOVE SP TO USER'S ERROR CALL	
11154	054634	112776	000322	000000	MOVB	#322,2(SP)	;	WRITE ERROR NUMBER	
11155	054642	162716	000002		SUB	#2,(SP)	;	MOVE SP TO RETURN IF ERROR	
11156	054646	004736			JSR	PC,2(SP)+	;	REPORT ERROR AND RETURN	
11157	054650	162716	000010		260\$:	SUB	#10,(SP)	;	MOVE SP TO NO ERROR
11158	054654	000137	055320		JMP	340\$;	OMIT FURTHER DATA CHECKS	
11159									
11160	054660				270\$:				
11161									
11162									
11163									
11164	054660	032737	000010	055710					
11165	054666	001002			BIT	#BIT3,510\$;	IF COMMAND WAS A WRITE COMMAND, GO DO WRITE ERROR CHECKS, OTHERWISE	
11166	054670	000137	055106		BNE	275\$;	DO READ ERROR CHECKS	
11167	054674				JMP	310\$;	WAS THIS A WRITE COMMAND?	
11168									
11169									
11170	054674	032737	100000	001342	275\$:				
11171	054702	001450							
11172	054704	013737	001342	001140					
11173	054712	042737	100000	001140					
11174	054720	013737	001342	001142					
11175	054726	062716	000004						
11176	054732	112776	000323	000000					
11177	054740	032737	004000	001360					
11178	054746	001021							
11179	054750	112776	000324	000000					
11180	054756	032737	000100	001342					
11181	054764	001007							
11182									
11183	054766	032737	000020	055710					
11184	054774	001406							
11185	054776	162716	000004						
11186	055002	000410							
11187	055004	112776	000325	000000					
11188	055012	162716	000002						
11189	055016	004736							
11190	055020	162716	000010						
11191									
11192	055024								
11193									
11194									
11195	055024	032737	000400	001336					
11196	055032	001423							
11197	055034	013737	001336	001140					
11198	055042	042737	000400	001140					
11199	055050	013737	001336	001142					
11200	055056	062716	000004						
11201	055062	112776	000326	000000					
11202	055070	162716	000002						
11203	055074	004736							
11204	055076	162716	000010						
11205	055102	000137	055320						
11206									
11207	055106								
11208									

```

11209 ;TEST TO SEE THAT OFFSET MODE WAS RESET; REPORT ERROR IF "OM" = 1
11210 055106 032737 000001 001340 BIT #OM,RMDSI ;IS OFFSET ON??
11211 055114 001423 BEQ 320$ ;NO
11212 055116 013737 001340 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
11213 055124 042737 000001 001140 BIC #OM,$GDDAT
11214 055132 013737 001340 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
11215 055140 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
11216 055144 112776 000327 000000 MOVB #327,2(SP) ;WRITE ERROR NUMBER IN CALL
11217 055152 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
11218 055156 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
11219 055160 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
11220 055164
11221
11222 ;TEST FOR DATA BUS PARITY ERROR; REPORT ERROR IF "DPE" = 1
11223 055164 032737 000010 001370 BIT #DPE,RMER2I ;DATA PARITY ERROR??
11224 055172 001423 BEQ 330$ ;NO!!
11225 055174 013737 001370 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
11226 055202 042737 000010 001140 BIC #DPE,$GDDAT
11227 055210 013737 001370 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
11228 055216 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
11229 055222 112776 000330 000000 MOVB #330,2(SP) ;WRITE ERROR NUMBER
11230 055230 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
11231 055234 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
11232 055236 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
11233 055242
11234
11235 ;TEST FOR WRITE CLOCK FAILURE; REPORT ERROR IF "WCF" = 1
11236 055242 032737 000040 001342 BIT #WCF,RMER1I ;IS "WCF" SET??
11237 055250 001423 BEQ 340$ ;NO!!
11238 055252 013737 001342 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
11239 055260 042737 000040 001140 BIC #WCF,$GDDAT
11240 055266 013737 001342 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
11241 055274 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
11242 055300 112776 000331 000000 MOVB #331,2(SP) ;WRITE ERROR NUMBER
11243 055306 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
11244 055312 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
11245 055314 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
11246 055320
11247
11248 ;REPORT "DATA LATE" ERROR IF "DLT" = 1
11249 055320 032737 100000 001336 BIT #DLT,RMCS2I ;IS "DLT" SET??
11250 055326 001423 BEQ 350$ ;NO!!
11251 055330 013737 001336 001140 MOV RMCS2I,$GDDAT ;EXPECTED STATUS
11252 055336 042737 100000 001140 BIC #DLT,$GDDAT
11253 055344 013737 001336 001142 MOV RMCS2I,$BDDAT ;RECEIVED STATUS
11254 055352 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
11255 055356 112776 000332 000000 MOVB #332,2(SP) ;WRITE ERROR NUMBER
11256 055364 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
11257 055370 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
11258 055372 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
11259 055376
11260 ;LOOK FOR UNEXPECTED CHANGES IN DRIVE STATUS
11261 055376 013746 001340 MOV RMDSI,-(SP) ;STACK DRIVE STATUS
11262 055402 042716 147677 BIC #1C<PIP!MOL!VV>,(SP) ;CLEAR DONT CARES
11263 055406 022726 010100 CMP #MOL!VV,(SP)+ ;IS DRIVE STATUS OK??
11264 055412 001522 BEQ 380$ ;YES!!

```

```

11265
11266
11267
11268 055414 032737 020000 001340
11269 055422 001430
11270 055424 032737 040000 001370
11271 055432 001024
11272 055434 013737 001340 001140
11273 055442 042737 020000 001140
11274 055450 013737 001340 001142
11275 055456 062716 000004
11276 055462 112776 000333 000000
11277 055470 162716 000002
11278 055474 004736
11279 055476 162716 000010
11280 055502 000240
11281 055504
11282
11283
11284
11285 055504 032737 010000 001340
11286 055512 001027
11287 055514 032737 020000 001342
11288 055522 001023
11289 055524 013737 001340 001140
11290 055532 052737 010000 001140
11291 055540 013737 001340 001142
11292 055546 062716 000004
11293 055552 112776 000334 000000
11294 055560 162716 000002
11295 055564 004736
11296 055566 162716 000010
11297 055572
11298
11299
11300
11301 055572 032737 000100 001340
11302 055600 001027
11303 055602 032737 010000 001370
11304 055610 001033
11305 055612 013737 001340 001140
11306 055620 052737 000100 001140
11307 055626 013737 001340 001142
11308 055634 062716 000004
11309 055640 112776 000335 000000
11310 055646 162716 000002
11311 055652 004736
11312 055654 162716 000010
11313 055660
11314
11315
11316 055660 062716 000004
11317 055664 105776 000000
11318 055670 001403
11319 055672 062716 000004
11320 055676 000402

;REPORT ERROR IF POSITIONING IN PROGRESS AND NO SEEK INCOMPLETE ERROR,
;I.E, PIP = 1 AND SKI = 0
      BIT      #PIP,RMSI      ;IS "PIP" SET??
      BEQ      360$          ;NO!!
      BIT      #SKI,RMER2I    ;WAS "SKI" ERROR REPORTED??
      BNE      360$          ;YES-DONT REPORT PIP
      MOV      RMSI,$GDDAT    ;EXPECTED STATUS
      BIC      #PIP,$GDDAT
      MOV      RMSI,$BDDAT    ;RECEIVED STATUS
      ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
      MOVB     #333,2(SP)     ;WRITE ERROR NUMBER
      SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
      JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
      SUB      #10,(SP)       ;MOVE SP TO NO ERROR
      NOP

360$:

;REPORT ERROR IF MEDIUM IS NOT ON LINE AND OPI ERROR WAS NOT
;REPORTED, I.E., MOL = OPI = 0
      BIT      #MOL,RMSI      ;IS MEDIUM ON LINE??
      BNE      370$          ;YES!!
      BIT      #OPI,RMER1I    ;WAS OPI ERROR REPORTED??
      BNE      370$          ;YES!!
      MOV      RMSI,$GDDAT    ;EXPECTED STATUS
      BIS      #MOL,$GDDAT
      MOV      RMSI,$BDDAT    ;RECEIVED STATUS
      ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR
      MOVB     #334,2(SP)     ;WRITE ERROR NUMBER
      SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
      JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
      SUB      #10,(SP)       ;MOVE SP TO NO ERROR

370$:

;REPORT ERROR IF VOLUME IS NOT VALID AND "IVC" ERROR WAS NOT
;REPORTED, I.E., VV = IVC = 0
      BIT      #VV,RMSI       ;IS VOLUME VALID??
      BNE      380$          ;YES!!
      BIT      #IVC,RMER2I    ;WAS IVC ERROR REPORTED??
      BNE      390$          ;YES!!
      MOV      RMSI,$GDDAT    ;EXPECTED STATUS
      BIS      #VV,$GDDAT
      MOV      RMSI,$BDDAT    ;RECEIVED STATUS
      ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
      MOVB     #335,2(SP)     ;WRITE ERROR NUMBER
      SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
      JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
      SUB      #10,(SP)       ;MOVE SP TO NO ERROR

380$:

;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS FOUND
      ADD      #4,(SP)        ;MOVE SP TO ERROR CALL
      TSTB     2(SP)          ;ANY ERROR??
      BEQ      390$          ;NO!!
      ADD      #4,(SP)        ;YES - MOVE SP TO ERROR RETURN
      BR       400$

```


DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 231
DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

SEQ 0233

11321	055700	162716	000004	390\$:	SUB	#4,(SP)	;MOVE SP TO NO ERROR RETURN
11322							
11323	055704	000207		400\$:	RTS	PC	;RETURN TO USER
11324							
11325	055706	000000		500\$:	.WORD		;ERROR FLAGS
11326	055710	000000		510\$:	.WORD		;TEMPORARY STORAGE

```

11327 .SBTTL STATIC DRIVE STATUS CHECK SUBROUTINE
11328
11329 ;THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
11330 ;STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID. THE SUBROUTINE
11331 ;CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
11332 ;SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.
11333
11334 ;THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
11335 ;IF TRUE:
11336
11337 .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
11338 ;THAT MOL IS ASSUMED TO HAVE BEEN SET
11339 .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
11340 .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
11341 .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
11342 .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT
11343
11344 ;THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.
11345
11346 ;(1) JSR PC,STCDRVSTS
11347 BR ??? RETURN HERE IF NO ERROR
11348 NOP RETURN HERE TO REPORT AN ERROR
11349 ERROR ERROR NUMBER DEFINED BY SUB
11350 JSR PC,2(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
11351 ??? RETURN HERE IF NO MORE ERRORS
11352
11353 STCDRVSTS:
11354
11355 ;CLEAR USER'S ERROR CALL
11356 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
11357 CLRB 2(SP) ;CLEAR ERROR NUMBER
11358 SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
11359
11360 ;SEE IF "MOL" = "VV" = 1, AND "PIP" = 0
11361 MOV RMDSI, -(SP) ;PUT DRIVE STATUS ON STACK
11362 BIC #1<PIP!MOL!VV>, (SP)
11363 CMP #MOL!VV, (SP)+ ;ARE MOL, VV AND PIP O.K.??
11364 BEQ 30$ ;YES!!
11365
11366 ;REPORT AN ERROR IF MOL = 0 AND "OPI" = 0
11367 BIT #MOL, RMDSI ;IS MOL ON ??
11368 BNE 10$ ;YES!!
11369 BIT #OPI, RMDSI ;WAS "OPI" SET??
11370 BNE 10$ ;YES-DONT REPORT "MOL" = 0
11371 MOV RMDSI, $GDDAT ;EXPECTED STATUS
11372 BIS #MOL, $GDDAT
11373 MOV RMDSI, $BDDAT ;RECEIVED STATUS
11374 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
11375 MOVB #207, 2(SP) ;WRITE ERROR NUMBER IN CALL
11376 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11377 JSR PC, 2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN

```

055712

055712 062716 000004

055716 105076 000000

055722 162716 000004

055726 013746 001340

055732 042716 147677

055736 022726 010100

055742 001524

055744 032737 010000 001340

055752 001030

055754 032737 020000 001342

055762 001024

055764 013737 001340 001140

055772 052737 010000 001140

056000 013737 001340 001142

056006 062716 000004

056012 112776 000207 000000

056020 162716 000002

056024 004736

056026 162716 000010

11378	056032	000240									
11379	056034										
11380											
11381											
11382	056034	032737	000100	001340							
11383	056042	001030									
11384	056044	032737	010000	001370							
11385	056052	001024									
11386	056054	013737	001340	001140							
11387	056062	052737	000100	001340							
11388	056070	013737	001340	001142							
11389	056076	062716	000004								
11390	056102	112776	000210	000000							
11391	056110	162716	000002								
11392	056114	004736									
11393	056116	162716	000010								
11394	056122	000240									
11395	056124										
11396											
11397											
11398	056124	032737	020000	001340							
11399	056132	001430									
11400	056134	032737	040000	001370							
11401	056142	001024									
11402	056144	013737	001340	001140							
11403	056152	042737	020000	001140							
11404	056160	013737	001340	001142							
11405	056166	062716	000004								
11406	056172	112776	000211	000000							
11407	056200	162716	000002								
11408	056204	004736									
11409	056206	162716	000010								
11410	056212	000240									
11411	056214										
11412											
11413											
11414	056214	013746	001370								
11415	056220	042726	137577								
11416	056224	001460									
11417	056226										
11418											
11419											
11420	056226	032737	000200	001370							
11421	056234	001424									
11422	056236	013737	001370	001140							
11423	056244	042737	000200	001140							
11424	056252	013737	001370	001142							
11425	056260	062716	000004								
11426	056264	112776	000212	000000							
11427	056272	162716	000002								
11428	056276	004736									
11429	056300	162716	000010								
11430	056304	000240									
11431	056306										
11432											
11433											

```

NOP
10$:
;REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND "IVC" = 0
BIT #VV,RMDSI ;IS "VV" = 0??
BNE 20$ ;NO!!
BIT #IVC,RMER2I ;WAS "IVC" SET??
BNE 20$ ;YES-DONT REPORT "VV" = 0
MOV RMDSI,SGDDAT ;EXPECTED STATUS
BIS #VV,RMDSI ;RECEIVED STATUS
MOV RMDSI,SBDDAT ;MOVE SP TO USER'S ERROR CALL
ADD #4,(SP) ;WRITE ERROR NUMBER IN CALL
MOVB #210,a(SP) ;MOVE SP TO RETURN FOR ERROR
SUB #2,(SP) ;REPORT ERROR VIA USER
JSR PC,a(SP)+ ;MOVE SP BACK TO NO ERROR
SUB #10,(SP)
NOP

20$:
;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND "SKI" = 0
BIT #PIP,RMDSI ;IS DRIVE OFF CYLINDER??
BEQ 30$ ;NO!!
BIT #SKI,RMER2I ;WAS "SKI" SET??
BNE 30$ ;YES-DONT REPORT "PIP" = 1
MOV RMDSI,SGDDAT ;EXPECTED STATUS
BIC #PIP,SGDDAT ;RECEIVED STATUS
MOV RMDSI,SBDDAT ;MOVE SP TO USER'S ERROR CALL
ADD #4,(SP) ;WRITE ERROR NUMBER IN USER'S CALL
MOVB #211,a(SP) ;MOVE SP TO RETURN FOR ERROR
SUB #2,(SP) ;REPORT ERROR VIA USER
JSR PC,a(SP)+ ;MOVE SP TO NO ERROR RETURN
SUB #10,(SP)
NOP

30$:
;SEE IF "SKI" = "DVC" = 0
MOV RMER2I,-(SP) ;PUT ERROR REG 2 ON STACK
BIC #1C<SKI!DVC>,(SP)+
BEQ 60$ ;BRANCH IF NO ERROR

40$:
;REPORT AN ERROR IF THERE IS A DEVICE FAULT
BIT #DVC,RMER2I ;ANY DEVICE FAULT??
BEQ 50$ ;NO!!
MOV RMER2I,SGDDAT ;EXPECTED STATUS
BIC #DVC,SGDDAT ;RECEIVED STATUS
MOV RMER2I,SBDDAT ;MOVE SP TO USER'S CALL
ADD #4,(SP) ;WRITE NUMBER OF ERROR IN CALL
MOVB #212,a(SP) ;MOVE SP TO RETURN FOR ERROR
SUB #2,(SP) ;REPORT ERROR VIA USER
JSR PC,a(SP)+ ;MOVE SP BACK TO NO ERP^R
SUB #10,(SP)
NOP

50$:
;REPORT AN ERROR IF "SKI" = 1

```

11434	056306	032737	040000	001370	BIT	#SKI,RMER2I	; IS THERE A SEEK INCOMPLETE ERROR
11435	056314	001424			BEQ	60\$; NO!!
11436	056316	013737	001370	001140	MOV	RMER2I,SGDDAT	; EXPECTED STATUS
11437	056324	042737	040000	001140	BIC	#SKI,SGDDAT	
11438	056332	013737	001370	001142	MOV	RMER2I,SBDDAT	; RECEIVED STATUS
11439	056340	062716	000004		ADD	#4,(SP)	; MOVE SP TO USER'S ERROR CALL
11440	056344	112776	000213	000000	MOVB	#213,a(SP)	; WRITE ERROR NUMBER IN USER'S ERROR CALL
11441	056352	162716	000002		SUB	#2,(SP)	; MOVE SP TO RETURN FOR ERROR
11442	056356	004736			JSR	PC,a(SP)+	; REPORT ERROR VIA USER
11443	056360	162716	000010		SUB	#10,(SP)	; MOVE SP BACK TO NO ERROR
11444	056364	000240			NOP		
11445	056366						
11446							
11447							
11448	056366	062716	000004		ADD	#4,(SP)	; MOVE SP TO USER'S ERROR CALL
11449	056372	105776	000000		TSTB	a(SP)	; WAS AN ERROR DETECTED??
11450	056376	001403			BEQ	70\$; NO!!
11451	056400	062716	000004		ADD	#4,(SP)	; YES - MOVE SP TO USER'S ERROR RETURN
11452	056404	000402			BR	80\$	
11453	056406	162716	000004		SUB	#4,(SP)	; NO - MOVE SP TO NO ERROR RETURN
11454	056412	000240			NOP		
11455	056414	000207			RTS	PC	; RETURN TO USER

```

11456
11457
11458
11459
11460
11461
11462
11463
11464
11465
11466
11467
11468
11469
11470
11471
11472
11473
11474
11475
11476 056416
11477
11478
11479 056416 013737 001342 001176
11480 056424 047637 000000 001176
11481 056432 062716 000002
11482 056436 013737 001370 001200
11483 056444 047637 000000 001200
11484
11485
11486
11487 056452 062716 000006
11488 056456 105076 000000
11489 056462 162716 000004
11490
11491
11492 056466 005737 001176
11493 056472 001420
11494 056474 013737 001176 001142
11495 056502 005037 001140
11496 056506 062716 000004
11497 056512 112776 000066 000000
11498 056520 162716 000002
11499 056524 004736
11500 056526 162716 000010
11501 056532 000240
11502 056534
11503
11504
11505
11506 056534 005737 001200
11507 056540 001420
11508
11509 056542 013737 001200 001142
11510 056550 005037 001140
11511 056554 062716 000004

```

```

.SBTTL COMPOSITE ERROR CHECK SUBROUTINE

;THIS SUBROUTINE CHECKS THE STORED CONTENTS OF RMER1 AND
;RMER2 AFTER MASKING EACH REGISTER WORD WITH THE USER'S STATUS
;MASKS AND REPORTS AN ERROR IF ANY BITS ARE LEFT ON AFTER
;THE MASKS ARE APPLIED.

;CALL:
;(1) JSR PC,CMPERRSTS
      .WORD MASK FOR ERROR REGISTER 1
      .WORD MASK FOR ERROR REGISTER 2
      BR ??? RETURN HERE IF NO ERROR
      NOP RETURN HERE TO REPORT AN ERROR
      ERROR ERROR NUMBER DEFINED BY SUB
      JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
      ??? RETURN HERE IF NO MORE ERRORS

;NOTE: BITS TO BE MASKED SHOULD BE ONE; BITS TO BE TESTED SHOULD
;BE ZERO

CMPERRSTS:
;MASK AND STORE THE CONTENTS OF RMER1 AND RMER2
      MOV RMER1I,$TMP1 ;STORE RMER1 AT TEMP STORAGE
      BIC @(SP),$TMP1 ;MASK RMER1
      ADD #2,(SP) ;MOVE SP TO NEXT MASK
      MOV RMER2I,$TMP2 ;STORE RMER2 AT TEMP STORAGE
      BIC @(SP),$TMP2 ;MASK RMER2

;CLEAR USER'S ERROR CALL
      ADD #6,(SP) ;MOVE SP TO USER'S ERROR CALL
      CLRB @(SP) ;CLEAR ERROR NUMBER
      SUB #4,(SP) ;LEAVE SP AT NO ERROR RETURN

;SEE IF THERE WERE ANY ERRORS IN RMER1, I.E. $TMP1
      TST $TMP1 ;ANY ERRORS TO REPORT??
      BEQ SS ;NO !!
      MOV $TMP1,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
      CLR $GDDAT ;EXPECTED STATUS FOR TYPEOUT
      ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
      MOVB #66,@(SP) ;CORRECTABLE DATA CHECK ERROR #
      SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
      JSR PC,@(SP)+ ;REPORT ERROR VIA USER
      SUB #10,(SP) ;MOVE SP BACK TO BRANCH
      NOP

SS:
;SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 ($TMP2)
      TST $TMP2 ;ANY ERRORS IN RMER2?
      BEQ 10$ ;NO!!
      MOV $TMP2,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
      CLR $GDDAT ;EXPECTED STATUS FOR TYPEOUT
      ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL

```

```

11512 056560 112776 000067 000000      MOVB    #67,2(SP)      ;WRITE ERROR NUMBER IN USER'S CALL
11513 056566 162716 000002              SUB     #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11514 056572 004736              JSR    PC,2(SP)+      ;REPORT ERROR VIA USER
11515 056574 162716 000010      SUB     #10,(SP)      ;MOVE SP TO NO ERROR RETURN
11516 056600 000240              NOP
11517 056602
11518
11519
11520 056602 062716 000004      ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
11521 056606 105776 000000      ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
11522 056612 001403              TSTB   2(SP)          ;WAS THERE AN ERROR CALLED??
11523 056614 062716 000004      BEQ    20$            ;NO!!
11524 056620 000402              ADD     #4,(SP)        ;YES - MOVE SP TO ERROR RETURN
11525 056622 162716 000004      BR     30$            ;MOVE SP TO NO ERROR RETURN
11526 056626 000207      20$:  SUB     #4,(SP)      ;RETURN TO USER
11527
11528      30$:  RTS    PC
    
```

```

11529 .SBTTL STOP AND SHUTDOWN SUBROUTINES
11530
11531 056630 STOP:
11532
11533 ;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT
11534 056630 012746 000140 MOV #PR3,-(SP) ;;PUT NEW PS ON STACK
11535 056634 012746 056642 MOV #64$,-(SP) ;;PUT NEW PC ON STACK
11536 056640 000002 RTI ;;POP NEW PC AND PS
11537 056642
11538 056642 000240 64$: NOP
11539 ;RAISE PRIORITY TO INHIBIT INTERRUPT
11540 056644 012746 000300 MOV #PR6,-(SP) ;;PUT NEW PS ON STACK
11541 056650 012746 056656 MOV #65$,-(SP) ;;PUT NEW PC ON STACK
11542 056654 000002 RTI ;;POP NEW PC AND PS
11543 056656
11544
11545 056656 000207 10$: RTS PC ;CONTINUE
11546
11547
11548 056660 SHUT:
11549 056660 104401 056702 TYPE 100$ ;TYPE THE HALT MESSAGE
11550 056664 005737 000042 TST 42 ;RUNNING STANDALONE ??
11551 056670 001402 BEQ 10$ ;YES !!
11552 056672 000137 032262 JMP SEOP ;NO - GO TO END OF PASS
11553 056676
11554 056676 000137 005322 10$: JMP START ;GO TO START
11555 056702 042524 052123 053440 100$: .ASCIZ @TEST WAS HALTED@<CR><LF><LF>
11556 056710 051501 044040 046101
11557 056716 042524 006504 005012
11558 056724 000
11559 056726 .EVEN

```

11560
 11561
 11562
 11563
 11564
 11565
 11566
 11567
 11568
 11569
 11570
 11571
 11572
 11573
 11574
 11575
 11576
 11577 056726
 11578 056726 010046
 11579 056730 010146
 11580 056732 010246
 11581 056734 010346
 11582 056736 010446
 11583 056740 010546
 11584 056742 016646 000022
 11585 056746 016646 000022
 11586 056752 016646 000022
 11587 056756 016646 000022
 11588 056762 000002
 11589
 11590
 11591
 11592
 11593 056764
 11594 056764 012666 000022
 11595 056770 012666 000022
 11596 056774 012666 000022
 11597 057000 012666 000022
 11598 057004 012605
 11599 057006 012604
 11600 057010 012603
 11601 057012 012602
 11602 057014 012601
 11603 057016 012600
 11604 057020 000002
 11605
 11606
 11607
 11608
 11609
 11610
 11611
 11612
 11613
 11614 057022 010146
 11615 057024 016601 000006

.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

```

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

```

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

;\$RESTORE RO-R5

```

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

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

```

:*****
:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
:;BINARY-ASCII NUMBER AND TYPE IT.
:CALL:
    
```

```

:* MOV NUMBER,-(SP) ;; NUMBER TO BE TYPED
:* TYPBN ;; TYPE IT
STYPBN: MOV R1,-(SP) ;; SAVE R1 ON THE STACK
MOV 6(SP),R1 ;; GET THE INPUT NUMBER
    
```



```

11616 057030 000261          SEC          ;; SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
11617 057032 112737 000060 057074 1$:  MOVB    #'0,$BIN  ;; SET CHARACTER TO AN ASCII "0".
11618 057040 006101          ROL     R1        ;; GET THIS BIT
11619 057042 001406          BEQ     2$        ;; DONE?
11620 057044 105537 057074  ADCB    $BIN      ;; NO--SET THE CHARACTER EQUAL TO THIS BIT
11621 057050 104401 057074  TYPE    , $BIN    ;; GO TYPE THIS BIT
11622 057054 000241          CLC          ;; CLEAR "C" SO CAN KEEP TRACK OF BITS
11623 057056 000765          BR      1$        ;; GO DO THE NEXT BIT
11624 057060 012601          2$:  MOV    (SP)+,R1  ;; POP THE STACK INTO R1
11625 057062 016666 000002 000004  MOV    2(SP),4(SP) ;; ADJUST THE STACK
11626 057070 012616          MOV    (SP)+,(SP)
11627 057072 000002          RTI          ;; RETURN TO USER
11628 057074      000      000  $BIN:  .BYTE  0,0      ;; STORAGE FOR ASCII CHAR. AND TERMINATOR
11629          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
11630
11631          ;; *****
11632          ;; THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
11633          ;; SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
11634          ;; NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
11635          ;; BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
11636          ;; REPLACED WITH SPACES.
11637          ;; CALL:
11638          ;; *   MOV    NUM,-(SP)      ;; PUT THE BINARY NUMBER ON THE STACK
11639          ;; *   TYPDS                    ;; GO TO THE ROUTINE
11640
11641          $TYPDS:
11642 057076 010046          MOV    R0,-(SP)  ;; PUSH R0 ON STACK
11643 057100 010146          MOV    R1,-(SP)  ;; PUSH R1 ON STACK
11644 057102 010246          MOV    R2,-(SP)  ;; PUSH R2 ON STACK
11645 057104 010346          MOV    R3,-(SP)  ;; PUSH R3 ON STACK
11646 057106 010546          MOV    R5,-(SP)  ;; PUSH R5 ON STACK
11647 057110 012746 020200  MOV    #20200,-(SP) ;; SET BLANK SWITCH AND SIGN
11648 057114 016605 000020  MOV    20(SP),R5  ;; GET THE INPUT NUMBER
11649 057120 100004          BPL    1$        ;; BR IF INPUT IS POS.
11650 057122 005405          NEG    R5        ;; MAKE THE BINARY NUMBER POS.
11651 057124 112766 000055 000001  MOVB   #'-,1(SP)  ;; MAKE THE ASCII NUMBER NEG.
11652 057132 005000          1$:  CLR    R0        ;; ZERO THE CONSTANTS INDEX
11653 057134 012703 057312  MOV    #SDBLK,R3  ;; SETUP THE OUTPUT POINTER
11654 057140 112723 000040  MOVB   #' ,(R3)+  ;; SET THE FIRST CHARACTER TO A BLANK
11655 057144 005002          2$:  CLR    R2        ;; CLEAR THE BCD NUMBER
11656 057146 016001 057302  MOV    $DTBL(R0),R1 ;; GET THE CONSTANT
11657 057152 160105          3$:  SUB    R1,R5     ;; FORM THIS BCD DIGIT
11658 057154 002402          BLT    4$        ;; BR IF DONE
11659 057156 005202          INC    R2        ;; INCREASE THE BCD DIGIT BY 1
11660 057160 000774          BR     3$
11661 057162 060105          4$:  ADD    R1,R5     ;; ADD BACK THE CONSTANT
11662 057164 005702          TST    R2        ;; CHECK IF BCD DIGIT=0
11663 057166 001002          BNE    5$        ;; FALL THROUGH IF 0
11664 057170 105716          TSTB  (SP)       ;; STILL DOING LEADING 0'S?
11665 057172 100407          BMI    7$        ;; BR IF YES
11666 057174 106316          5$:  ASLB  (SP)       ;; MSD?
11667 057176 103003          BCC    6$        ;; BR IF NO
11668 057200 116663 000001 177777  MOVB   1(SP),-1(R3) ;; YES--SET THE SIGN
11669 057206 052702 000060  6$:  BIS    #'0,R2    ;; MAKE THE BCD DIGIT ASCII
11670 057212 052702 000040  7$:  BIS    #' ,R2    ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
11671 057216 110223          MOVB  R2,(R3)+  ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER

```

H03

DZRNDA - RND3 FUNCTIONAL TEST, PART 2
 DZRNDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 240
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0242

11672	057220	005720			TST	(RO)+	:: JUST INCREMENTING
11673	057222	020027	000010		CMP	RO,#10	:: CHECK THE TABLE INDEX
11674	057226	002746			BLT	2\$:: GO DO THE NEXT DIGIT
11675	057230	003002			BGT	8\$:: GO TO EXIT
11676	057232	010502			MOV	R5,R2	:: GET THE LSD
11677	057234	000764			BR	6\$:: GO CHANGE TO ASCII
11678	057236	105726		8\$:	TSTB	(SP)+	:: WAS THE LSD THE FIRST NON-ZERO?
11679	057240	100003			BPL	9\$:: BR IF NO
11680	057242	116663	177777	177776	MOVB	-1(SP),-2(R3)	:: YES--SET THE SIGN FOR TYPING
11681	057250	105013		9\$:	CLRB	(R3)	:: SET THE TERMINATOR
11682	057252	012605			MOV	(SP)+,R5	:: POP STACK INTO R5
11683	057254	012603			MOV	(SP)+,R3	:: POP STACK INTO R3
11684	057256	012602			MOV	(SP)+,R2	:: POP STACK INTO R2
11685	057260	012601			MOV	(SP)+,R1	:: POP STACK INTO R1
11686	057262	012600			MOV	(SP)+,RO	:: POP STACK INTO RO
11687	057264	104401	057312		TYPE	\$DBLK	:: NOW TYPE THE NUMBER
11688	057270	016666	000002	000004	MOV	2(SP),4(SP)	:: ADJUST THE STACK
11689	057276	012616			MOV	(SP)+,(SP)	
11690	057300	000002			RTI		:: RETURN TO USER
11691	057302	023420			SDTBL:	10000.	
11692	057304	001750				1000.	
11693	057306	000144				100.	
11694	057310	000012				10.	
11695	057312	000004			SDBLK:	.BLKW 4	
11696					.SBTTL	BINARY TO OCTAL (ASCII) AND TYPE	

11697					*****			
11698					*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT			
11699					*OCTAL (ASCII) NUMBER AND TYPE IT.			
11700					*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE			
11701					*CALL:			
11702					MOV	NUM,-(SP)	:: NUMBER TO BE TYPED	
11703					TYPOS		:: CALL FOR TYPEOUT	
11704					.BYTE	N	:: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE	
11705					.BYTE	M	:: M=1 OR 0	
11706							:: 1=TYPE LEADING ZEROS	
11707							:: 0=SUPPRESS LEADING ZEROS	
11708								
11709								
11710					*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST			
11711					*STYPOS OR STYPOC			
11712					*CALL:			
11713					MOV	NUM,-(SP)	:: NUMBER TO BE TYPED	
11714					TYPON		:: CALL FOR TYPEOUT	
11715								
11716					*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER			
11717					*CALL:			
11718					MOV	NUM,-(SP)	:: NUMBER TO BE TYPED	
11719					TYPOC		:: CALL FOR TYPEOUT	
11720								
11721	057322	017646	000000		STYPOS:	MOV	2(SP),-(SP)	:: PICKUP THE MODE
11722	057326	116637	000001	057545		MOVB	1(SP),SOFILL	:: LOAD ZERO FILL SWITCH
11723	057334	112637	057547			MOVB	(SP)+,SOMODE+1	:: NUMBER OF DIGITS TO TYPE
11724	057340	062716	000002			ADD	82,(SP)	:: ADJUST RETURN ADDRESS
11725	057344	000406				BR	STYPON	
11726	057346	112737	000001	057545	STYPOC:	MOVB	81,SOFILL	:: SET THE ZERO FILL SWITCH
11727	057354	112737	000006	057547		MOVB	86,SOMODE+1	:: SET FOR SIX(6) DIGITS

```

11728 057362 112737 000005 057544 STYPON: MOVB #5,SOCNT      ;; SET THE ITERATION COUNT
11729 057370 010346          MOV R3,-(SP)      ;; SAVE R3
11730 057372 010446          MOV R4,-(SP)      ;; SAVE R4
11731 057374 010546          MOV R5,-(SP)      ;; SAVE R5
11732 057376 113704 057547  MOVB $OMODE+1,R4  ;; GET THE NUMBER OF DIGITS TO TYPE
11733 057402 005404          NEG R4
11734 057404 062704 000006  ADD #6,R4          ;; SUBTRACT IT FOR MAX. ALLOWED
11735 057410 110437 057546  MOVB R4,$OMODE    ;; SAVE IT FOR USE
11736 057414 113704 057545  MOVB $OFILL,R4    ;; GET THE ZERO FILL SWITCH
11737 057420 016605 000012  MOV 12(SP),R5    ;; PICKUP THE INPUT NUMBER
11738 057424 005003          CLR R3           ;; CLEAR THE OUTPUT WORD
11739 057426 006105          1S: ROL R5     ;; ROTATE MSB INTO "C"
11740 057430 000404          BR 3S          ;; GO DO MSB
11741 057432 006105          2S: ROL R5     ;; FORM THIS DIGIT
11742 057434 006105          ROL R5
11743 057436 006105          ROL R5
11744 057440 010503          MOV R5,R3
11745 057442 006103          3S: ROL R3     ;; GET LSB OF THIS DIGIT
11746 057444 105337 057546  DECB $OMODE      ;; TYPE THIS DIGIT?
11747 057450 100016          BPL 7S          ;; BR IF NO
11748 057452 042703 177770  BIC #177770,R3  ;; GET RID OF JUNK
11749 057456 001002          BNE 4S          ;; TEST FOR 0
11750 057460 005704          TST R4         ;; SUPPRESS THIS 0?
11751 057462 001403          BEQ 5S          ;; BR IF YES
11752 057464 005204          4S: INC R4     ;; DON'T SUPPRESS ANYMORE 0'S
11753 057466 052703 000060  BIS #'0,R3      ;; MAKE THIS DIGIT ASCII
11754 057472 052703 000040  5S: BIS #' ,R3  ;; MAKE ASCII IF NOT ALREADY
11755 057476 110337 057542  MOVB R3,$S      ;; SAVE FOR TYPING
11756 057502 104401 057542  TYPE #S        ;; GO TYPE THIS DIGIT
11757 057506 105337 057544  7S: DECB $OCNT  ;; COUNT BY 1
11758 057512 003347          BGT 2S          ;; BR IF MORE TO DO
11759 057514 002402          BLT 6S          ;; BR IF DONE
11760 057516 005204          INC R4         ;; INSURE LAST DIGIT ISN'T A BLANK
11761 057520 000744          BR 2S          ;; GO DO THE LAST DIGIT
11762 057522 012605          6S: MOV (SP)+,R5  ;; RESTORE R5
11763 057524 012604          MOV (SP)+,R4  ;; RESTORE R4
11764 057526 012603          MOV (SP)+,R3  ;; RESTORE R3
11765 057530 016666 000002 000004  MOV 2(SP),4(SP) ;; SET THE STACK FOR RETURNING
11766 057536 012616          MOV (SP)+,(SP)
11767 057540 000002          RTI          ;; RETURN
11768 057542          8S: .BYTE 0    ;; STORAGE FOR ASCII DIGIT
11769 057543          .BYTE 0    ;; TERMINATOR FOR TYPE ROUTINE
11770 057544          .BYTE 0    ;; OCTAL DIGIT COUNTER
11771 057545          .BYTE 0    ;; ZERO FILL SWITCH
11772 057546 000000          .WORD 0     ;; NUMBER OF DIGITS TO TYPE
11773          .SBTTL TYPE ROUTINE
11774
11775          ;; *****
11776          ;; *ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
11777          ;; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
11778          ;; *NOTE1: SNUL contains the character to be used as the filler character.
11779          ;; *NOTE2: SFILLS contains the number of filler characters required.
11780          ;; *NOTE3: SFILLC contains the character to fill after.
11781          ;; *
11782          ;; *CALL:
11783          ;; *1) USING A TRAP INSTRUCTION
    
```

```

11784      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
11785      ;*OR
11786      ;*      TYPE
11787      ;*      MESADR
11788      ;*
11789
11790 057550 105737 001173      STYPE:  TSTB      STPFLG      ;; IS THERE A TERMINAL?
11791 057554 100002      BPL      1S      ;; BR IF YES
11792 057556 000000      HALT      ;; HALT HERE IF NO TERMINAL
11793 057560 000430      BR      3S      ;; LEAVE
11794 057562 010046      1S:  MOV      RO,-(SP)      ;; SAVE RO
11795 057564 017600 000002      MOV      22(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
11796 057570 122737 000001 001242      CMPB     #APTENV,SENV      ;; RUNNING IN APT MODE
11797 057576 001011      BNE     62S      ;; NO, GO CHECK FOR APT CONSOLE
11798 057600 132737 000100 001243      BITB     #APTPOOL,SENV      ;; SPOOL MESSAGE TO APT
11799 057606 001405      BEQ     62S      ;; NO, GO CHECK FOR CONSOLE
11800 057610 010037 057620      MOV      RO,61S      ;; SETUP MESSAGE ADDRESS FOR APT
11801 057614 004737 062474      JSR     PC,SATY3      ;; SPOOL MESSAGE TO APT
11802 057620 000000      61S:  .WORD     0      ;; MESSAGE ADDRESS
11803 057622 132737 000040 001243      62S:  BITB     #APTCSUP,SENV      ;; APT CONSOLE SUPPRESSED
11804 057630 001003      BNE     60S      ;; YES, SKIP TYPE OUT
11805 057632 112046      2S:  MOVB     (RO)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
11806 057634 001005      BNE     4S      ;; BR IF IT ISN'T THE TERMINATOR
11807 057636 005726      TST     (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
11808 057640 012600      60S:  MOV      (SP)+,RO      ;; RESTORE RO
11809 057642 062716 000002      3S:  ADD      #2,(SP)      ;; ADJUST RETURN PC
11810 057646 000002      RTI
11811 057650 122716 000011      4S:  CMPB     #HT,(SP)      ;; BRANCH IF <HT>
11812 057654 001430      BEQ     8S
11813 057656 122716 000200      CMPB     #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
11814 057662 001006      BNE     5S
11815 057664 005726      TST     (SP)+      ;; POP <CR><LF> EQUIV
11816 057666 104401      TYPE
11817 057670 001217      SCRLF
11818 057672 105037 060026      CLRB     $CHARCNT      ;; CLEAR CHARACTER COUNT
11819 057676 000755      BR      2S      ;; GET NEXT CHARACTER
11820 057700 004737 057762      5S:  JSR     PC,$TYPEC      ;; GO TYPE THIS CHARACTER
11821 057704 123726 001172      6S:  CMPB     $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
11822 057710 001350      BNE     2S      ;; IF NO GO GET NEXT CHAR.
11823 057712 013746 001170      MOV      $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
11824      AND     THE NULL CHAR.
11825 057716 105366 000001      7S:  DECB     1(SP)      ;; DOES A NULL NEED TO BE TYPED?
11826 057722 002770      BLT     6S      ;; BR IF NO--GO POP THE NULL OFF OF STACK
11827 057724 004737 057762      JSR     PC,$TYPEC      ;; GO TYPE A NULL
11828 057730 105337 060026      DECB     $CHARCNT      ;; DO NOT COUNT AS A COUNT
11829 057734 000770      BR      7S      ;; LOOP
11830
11831      ;HORIZONTAL TAB PROCESSOR
11832
11833 057736 112716 000040      8S:  MOVB     #' ,(SP)      ;; REPLACE TAB WITH SPACE
11834 057742 004737 057762      9S:  JSR     PC,$TYPEC      ;; TYPE A SPACE
11835 057746 132737 000007 060026      BITB     #7,$CHARCNT      ;; BRANCH IF NOT AT
11836 057754 001372      BNE     9S      ;; TAB STOP
11837 057756 005726      TST     (SP)+      ;; POP SPACE OFF STACK
11838 057760 000724      BR      2S      ;; GET NEXT CHARACTER
11839 057762 105777 121176      STYPEC: TSTB     $STPS      ;; WAIT UNTIL PRINTER IS READY
    
```

```

11840 057766 100375          BPL      STYPEC
11841 057770 116677 000002 121170  MOVB    2(SP),2STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
11842 057776 122766 000015 000002  CMPB    #CR,2(SP)   ;;IS CHARACTER A CARRIAGE RETURN?
11843 060004 001003          BNE     1$         ;;BRANCH IF NO
11844 060006 105037 060026          CLRB    $CHARCNT   ;;YES--CLEAR CHARACTER COUNT
11845 060012 000406          BR     STYPEX      ;;EXIT
11846 060014 122766 000012 000002 1$:  CMPB    #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
11847 060022 001402          BEQ    STYPEX      ;;BRANCH IF YES
11848 060024 105227          INCB   (PC)+      ;;COUNT THE CHARACTER
11849 060026 000000          $CHARCNT: .WORD  0 ;;CHARACTER COUNT STORAGE
11850 060030 000207          STYPEX: RTS      PC

11851
11852          .SBTTL  SCOPE HANDLER ROUTINE
11853
11854          ;;*****
11855          ;;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
11856          ;;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
11857          ;;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
11858          ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
11859          ;;SM14=1      LOOP ON TEST
11860          ;;SM11=1      INHIBIT ITERATIONS
11861          ;;SM09=1      LOOP ON ERROR
11862          ;;SM08=1      LOOP ON TEST IN SWR<7:0>
11863          ;;CALL
11864          ;;          SCOPE          ;;SCOPE=IOT
11865
11866          $SCOPE:
11867 060032 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
11868 060034 004737 056630 121106  JSR     PC_STOP
11869 060040 032777 040000          BIT     #BIT14,$SWR  ;;LOOP ON PRESENT TEST?
11870 060046 001131          BNE     $OVER      ;;YES IF SM14=1
11871          ;;#####START OF CODE FOR THE XOR TESTER#####
11872 060050 000416          $XTSTR: BR     6$   ;;IF RUNNING ON THE "XOR" TESTER CHANGE
11873          ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
11874 060052 013746 000004          MOV     2#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
11875 060056 012737 060076 000004          MOV     #5$,$ERRVEC  ;;SET FOR TIMEOUT
11876 060064 005737 177060          TST    2#177060     ;;TIME OUT ON XOR?
11877 060070 012637 000004          MOV     (SP)+,2#ERRVEC ;;RESTORE THE ERROR VECTOR
11878 060074 000500          BR     $$VLAD      ;;GO TO THE NEXT TEST
11879 060076 022626          5$:  CMP     (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
11880 060100 012637 000004          MOV     (SP)+,2#ERRVEC ;;RESTORE THE ERROR VECTOR
11881 060104 000440          BR     7$         ;;LOOP ON THE PRESENT TEST
11882 060106          6$:;#####END OF CODE FOR THE XOR TESTER#####
11883 060106 032777 000400 121040          BIT     #BIT08,$SWR  ;;LOOP ON SPEC. TEST?
11884 060114 001421          BEQ    2$         ;;BR IF NO
11885 060116 005046          CLR    -(SP)     ;;CLEAR A TEMP. LOCATION
11886 060120 117716 121030          MOVB   2$SWR,(SP)   ;;PICKUP THE DESIRED TEST NUMBER
11887 060124 001414          BEQ    8$         ;;BRANCH IF BAD TEST NUMBER IN SWR
11888 060126 022716 000035          CMP    #35,(SP)   ;;CHECK THE NUMBER IN THE SWR
11889 060132 002411          BLT    8$         ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
11890 060134 011637 001116          MOV    (SP),$TSTNM ;;UPDATE THE TEST NUMBER
11891 060140 005316          DEC    (SP)      ;;BACKUP BY ONE
11892 060142 006316          ASL    (SP)      ;;SCALE THE TEST NUMBER AS AN INDEX
11893 060144 062716 060350          ADD    #$$SW08TBL,(SP) ;;FORM THE ADDRESS OF TEST POINTER
11894 060150 013637 001122          MOV    2(SP)+,$LPRDR ;;SET LOOP ADDRESS TO DESIRED TEST
11895 060154 000466          BR     $OVER      ;;GO LOOP ON THE TEST

```

11896	060156	005726		8S:	TST	(SP)+	::	CLEAN THE BAD TEST NUMBER OFF OF THE STACK
11897	060160	105737	001117	2S:	TSTB	SERFLG	::	HAS AN ERROR OCCURRED?
11898	060164	001421			BEQ	3S	::	BR IF NO
11899	060166	123737	001131 001117		CMPB	SERMAX,SERFLG	::	MAX. ERRORS FOR THIS TEST OCCURRED?
11900	060174	101015			BHI	3S	::	BR IF NO
11901	060176	032777	001000 120750		BIT	#BIT09,ASWR	::	LOOP ON ERROR?
11902	060204	001404			BEQ	4S	::	BR IF NO
11903	060206	013737	001124 001122	7S:	MOV	SLPERR,SLPADR	::	SET LOOP ADDRESS TO LAST SCOPE
11904	060214	000446			BR	SOVER		
11905	060216	105037	001117	4S:	CLRB	SERFLG	::	ZERO THE ERROR FLAG
11906	060222	005037	001206		CLR	STIMES	::	CLEAR THE NUMBER OF ITERATIONS TO MAKE
11907	060226	000415			BR	1S	::	ESCAPE TO THE NEXT TEST
11908	060230	032777	004000 120716	3S:	BIT	#BIT11,ASWR	::	INHIBIT ITERATIONS?
11909	060236	001011			BNE	1S	::	BR IF YES
11910	060240	005737	001230		TST	SPASS	::	IF FIRST PASS OF PROGRAM
11911	060244	001406			BEQ	1S	::	INHIBIT ITERATIONS
11912	060246	005237	001120		INC	SICNT	::	INCREMENT ITERATION COUNT
11913	060252	023737	001206 001120		CMP	STIMES,SICNT	::	CHECK THE NUMBER OF ITERATIONS MADE
11914	060260	002024			BGE	SOVER	::	BR IF MORE ITERATION REQUIRED
11915	060262	012737	000001 001120	1S:	MOV	#1,SICNT	::	REINITIALIZE THE ITERATION COUNTER
11916	060270	013737	060346 001206		MOV	SMXCNT,STIMES	::	SET NUMBER OF ITERATIONS TO DO
11917	060276	105237	001116	SSVLAD:	INCB	STSTNM	::	COUNT TEST NUMBERS
11918	060302	113737	001116 001226		MOVB	STSTNM,STSTNM	::	SET TEST NUMBER IN APT MAILBOX
11919	060310	011637	001122		MOV	(SP),SLPADR	::	SAVE SCOPE LOOP ADDRESS
11920	060314	011637	001124		MOV	(SP),SLPERR	::	SAVE ERROR LOOP ADDRESS
11921	060320	005037	001210		CLR	SESCAPE	::	CLEAR THE ESCAPE FROM ERROR ADDRESS
11922	060324	112737	000001 001131		MOVB	#1,SERMAX	::	ONLY ALLOW ONE(1) ERROR ON NEXT TEST
11923	060332	013777	001116 120616	SOVER:	MOV	STSTNM,DISP	::	DISPLAY TEST NUMBER
11924	060340	013716	001122		MOV	SLPADR,(SP)	::	FUDGE RETURN ADDRESS
11925	060344	000002			RTI		::	FIXES PS
11926	060346	000012		SMXCNT:	10.		::	MAX. NUMBER OF ITERATIONS
11927	060350			SSWOBTBL:				
11928	060350	007020			.WORD	TST1+2	::	STARTING ADDRESS OF TEST 1
11929	060352	007216			.WORD	TST2+2	::	STARTING ADDRESS OF TEST 2
11930	060354	007402			.WORD	TST3+2	::	STARTING ADDRESS OF TEST 3
11931	060356	007544			.WORD	TST4+2	::	STARTING ADDRESS OF TEST 4
11932	060360	010344			.WORD	TST5+2	::	STARTING ADDRESS OF TEST 5
11933	060362	011144			.WORD	TST6+2	::	STARTING ADDRESS OF TEST 6
11934	060364	011760			.WORD	TST7+2	::	STARTING ADDRESS OF TEST 7
11935	060366	012534			.WORD	TST10+2	::	STARTING ADDRESS OF TEST 10
11936	060370	013454			.WORD	TST11+2	::	STARTING ADDRESS OF TEST 11
11937	060372	014254			.WORD	TST12+2	::	STARTING ADDRESS OF TEST 12
11938	060374	015030			.WORD	TST13+2	::	STARTING ADDRESS OF TEST 13
11939	060376	015746			.WORD	TST14+2	::	STARTING ADDRESS OF TEST 14
11940	060400	016522			.WORD	TST15+2	::	STARTING ADDRESS OF TEST 15
11941	060402	017276			.WORD	TST16+2	::	STARTING ADDRESS OF TEST 16
11942	060404	020052			.WORD	TST17+2	::	STARTING ADDRESS OF TEST 17
11943	060406	020654			.WORD	TST20+2	::	STARTING ADDRESS OF TEST 20
11944	060410	021400			.WORD	TST21+2	::	STARTING ADDRESS OF TEST 21
11945	060412	022124			.WORD	TST22+2	::	STARTING ADDRESS OF TEST 22
11946	060414	022646			.WORD	TST23+2	::	STARTING ADDRESS OF TEST 23
11947	060416	023144			.WORD	TST24+2	::	STARTING ADDRESS OF TEST 24
11948	060420	023442			.WORD	TST25+2	::	STARTING ADDRESS OF TEST 25
11949	060422	024170			.WORD	TST26+2	::	STARTING ADDRESS OF TEST 26
11950	060424	024716			.WORD	TST27+2	::	STARTING ADDRESS OF TEST 27
11951	060426	025444			.WORD	TST30+2	::	STARTING ADDRESS OF TEST 30

```

11952 060430 026336 .WORD TST31+2 ;: STARTING ADDRESS OF TEST 31
11953 060432 027144 .WORD TST32+2 ;: STARTING ADDRESS OF TEST 32
11954 060434 027666 .WORD TST33+2 ;: STARTING ADDRESS OF TEST 33
11955 060436 030410 .WORD TST34+2 ;: STARTING ADDRESS OF TEST 34
11956 060440 031416 .WORD TST35+2 ;: STARTING ADDRESS OF TEST 35
11957 .SBTTL ERROR HANDLER ROUTINE
11958
11959 ;: *****
11960 ;: *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
11961 ;: *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
11962 ;: *AND GO TO ERRTP ON ERROR
11963 ;: *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
11964 ;: *SW15=1 HALT ON ERROR
11965 ;: *SW13=1 INHIBIT ERROR TYPEOUTS
11966 ;: *SW10=1 BELL ON ERROR
11967 ;: *SW09=1 LOOP ON ERROR
11968 ;: *CALL
11969 ;: * ERROR N ;: ERROR=EMT AND N=ERROR ITEM NUMBER
11970
11971 060442 SERROR:
11972 060442 104410 001117 7$: CKSWR ;: TEST FOR CHANGE IN SOFT-SWR
11973 060444 105237 001117 INCB SERFLG ;: SET THE ERROR FLAG
11974 060450 001775 001116 120476 MOV $STNM, $DISPLAY ;: DON'T LET THE FLAG GO TO ZERO
11975 060452 013777 002000 120466 BIT $BIT10, $SWR ;: DISPLAY TEST NUMBER AND ERROR FLAG
11976 060460 032777 001402 BEQ 1$ ;: BELL ON ERROR?
11977 060466 001402 TYPE $BELL ;: NO - SKIP
11978 060470 104401 001212 1$: INC $ERTTL ;: RING BELL
11979 060474 005237 001126 MOV (SP), $ERRPC ;: COUNT THE NUMBER OF ERRORS
11980 060500 011637 001132 SUB #2, $ERRPC ;: GET ADDRESS OF ERROR INSTRUCTION
11981 060504 162737 000002 001132 MOVVB $ERRPC, $ITEMB ;: STRIP AND SAVE THE ERROR ITEM CODE
11982 060512 117737 120414 001130 BIT $BIT13, $SWR ;: SKIP TYPEOUT IF SET
11983 060520 032777 020000 120426 BNE 20$ ;: SKIP TYPEOUTS
11984 060526 001004 JSR PC, ERRTP ;: GO TO USER ERROR ROUTINE
11985 060530 004737 032472 TYPE , $CALF
11986 060534 104401 001217 20$: CMPB $APTENV, $ENV ;: RUNNING IN APT MODE
11987 060540 122737 000001 001242 BNE 2$ ;: NO, SKIP APT ERROR REPORT
11988 060540 001007 MOVVB $ITEMB, 21$ ;: SET ITEM NUMBER AS ERROR NUMBER
11989 060546 001007 JSR PC, SATY4 ;: REPORT FATAL ERROR TO APT
11990 060550 113737 001130 060562 21$: .BYTE 0
11991 060556 004737 062504 .BYTE 0
11992 060562 000 .BYT 0
11993 060563 000 .BYT 0
11994 060564 000777 22$: BR 22$ ;: APT ERROR LOOP
11995 060566 005777 120362 2$: TST $SWR ;: HALT ON ERROR
11996 060572 100002 BPL 3$ ;: SKIP IF CONTINUE
11997 060574 000000 HALT ;: HALT ON ERROR!
11998 060576 104410 CKSWR ;: TEST FOR CHANGE IN SOFT-SWR
11999 060600 032777 001000 120346 3$: BIT $BIT09, $SWR ;: LOOP ON ERROR SWITCH SET?
12000 060606 001402 BEQ 4$ ;: BR IF NO
12001 060610 013716 001124 MOV $LPERR, (SP) ;: FUDGE RETURN FOR LOOPING
12002 060614 005737 001210 4$: TST $ESCAPE ;: CHECK FOR AN ESCAPE ADDRESS
12003 060620 001402 BEQ 5$ ;: BR IF NONE
12004 060622 013716 001210 MOV $ESCAPE, (SP) ;: FUDGE RETURN ADDRESS FOR ESCAPE
12005 060626 022737 032452 000042 5$: CMP $SENDAD, $42 ;: ACT-11 AUTO-ACCEPT?
12006 060626 001001 BNE 6$ ;: BRANCH IF NO
12007 060634 001001

```

```

12008 060636 000000          HALT                ;; YES
12009 060640                6S:                ;;
12010 060640 000002          RTI                 ;; RETURN
12011                                .SBTTL  TTY INPUT ROUTINE
12012
12013                                ;:*****
12014                                .ENABL  LSB
12015 060642 000000          $TKCNT: .WORD 0                ;; NUMBER OF ITEMS IN QUEUE
12016 060644 000000          $TKQIN: .WORD 0              ;; INPUT POINTER
12017 060646 000000          $TKQOUT: .WORD 0             ;; OUTPUT POINTER
12018 060650 000001          $TKQSRV: .BLKB 1            ;; TTY KEYBOARD QUEUE
12019                                $TKQEND=.
12020                                .EVEN
12021
12022                                ;: #TK INITIALIZE ROUTINE
12023                                ;: #THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
12024                                ;: #SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
12025
12026                                ;: #CALL:
12027                                ;:      JSR      PC, $TKINT
12028                                ;:      RETURN
12029
12030 060652 005037 060642          $TKINT: CLR      $TKCNT                ;; CLEAR COUNT OF ITEMS IN QUEUE
12031 060656 012737 060650 060644  MOV      $TKQSRV, $TKQIN          ;; MOVE THE STARTING ADDRESS OF THE
12032 060664 013737 060644 060646  MOV      $TKQIN, $TKQOUT         ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
12033 060672 012737 060722 000060  MOV      $TKSRV, $TKVEC         ;; INITIALIZE THE KEYBOARD VECTOR
12034 060700 012737 000200 000062  MOV      $200, $TKVEC+2         ;; "BR" LEVEL 4
12035 060706 005777 120250                TST      $TKB                    ;; CLEAR DONE FLAG
12036 060712 012777 000100 120240          MOV      $100, $TKS             ;; ENABLE TTY KEYBOARD INTERRUPT
12037 060720 000207                RTS      PC                       ;; RETURN TO CALLER
12038
12039                                ;: #TK SERVICE ROUTINE
12040                                ;: #THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
12041                                ;: #BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
12042                                ;: #IT IN THE QUEUE.
12043                                ;: #IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
12044                                ;: #UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS ($HUT)
12045
12046 060722 117746 120234          $TKSRV: MOVB     $TKB, -(SP)        ;; PICKUP THE CHARACTER
12047 060726 042716 177600          BIC      $1C177, (SP)          ;; STRIP THE JUNK
12048 060732 021627 000003          CMP      (SP), #3              ;; IS IT A CONTROL C?
12049 060736 001007                BNE      1$                     ;; BRANCH IF NO
12050 060740 104401 062036          TYPE     $CNTLC                ;; TYPE A CONTROL-C (^C)
12051 060744 004737 060652          JSR      PC, $TKINT            ;; INIT THE KEYBOARD
12052 060750 005726                TST      (SP)+                  ;; CLEAN UP STACK
12053 060752 000137 056660          JMP      $HUT                  ;; CONTROL C RESTART
12054 060756 021627 000007          1$:    CMP      (SP), #7         ;; IS IT A CONTROL G?
12055 060762 001004                BNE      2$                     ;; BRANCH IF NO
12056 060764 022737 000176 001154          CMP      $SWREG, $SWR          ;; IS SOFT-SWR SELECTED?
12057 060772 001500                BEQ      6$                     ;; GO TO SWR CHANGE
12058
12059                                2$:
12060 060774 022737 000001 060642          CMP      #1, $TKCNT            ;; IS THE QUEUE FULL?
12061 061002 001004                BNE      3$                     ;; BRANCH IF NO
12062 061004 104401 001212          TYPE     $BELL                  ;; RING THE TTY BELL
12063 061010 005726                TST      (SP)+                  ;; CLEAN CHARACTER OFF OF STACK

```


12064	061012	000451			BR	5S	::EXIT
12065	061014	021627	000023		3S: CMP	(SP),#23	::IS IT A CONTROL-S?
12066	061020	001021			BNE	32S	::BRANCH IF NO
12067	061022	005077	120132		CLR	2STKS	::DISABLE TTY KEYBOARD INTERRUPTS
12068	061026	005726			TST	(SP)+	::CLEAN CHAR OFF STACK
12069	061030	105777	120124		31S: TSTB	2STKS	::WAIT FOR A CHAR
12070	061034	100375			BPL	31S	::LOOP UNTIL ITS THERE
12071	061036	117746	120120		MOVB	2STKB, -(SP)	::GET THE CHARACTER
12072	061042	042716	177600		BIC	#1C177 (SP)	::MAKE IT 7-BIT ASCII
12073	061046	022627	000021		CMP	(SP)+, #21	::IS IT A CONTROL-Q?
12074	061052	001366			BNE	31S	::BRANCH IF NO
12075	061054	012777	000100	120076	MOV	#100, 2STKS	::REENABLE TTY KEYBOARD INTERRUPTS
12076	061062	000002			RTI		::RETURN
12077	061064	005237	060642		32S: INC	STKCNT	::COUNT THIS CHARACTER
12078	061070	021627	000140		CMP	(SP), #140	::IS IT UPPER CASE?
12079	061074	002405			BLT	4S	::BRANCH IF YES
12080	061076	021627	000175		CMP	(SP), #175	::IS IT A SPECIAL CHAR?
12081	061102	003002			BGT	4S	::BRANCH IF YES
12082	061104	042716	000040		BIC	#40, (SP)	::MAKE IT UPPER CASE
12083	061110	112677	177530		4S: MOVB	(SP)+, 2STKQIN	::AND PUT IT IN QUEUE
12084	061114	005237	060644		INC	STKQIN	::UPDATE THE POINTER
12085	061120	023727	060644	060651	CMP	STKQIN, #STKQEND	::GO OFF THE END?
12086	061126	001003			BNE	5S	::BRANCH IF NO
12087	061130	012737	060650	060644	MOV	#STKQSR, STKQIN	::RESET THE POINTER
12088	061136	000002			5S: RTI		::RETURN

```

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

```

12095	061140	022737	000176	001154	SCKSWR: CMP	#SWREG, SWR	::IS THE SOFT-SWR SELECTED
12096	061146	001124			BNE	15S	::EXIT IF NOT
12097	061150	105777	120004		TSTB	2STKS	::IS A CHAR WAITING?
12098	061154	100121			BPL	15S	::IF NOT, EXIT
12099	061156	117746	120000		MOVB	2STKB, -(SP)	::YES
12100	061162	042716	177600		BIC	#1C177 (SP)	::MAKE IT 7-BIT ASCII
12101	061166	021627	000007		CMP	(SP), #7	::IS IT A CONTROL-G?
12102	061172	001300			BNE	2S	::IF NOT, PUT IT IN THE TTY QUEUE
12103							::AND EXIT

```

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

```

12109	061174	123727	001150	000001	6S: CMPB	\$AUTOB, #1	::ARE WE RUNNING IN AUTO-MODE?
12110	061202	001674			BEQ	2S	::BRANCH IF YES
12111	061204	005726			TST	(SP)+	::CLEAR CONTROL-G OFF STACK
12112	061206	004737	060652		JSR	PC, STKINT	::FLUSH THE TTY INPUT QUEUE
12113	061212	005077	117742		CLR	2STKS	::DISABLE TTY KEYBOARD INTERRUPTS
12114	061216	112737	000001	001151	MOVB	#1, \$INTAG	::SET INTERRUPT MODE INDICATOR
12115							
12116	061224	104401	062050		SGTSWR: TYPE	, \$CNTLG	::ECHO THE CONTROL-G (#G)
12117	061230	104401	062055		TYPE	, \$MSWR	::TYPE CURRENT CONTENTS
12118	061234	013746	000176		MOV	\$WREG, -(SP)	::SAVE SWREG FOR TYPEOUT
12119	061240	104402			TYPOC		::GO TYPE--OCTAL ASCII (ALL DIGITS)

12120	061242	104401	062066			TYPE	,SMNEW	:: PROMPT FOR NEW SWR
12121	061246	005046		19\$:		CLR	-(SP)	:: CLEAR COUNTER
12122	061250	005046				CLR	-(SP)	:: THE NEW SWR
12123	061252	105777	117702	7\$:		TSTB	2\$TKS	:: CHAR THERE?
12124	061256	100375				BPL	7\$:: IF NOT TRY AGAIN
12125								
12126	061260	117746	117676			MOVB	2\$TKB, -(SP)	:: PICK UP CHAR
12127	061264	042716	177600			BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
12128								
12129	061270	021627	000003			CMP	(SP), #3	:: IS IT A CONTROL-C?
12130	061274	001015				BNE	9\$:: BRANCH IF NOT
12131	061276	104401	062036			TYPE	, \$CNTLC	:: YES, ECHO CONTROL-C (↑C)
12132	061302	062706	000006			ADD	#6, SP	:: CLEAN UP STACK
12133	061306	123727	001151	000001		CMPB	\$INTAG, #1	:: REENABLE TTY KEYBOARD INTERRUPTS?
12134	061314	001003				BNE	8\$:: BRANCH IF NO
12135	061316	012777	000100	117634		MOV	#100, 2\$TKS	:: ALLOW TTY KEYBOARD INTERRUPTS
12136	061324	000137	056660	8\$:		JMP	SHUT	:: CONTROL-C RESTART
12137								
12138								
12139	061330	021627	000025	9\$:		CMP	(SP), #25	:: IS IT A CONTROL-U?
12140	061334	001005				BNE	10\$:: BRANCH IF NOT
12141	061336	104401	062043			TYPE	, \$CNTLU	:: YES, ECHO CONTROL-U (↑U)
12142	061342	062706	000006	20\$:		ADD	#6, SP	:: IGNORE PREVIOUS INPUT
12143	061346	000737				BR	19\$:: LET'S TRY IT AGAIN
12144								
12145								
12146	061350	021627	000015	10\$:		CMP	(SP), #15	:: IS IT A <CR>?
12147	061354	001022				BNE	16\$:: BRANCH IF NO
12148	061356	005766	000004			TST	4(SP)	:: YES, IS IT THE FIRST CHAR?
12149	061362	001403				BEQ	11\$:: BRANCH IF YES
12150	061364	016677	000002	117562		MOV	2(SP), 2\$SWR	:: SAVE NEW SWR
12151	061372	062706	000006		11\$:	ADD	#6, SP	:: CLEAN UP STACK
12152	061376	104401	001217		14\$:	TYPE	, \$CRLF	:: ECHO <CR> AND <LF>
12153	061402	123727	001151	000001		CMPB	\$INTAG, #1	:: RE-ENABLE TTY KBD INTERRUPTS?
12154	061410	001003				BNE	15\$:: BRANCH IF NOT
12155	061412	012777	000100	117540		MOV	#100, 2\$TKS	:: RE-ENABLE TTY KBD INTERRUPTS
12156	061420	000002		15\$:		RTI		:: RETURN
12157	061422	004737	057762	16\$:		JSR	PC, \$TYPEC	:: ECHO CHAR
12158	061426	021627	000060			CMP	(SP), #60	:: CHAR < 0?
12159	061432	002420				BLT	18\$:: BRANCH IF YES
12160	061434	021627	000067			CMP	(SP), #67	:: CHAR > 7?
12161	061440	003015				BGT	18\$:: BRANCH IF YES
12162	061442	042726	000060			BIC	#60, (SP)+	:: STRIP-OFF ASCII
12163	061446	005766	000002			TST	2(SP)	:: IS THIS THE FIRST CHAR
12164	061452	001403				BEQ	17\$:: BRANCH IF YES
12165	061454	006316				ASL	(SP)	:: NO, SHIFT PRESENT
12166	061456	006316				ASL	(SP)	:: CHAR OVER TO MAKE
12167	061460	006316				ASL	(SP)	:: ROOM FOR NEW ONE.
12168	061462	005266	000002	17\$:		INC	2(SP)	:: KEEP COUNT OF CHAR
12169	061466	056616	177776			BIS	-2(SP), (SP)	:: SET IN NEW CHAR
12170	061472	000667				BR	7\$:: GET THE NEXT ONE
12171	061474	104401	001216	18\$:		TYPE	, \$QUES	:: TYPE ?<CR><LF>
12172	061500	000720				BR	20\$:: SIMULATE CONTROL-U
12173						.DSABL	LSB	
12174								
12175								

```

12176 .....*****
12177 : THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
12178 : *CALL:
12179 : *      RDCHR          ;; GET A CHARACTER FROM THE QUEUE
12180 : *      RETURN HERE    ;; CHARACTER IS ON THE STACK
12181 : *                    ;; WITH PARITY BIT STRIPPED OFF
12182 :
12183 :
12184 061502 011646 SRDCHR: MOV      (SP), -(SP)    ;; PUSH DOWN THE PC AND
12185 061504 016666 000004 000002 MOV      4(SP), 2(SP)    ;; THE PS
12186 061512 005066 000004          CLR      4(SP)        ;; GET READY FOR A CHARACTER
12187 061516 005046          CLR      -(SP)         ;; PUT NEW PS ON STACK
12188 061520 012746 061526          MOV      #64$, -(SP)    ;; PUT NEW PC ON STACK
12189 061524 000002          RTI                    ;; POP NEW PC AND PS
12190 061526
12191 061526 005737 060642 64$:  TST      STKCNT    ;; WAIT ON A CHARACTER
12192 061532 001775          BEQ      1$
12193 061534 005337 060642          DEC      STKCNT    ;; DECREMENT THE COUNTER
12194 061540 117766 177102 000004 MOVB    2$STKQOUT, 4(SP) ;; GET ONE CHARACTER
12195 061546 005237 060646          INC      STKQOUT    ;; UPDATE THE POINTER
12196 061552 023727 060646 060651 CMP     STKQOUT, #STKQEND ;; DID IT GO OFF OF THE END?
12197 061560 001003          BNE     2$        ;; BRANCH IF NO
12198 061562 012737 060650 060646 MOV     #STKQSR, STKQOUT ;; RESET THE POINTER
12199 061570 000002          RTI                    ;; RETURN
12200 .....*****
12201 : THIS ROUTINE WILL INPUT A STRING FROM THE TTY
12202 : *CALL:
12203 : *      RDLIN         ;; INPUT A STRING FROM THE TTY
12204 : *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
12205 : *                    ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
12206 :
12207 061572 010346 SRDLIN: MOV      R3, -(SP)    ;; SAVE R3
12208 061574 005046          CLR      -(SP)        ;; CLEAR THE RUBOUT KEY
12209 061576 012703 062026 1$:  MOV     #STTYIN, R3    ;; GET ADDRESS
12210 061602 022703 062036 2$:  CMP     #STTYIN+8., R3 ;; BUFFER FULL?
12211 061606 101456          BLOS   4$            ;; BR IF YES
12212 061610 104411          RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
12213 061612 112613          MOVB   (SP)+, (R3)    ;; GET CHARACTER
12214 061614 122713 000177 10$: CMPB   #177, (R3)    ;; IS IT A RUBOUT
12215 061620 001022          BNE   5$            ;; BR IF NO
12216 061622 005716          TST   (SP)         ;; IS THIS THE FIRST RUBOUT?
12217 061624 001007          BNE   6$            ;; BR IF NO
12218 061626 112737 000134 062024 MOVB   #' \, 9$      ;; TYPE A BACK SLASH
12219 061634 104401 062024          TYPE   9$
12220 061640 012716 177777          MOV   #-1, (SP)    ;; SET THE RUBOUT KEY
12221 061644 005303 6$:  DEC     R3          ;; BACKUP BY ONE
12222 061646 020327 062026          CMP   R3, #STTYIN ;; STACK EMPTY?
12223 061652 103434          BLOS  4$            ;; BR IF YES
12224 061654 111337 062024          MOVB  (R3), 9$     ;; SETUP TO TYPEOUT THE DELETED CHAR.
12225 061660 104401 062024          TYPE  9$
12226 061664 000746          BR    2$           ;; GO TYPE
12227 061666 005716 5$:  TST   (SP)         ;; GO READ ANOTHER CHAR.
12228 061670 001406          BEQ   7$           ;; RUBOUT KEY SET?
12229 061672 112737 000134 062024 MOVB   #' \, 9$      ;; BR IF NO
12230 061700 104401 062024          TYPE  9$
12231 061704 005016          CLR   (SP)        ;; CLEAR THE RUBOUT KEY

```

```

12232 061706 122713 000025      7S:  CMPB    #25,(R3)      ;; IS CHARACTER A CTRL U?
12233 061712 001003              BNE     8S              ;; BR IF NO
12234 061714 104401 062043      TYPE   SCNTLU          ;; TYPE A CONTROL "U"
12235 061720 000726              BR      1S              ;; GO START OVER
12236 061722 122713 000022      8S:  CMPB    #22,(R3)      ;; IS CHARACTER A "r"?
12237 061726 001011              BNE     3S              ;; BRANCH IF NO
12238 061730 105013              CLRB   (R3)            ;; CLEAR THE CHARACTER
12239 061732 104401 001217      TYPE   ,SCRLF          ;; TYPE A "CR" & "LF"
12240 061736 104401 062026      TYPE   ,STTYIN         ;; TYPE THE INPUT STRING
12241 061742 000717              BR      2S              ;; GO PICKUP ANOTHER CHACTER
12242 061744 104401 001216      4S:  TYPE   ,SQUES          ;; TYPE A '?'
12243 061750 000712              BR      1S              ;; CLEAR THE BUFFER AND LOOP
12244 061752 111337 062024      3S:  MOVB   (R3),9S        ;; ECHO THE CHARACTER
12245 061756 104401 062024      TYPE   ,9S
12246 061762 122723 000015      CMPB   #15,(R3)+      ;; CHECK FOR RETURN
12247 061766 001305              BNE     2S              ;; LOOP IF NOT RETURN
12248 061770 105063 177777      CLRB   -1(R3)          ;; CLEAR RETURN (THE 15)
12249 061774 104401 001220      TYPE   ,SLF            ;; TYPE A LINE FEED
12250 062000 005726              TST    (SP)+           ;; CLEAN RUBOUT KEY FROM THE STACK
12251 062002 012603              MOV    (SP)+,R3        ;; RESTORE R3
12252 062004 011646              MOV    (SP),-(SP)      ;; ADJUST THE STACK AND PUT ADDRESS OF THE
12253 062006 016666 000004 000002  MOV    4(SP),2(SP)     ;; FIRST ASCII CHARACTER ON IT
12254 062014 012766 062026 000004  MOV    #STTYIN,4(SP)
12255 062022 000002              RTI
12256 062024 000              9S:  .BYTE   0            ;; RETURN
12257 062025 000              .BYTE   0            ;; STORAGE FOR ASCII CHAR. TO TYPE
12258 062026 000010              STTYIN: .BLKB  8        ;; TERMINATOR
12259 062036 041536 005015 000      SCNTLC: .ASCIZ /tC/<15><12> ;; RESERVE 8 BYTES FOR TTY INPUT
12260 062043 0136 006525 000012  SCNTLU: .ASCIZ /tU/<15><12> ;; CONTROL "C"
12261 062050 043536 005015 000      SCNTLG: .ASCIZ /tG/<15><12> ;; CONTROL "U"
12262 062055 015 051412 051127  SMSMR: .ASCIZ <15><12>/SMR = / ;; CONTROL "G"
12263 062062 036440 000040
12264 062066 020040 042516 020127  SMNEW: .ASCIZ / NEW = /
12265 062074 020075 000
12266 062100
12267 .EVEN
12268 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
12269
12270 ;; *****
12271 ;; *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
12272 ;; *CHANGE IT TO BINARY.
12273 ;; *CALL:
12274 ;; *
12275 ;; *   RDOCT
12276 ;; *   RETURN HERE
12277 ;; *
12278 ;; * READ AN OCTAL NUMBER
12279 ;; * LOW ORDER BITS ARE ON TOP OF THE STACK
12280 ;; * HIGH ORDER BITS ARE IN SHIOCT
12281
12282 SRDOCT: MOV    (SP),-(SP)      ;; PROVIDE SPACE FOR THE
12283         MOV    4(SP),2(SP)    ;; INPUT NUMBER
12284         MOV    R0,-(SP)       ;; PUSH R0 ON STACK
12285         MOV    R1,-(SP)       ;; PUSH R1 ON STACK
12286         MOV    R2,-(SP)       ;; PUSH R2 ON STACK
12287         RDLIN          ;; READ AN ASCII LINE
12288         MOV    (SP)+,R0       ;; GET ADDRESS OF 1ST CHARACTER
12289         CLR    R1              ;; CLEAR DATA WORD
12290         CLR    R2
12291
12292 1S:  MOVB   (R0)+,-(SP)        ;; PICKUP THIS CHARACTER
12293         BEQ    3S              ;; IF ZERO GET OUT
12294         3S

```

```

12288 062132 006301
12289 062134 006102
12290 062136 006301
12291 062140 006102
12292 062142 006301
12293 062144 006102
12294 062146 042716 177770
12295 062152 062601
12296 062154 000764
12297 062156 005726 3S:
12298 062160 010166 000012
12299 062164 010237 062200
12300 062170 012602
12301 062172 012601
12302 062174 012600
12303 062176 000002
12304 062200 000000
12305
12306
12307
12308
12309
12310
12311
12312
12313 062202 016646 000002
12314 062206 042716 000020
12315 062212 012746 062220
12316 062216 000002
12317 062220 010046 1S:
12318 062222 016600 000002
12319 062226 005740
12320 062230 111000
12321 062232 006300
12322 062234 016000 062254
12323 062240 000200
12324
12325
12326
12327
12328 062242 011646
12329 062244 016666 000004 000002
12330 062252 000002
12331
12332
12333
12334
12335
12336
12337
12338
12339 062254 062242
12340 062256 057550
12341 062260 057346
12342 062262 057322
12343 062264 057362

```

```

ASL R1 ;;#2
ROL R2
ASL R1 ;;#4
ROL R2
ASL R1 ;;#8
ROL R2
BIC #1C7,(SP) ;;STRIP THE ASCII JUNK
ADD (SP)+,R1 ;;ADD IN THIS DIGIT
BR 2$ ;;LOOP
3S: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
MOV R1,12(SP) ;;SAVE THE RESULT
MOV R2,$HIOCT
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI ;;RETURN
$HIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE
.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.

```

```

$TRAP: MOV 2(SP),-(SP) ;;ASSUME THE STATUS OF
BIC #20,(SP) ;; THE CALLER--DO NOT ALLOW
MOV #1$,-(SP) ;; T-BIT TRAPS
RTI ;;SET THE NEW STATUS
1S: MOV R0, -(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE

```

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

```

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

```

```
.SBTTL TRAP TABLE
```

```

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

```

ROUTINE	STARTING ADDRESS	ROUTINE NAME
\$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)

12344	062266	057076			\$TYPDS	::CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
12345	062270	057022			\$TYPBN	::CALL=TYPBN	TRAP+6(104406)	TYPE BINARY (ASCII) NUMBER
12346								
12347	062272	061230			\$GTSWR	::CALL=GTSWR	TRAP+7(104407)	GET SOFT-SWR SETTING
12348								
12349	062274	061140			\$CKSWR	::CALL=CKSWR	TRAP+10(104410)	TEST FOR CHANGE IN SOFT-SWR
12350	062276	061502			\$RDCHR	::CALL=RDCHR	TRAP+11(104411)	TTY TYPEIN CHARACTER ROUTINE
12351	062300	061572			\$RDLIN	::CALL=RDLIN	TRAP+12(104412)	TTY TYPEIN STRING ROUTINE
12352	062302	062100			\$RDOCT	::CALL=RDOCT	TRAP+13(104413)	READ AN OCTAL NUMBER FROM TTY
12353	062304	056726			\$SAVREG	::CALL=SAVREG	TRAP+14(104414)	SAVE R0-R5 ROUTINE
12354	062306	056764			\$RESREG	::CALL=RESREG	TRAP+15(104415)	RESTORE R0-R5 ROUTINE

.SBTTL POWER DOWN AND UP ROUTINES

::*****

:POWER DOWN ROUTINE

12358								
12359	062310	012737	062450	000024	\$PWRDN:	MOV	\$SILLUP, @PWRVEC	::SET FOR FAST UP
12360	062316	012737	000340	000026		MOV	@340, @PWRVEC+2	::PRIO:7
12361	062324	010046				MOV	R0, -(SP)	::PUSH R0 ON STACK
12362	062326	010146				MOV	R1, -(SP)	::PUSH R1 ON STACK
12363	062330	010246				MOV	R2, -(SP)	::PUSH R2 ON STACK
12364	062332	010346				MOV	R3, -(SP)	::PUSH R3 ON STACK
12365	062334	010446				MOV	R4, -(SP)	::PUSH R4 ON STACK
12366	062336	010546				MOV	R5, -(SP)	::PUSH R5 ON STACK
12367	062340	017746	116610			MOV	@SWR, -(SP)	::PUSH @SWR ON STACK
12368	062344	010637	062454			MOV	SP, \$SAVR6	::SAVE SP
12369	062350	012737	062362	000024		MOV	\$PWRUP, @PWRVEC	::SET UP VECTOR
12370	062356	000000				HALT		
12371	062360	000776				BR	.-2	::HANG UP

::*****

:POWER UP ROUTINE

12372								
12373								
12374								
12375	062362	012737	062450	000024	\$PWRUP:	MOV	\$SILLUP, @PWRVEC	::SET FOR FAST DOWN
12376	062370	013706	062454			MOV	\$SAVR6, SP	::GET SP
12377	062374	005037	062454			CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
12378	062400	005237	062454		1\$:	INC	\$SAVR6	::WAIT FOR THE INC
12379	062404	001375				BNE	1\$::OF WORD
12380	062406	012677	116542			MOV	(SP)+, @SWR	::POP STACK INTO @SWR
12381	062412	012605				MOV	(SP)+, R5	::POP STACK INTO R5
12382	062414	012604				MOV	(SP)+, R4	::POP STACK INTO R4
12383	062416	012603				MOV	(SP)+, R3	::POP STACK INTO R3
12384	062420	012602				MOV	(SP)+, R2	::POP STACK INTO R2
12385	062422	012601				MOV	(SP)+, R1	::POP STACK INTO R1
12386	062424	012600				MOV	(SP)+, R0	::POP STACK INTO R0
12387	062426	012737	062310	000024		MOV	\$PWRDN, @PWRVEC	::SET UP THE POWER DOWN VECTOR
12388	062434	012737	000340	000026		MOV	@340, @PWRVEC+2	::PRIO:7
12389	062442	104401				TYPE		::REPORT THE POWER FAILURE
12390	062444	062456			\$PWRMG:	.WORD	\$POWER	::POWER FAIL MESSAGE POINTER
12391	062446	000002				RTI		
12392	062450	000000			\$SILLUP:	HALT		::THE POWER UP SEQUENCE WAS STARTED
12393	062452	000776				BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
12394	062454	000000			\$SAVR6:	0		::PUT THE SP HERE
12395	062456	005015	047520	042527	\$POWER:	.ASCIZ	<15><12>"POWER"	
12396	062464	000122						

.EVEN
.SBTTL APT COMMUNICATIONS ROUTINE

12397
12398
12399

```

12400 .....
12401 062466 112737 000001 062732 $ATY1: MOV  #1,$FFLG ;; TO REPORT FATAL ERROR
12402 062474 112737 000001 062730 $ATY3: MOV  #1,$MFLG ;; TO TYPE A MESSAGE
12403 062502 000403          $ATYC: BR   $ATYC
12404 062504 112737 000001 062732 $ATY4: MOV  #1,$FFLG ;; TO ONLY REPORT FATAL ERROR
12405 062512 010046          $ATYC: MOV  R0,-(SP) ;; PUSH R0 ON STACK
12406 062512 010146          MOV  R1,-(SP) ;; PUSH R1 ON STACK
12407 062514 010146          TSTB $MFLG ;; SHOULD TYPE A MESSAGE?
12408 062516 105737 062730          BEQ  5$ ;; IF NOT: BR
12409 062522 001450          CMPB $APTENV,$ENV ;; OPERATING UNDER APT?
12410 062524 122737 000001 001242          BNE  3$ ;; IF NOT: BR
12411 062532 001031          BITB $APTSPool,$ENVM ;; SHOULD SPOOL MESSAGES?
12412 062534 132737 000100 001243          BEQ  3$ ;; IF NOT: BR
12413 062542 001425          MOV  #4(SP),R0 ;; GET MESSAGE ADDR.
12414 062544 017600 000004          ADD  #2,4(SP) ;; BUMP RETURN ADDR.
12415 062550 062766 000002 000004          TST $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
12416 062556 005737 001222          BNE  1$ ;; IF NOT: WAIT
12417 062562 001375          MOV  R0,$MSGAD ;; PUT ADDR IN MAILBOX
12418 062564 010037 001236          TSTB (R0)+ ;; FIND END OF MESSAGE
12419 062570 105720          BNE  2$
12420 062572 001376          SUB  $MSGAD,R0 ;; SUB START OF MESSAGE
12421 062574 163700 001236          ASR  R0 ;; GET MESSAGE LNTH IN WORDS
12422 062600 006200          MOV  R0,$MSGLEN ;; PUT LENGTH IN MAILBOX
12423 062602 010037 001240          MOV  #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
12424 062606 012737 000004 001222          BR   5$
12425 062614 000413          MOV  #4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
12426 062616 017637 000004 062642 3$: ADD  #2,4(SP) ;; BUMP RETURN ADDRESS
12427 062624 062766 000002 000004          MOV  177776,-(SP) ;; PUSH 177776 ON STACK
12428 062632 013746 177776          JSR  PC,$TYPE ;; CALL TYPE MACRO
12429 062636 004737 057550          .WORD 0
12430 062642 000000          4$:
12431 062644          5$:
12432 062644 105737 062732          10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
12433 062650 001416          BEQ  12$ ;; IF NOT: BR
12434 062652 005737 001242          TST  $ENV ;; RUNNING UNDER APT?
12435 062656 001413          BEQ  12$ ;; IF NOT: BR
12436 062660 005737 001222          11$: TST  $MSGTYPE ;; FINISHED LAST MESSAGE?
12437 062664 001375          BNE  11$ ;; IF NOT: WAIT
12438 062666 017637 000004 001224          MOV  #4(SP),$FATAL ;; GET ERROR #
12439 062674 062766 000002 000004          ADD  #2,4(SP) ;; BUMP RETURN ADDR.
12440 062702 005237 001222          INC  $MSGTYPE ;; TELL APT TO TAKE ERROR
12441 062706 105037 062732          12$: CLRB $FFLG ;; CLEAR FATAL FLAG
12442 062712 105037 062731          CLRB $LFLG ;; CLEAR LOG FLAG
12443 062716 105037 062730          CLRB $MFLG ;; CLEAR MESSAGE FLAG
12444 062722 012601          MOV  (SP)+,R1 ;; POP STACK INTO R1
12445 062724 012600          MOV  (SP)+,R0 ;; POP STACK INTO R0
12446 062726 000207          RTS  PC ;; RETURN
12447 062730 000          $MFLG: .BYTE 0 ;; MESSG. FLAG
12448 062731 000          $LFLG: .BYTE 0 ;; LOG FLAG
12449 062732 000          $FFLG: .BYTE 0 ;; FATAL FLAG
12450          062734          .EVEN
12451          000200          APTSIZE=200
12452          000001          APTENV=001
12453          000100          APTSPool=100
12454          000040          APTCSUP=040
12455

```

I04

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 254
APT COMMUNICATIONS ROUTINE

SEQ 0256

12456

.NLIST BEX

.SBTTL CONSOLE MESSAGES

062734				SCTMSG:	
062734	005015	040503	047116	.ASCII	<CR><LF>CANNOT RECOVER THE BAD SECTOR FILES FROM LAST <CR><LF>
063016	051124	041501	020113	.ASCIZ	TRACK FOR THIS DEVICE
063044	051			CLSPRN:	.ASCII <CR><LF>
063045	075	000		EQUALS:	.ASCIZ <CR><LF>
063047	015	025012	000	PROMPT:	.ASCIZ <CR><LF>*
063053	077	000		QSTMRK:	.ASCIZ <CR><LF>
063055				HELPOST:	
063055	015	052012	050131	.ASCIZ	<CR><LF>TYPE HELP TEXT (Y OR N)??
063111				UBUSQST:	
063111	015	041412	040510	.ASCIZ	<CR><LF>CHANGE RMD3 UNIBUS ADDRESS OR VECTOR ADDRESS (Y OR N) ??
063204	005015	051525	020105	CNSL00:	.ASCIZ <CR><LF>USE SAME DEVICES (Y OR N) ??
063243	015	051012	030115	CNSL01:	.ASCIZ <CR><LF>RMD3 BUS ADDRESS (<CR><LF>RMD3 BUS ADDRESS (<CR><LF>ENTRY NOT IN I/O PAGE<CR><LF>ADDRESS MUST BE >160000<CR><LF>RMD3 VECTOR ADDRESS (<CR><LF>ENTRY OUT OF RANGE<CR><LF>ADDRESS MUST BE <10000<CR><LF>RMD3 INTERRUPT PRIORITY (<CR><LF>ENTRY OUT OF RANGE<CR><LF>TYPE (A) TO TEST ALL DEVICES, OR TYPE DEVICE<CR><LF>NUMBER(S)<CR><LF>TERMINATE INPUT WITH CARRIAGE RETURN
063270	005015	047105	051124	CNSL02:	.ASCII <CR><LF>ENTRY NOT IN I/O PAGE
063317	015	040412	042104	.ASCIZ	<CR><LF>ADDRESS MUST BE >160000
063351	015	051012	030115	CNSL03:	.ASCIZ <CR><LF>RMD3 VECTOR ADDRESS (<CR><LF>ENTRY OUT OF RANGE
063401	015	042412	052116	CNSL04:	.ASCII <CR><LF>ENTRY OUT OF RANGE
063425	015	040412	042104	.ASCIZ	<CR><LF>ADDRESS MUST BE <10000
063455	015	051012	030115	CNSL05:	.ASCIZ <CR><LF>RMD3 INTERRUPT PRIORITY (<CR><LF>ENTRY OUT OF RANGE
063511	015	042412	052116	CNSL06:	.ASCIZ <CR><LF>ENTRY OUT OF RANGE
063536				CNSL07:	
063536	005015	054524	042520	.ASCII	<CR><LF>TYPE (A) TO TEST ALL DEVICES, OR TYPE DEVICE
063614	047040	046525	042502	.ASCII	<CR><LF>NUMBER(S)
063626	005015	042524	046522	.ASCIZ	<CR><LF>TERMINATE INPUT WITH CARRIAGE RETURN
063676				.EVEN	

.SBTTL FUNCTION CODE TABLE

; THE FUNCTION CODE TABLE IS USED TO DEFINE STATUS CONDITIONS FOR
; EACH FUNCTION CODE. BIT USAGE IS AS FOLLOWS:

; ATA - BIT 15 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF ATA SHOULD BE SET WHEN THE FUNCTION CODE IS EXECUTED, OTHERWISE,
; BIT 15 IS ZERO, INDICATING THAT ATA SHOULD NOT NORMALLY BE SET.
; NOTE THAT ATA MAY BE SET WHEN A COMMAND IS EXECUTED EVEN THOUGH
; IT IS NOT EXPECTED AS A RESULT OF THE COMMAND.

; WCE - BIT 14 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF WRITE CHECK ERRORS ARE ENABLED AS A FUNCTION OF THE COMMAND.

; OPI - BIT 13 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF OPI ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.

; IVC - BIT 12 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF IVC ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.

; WLE - BIT 11 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF WRITE ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.
; THE WRITE ERRORS WHICH ARE ENABLED ARE "WLE", "WCF", "DPE", "UPE".

; IAE - BIT 10 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF INVALID ADDRESS ERROR IS ENABLED FOR THAT COMMAND.

; AOE - BIT 09 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF READ AND WRITE ERRORS ARE ENABLED DURING THE EXECUTION OF THE
; COMMAND. THE ERRORS ENABLED BY THIS BIT ARE "TRE", "DLT", "NEM",
; "MXF", "LBT", AND "AOE".

; BIT 08 IS NOT USED.

; HCE - BIT 07 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF HEADER ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.
; HEADER ERRORS INCLUDE "HCRC", "HCE", "FER", AND "BSE".

; ECH - BIT 06 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF DATA FIELD ERRORS ARE ENABLED DURING THE EXECUTION OF THAT
; COMMAND. THESE ERRORS INCLUDE "MDPE", "DCK", AND "ECH".

; BIT 05 IS NOT USED.

; BIT 04 IS NOT USED.

; BIT 03 IS NOT USED.

; BIT 02 IS NOT USED.

; BIT 01 IS NOT USED.

; ILF - BIT 00 IS SET IF THE FUNCTION CODE IS ILLEGAL.

063676	020000	.WORD	OPI	:NOP
063700	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (2)
063702	132000	.WORD	ATA:OPI:IVC:IAE	:SEEK
063704	130000	.WORD	ATA:OPI:IVC	:RECALIBRATE
063706	020000	.WORD	OPI	:DRIVE CLEAR
063710	030000	.WORD	OPI:IVC	:RELEASE
063712	130000	.WORD	OPI:ATA:IVC	:OFFSET
063714	130000	.WORD	OPI:ATA:IVC	:RETURN TO CENTERLINE
063716	020000	.WORD	OPI	:READ IN PRESET
063720	020000	.WORD	OPI	:PACK ACKNOWLEDGE
063722	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (24)
063724	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (26)
063726	132000	.WORD	ATA:OPI:IVC:IAE	:SEARCH
063730	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (32)
063732	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (34)
063734	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (36)
063736	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (40)
063740	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (42)
063742	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (44)
063744	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (46)
063746	073300	.WORD	WCE:OPI:IVC:IAE:AOE:HCE:ECH	:WRITE CHECK DATA
063750	073300	.WORD	WCE:OPI:IVC:IAE:AOE:HCE:ECH	:WRITE CHECK HEADER AND DATA
063752	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (54)
063754	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (56)
063756	037200	.WORD	OPI:IVC:WLE:IAE:AOE:HCE	:WRITE DATA
063760	037000	.WORD	OPI:IVC:WLE:IAE:AOE	:WRITE HEADER AND DATA
063762	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (64)
063764	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (66)
063766	033300	.WORD	OPI:IVC:IAE:AOE:HCE:ECH	:READ DATA
063770	033300	.WORD	OPI:IVC:IAE:AOE:HCE:ECH	:READ HEADER AND DATA
063772	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (74)
063774	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (76)

M04

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 258
ATTENTION (ATA) TABLE

SEQ 0260

.SBTTL ATTENTION (ATA) TABLE

063776	001	ATNTBL: .BYTE	1.
063777	002	.BYTE	2.
064000	004	.BYTE	4.
064001	010	.BYTE	8.
064002	020	.BYTE	16.
064003	040	.BYTE	32.
064004	100	.BYTE	64.
064005	200	.BYTE	128.

.SBTTL DATA PATTERN TABLE

RGDTPT:
 MIXED:

064006
 064006
 064006 000000
 064010 000001
 064012 000003
 064014 000007
 064016 000017
 064020 000037
 064022 000077
 064024 000177
 064026 000377
 064030 000777
 064032 001777
 064034 003777
 064036 007777
 064040 017777
 064042 037777
 064044 077777
 064046 177777
 064050 177777
 064052 077777
 064054 037777
 064056 017777
 064060 007777
 064062 003777
 064064 001777
 064066 000777
 064070 000377
 064072 000177
 064074 000077
 064076 000037
 064100 000017
 064102 000007
 064104 000003
 064106 000001
 064110 000000
 064112 000000
 064114 000001
 064116 000002
 064120 000004
 064122 000010
 064124 000020
 064126 000040
 064130 000100
 064132 000200
 064134 000400
 064136 001000
 064140 002000
 064142 004000
 064144 010000
 064146 020000
 064150 040000
 064152 100000
 064154 100000

ONES:

ZEROS:

.WORD 0.
 .WORD 1.
 .WORD 3.
 .WORD 7.
 .WORD 15.
 .WORD 31.
 .WORD 63.
 .WORD 127.
 .WORD 255.
 .WORD 511.
 .WORD 1023.
 .WORD 2047.
 .WORD 4095.
 .WORD 8191.
 .WORD 16383.
 .WORD 32767.
 .WORD 65535.
 .WORD 65535.
 .WORD 32767.
 .WORD 16383.
 .WORD 8191.
 .WORD 4095.
 .WORD 2047.
 .WORD 1023.
 .WORD 511.
 .WORD 255.
 .WORD 127.
 .WORD 63.
 .WORD 31.
 .WORD 15.
 .WORD 7.
 .WORD 3.
 .WORD 1.
 .WORD 0.
 .WORD 0.
 .WORD 1.
 .WORD 2.
 .WORD 4.
 .WORD 8.
 .WORD 16.
 .WORD 32.
 .WORD 64.
 .WORD 128.
 .WORD 256.
 .WORD 512.
 .WORD 1024.
 .WORD 2048.
 .WORD 4096.
 .WORD 8192.
 .WORD 16384.
 .WORD 32768.
 .WORD 32768.

064156	040000	.WORD	16384.
064160	020000	.WORD	8192.
064162	010000	.WORD	4096.
064164	004000	.WORD	2048.
064166	002000	.WORD	1024.
064170	001000	.WORD	512.
064172	000400	.WORD	256.
064174	000200	.WORD	128.
064176	000100	.WORD	64.
064200	000040	.WORD	32.
064202	000020	.WORD	16.
064204	000010	.WORD	8.
064206	000004	.WORD	4.
064210	000002	.WORD	2.
064212	000001	.WORD	1.
064214	000000	.WORD	0.
064216	177777	.WORD	65535.
064220	177776	.WORD	65534.
064222	177774	.WORD	65532.
064224	177770	.WORD	65528.
064226	177760	.WORD	65520.
064230	177740	.WORD	65504.
064232	177700	.WORD	65472.
064234	177600	.WORD	65408.
064236	177400	.WORD	65280.
064240	177000	.WORD	65024.
064242	176000	.WORD	64512.
064244	174000	.WORD	63488.
064246	170000	.WORD	61440.
064250	160000	.WORD	57344.
064252	140000	.WORD	49152.
064254	100000	.WORD	32768.
064256	000000	.WORD	0.
064260	000000	.WORD	0.
064262	100000	.WORD	32768.
064264	140000	.WORD	49152.
064266	160000	.WORD	57344.
064270	170000	.WORD	61440.
064272	174000	.WORD	63488.
064274	176000	.WORD	64512.
064276	177000	.WORD	65024.
064300	177400	.WORD	65280.
064302	177600	.WORD	65408.
064304	177700	.WORD	65472.
064306	177740	.WORD	65504.
064310	177760	.WORD	65520.
064312	177770	.WORD	65528.
064314	177774	.WORD	65532.
064316	177776	.WORD	65534.
064320	177777	.WORD	65535.
064322	125252	.WORD	43690.
064324	152525	.WORD	43690./2
064326	125252	.WORD	43690.
064330	177777	.WORD	65535.
064332	177776	.WORD	65534.
064334	177775	.WORD	65533.

EARLY:

064336	177773	.WORD	65531.
064340	177767	.WORD	65527.
064342	177757	.WORD	65519.
064344	177737	.WORD	65503.
064346	177677	.WORD	65471.
064350	177577	.WORD	65407.
064352	177377	.WORD	65279.
064354	176777	.WORD	65023.
064356	175777	.WORD	64511.
064360	173777	.WORD	63487.
064362	167777	.WORD	61439.
064364	157777	.WORD	57343.
064366	137777	.WORD	49151.
064370	077777	.WORD	32767.
064372	077777	.WORD	32767.
064374	137777	.WORD	49151.
064376	157777	.WORD	57343.
064400	167777	.WORD	61439.
064402	173777	.WORD	63487.
064404	175777	.WORD	64511.
064406	176777	.WORD	65023.
064410	177377	.WORD	65279.
064412	177577	.WORD	65407.
064414	177677	.WORD	65471.
064416	177737	.WORD	65503.
064420	177757	.WORD	65519.
064422	177767	.WORD	65527.
064424	177773	.WORD	65531.
064426	177775	.WORD	65533.
064430	177776	.WORD	65534.
064432	177777	.WORD	65535.
064434			

ENRGDT:

.SBTTL ERROR MESSAGE TABLE

064434	071030	000000		EMT1:	.WORD	EMS1,0
064440	071077	071114	000000	EMT2:	.WORD	EMS2,EMS3,0
064446	071077	071157	000000	EMT3:	.WORD	EMS2,EMS4,0
064454	071222	071252	000000	EMT4:	.WORD	EMS5,EMS6,0
064462	071222	071364	000000	EMT5:	.WORD	EMS5,EMS10,0
064470	076057	073245	000000	EMT6:	.WORD	EMS167,EMS64,0
064476	074013	076104	000000	EMT7:	.WORD	EMS110,EMS170,0
064504	071317	000000		EMT10:	.WORD	EMS7,0
064510	071364	000000		EMT11:	.WORD	EMS10,0
064514	071426	071437	000000	EMT12:	.WORD	EMS11,EMS12,0
064522	071500	071511	071522	EMT13:	.WORD	EMS13,EMS14,EMS15,EMS16,0
064534	071574	073245	000000	EMT14:	.WORD	EMS17,EMS64,0
064542	071426	071657	000000	EMT15:	.WORD	EMS11,EMS21,0
064550	071426	071702	072031	EMT16:	.WORD	EMS11,EMS22,EMS27,0
064560	071426	071716	072042	EMT17:	.WORD	EMS11,EMS23,EMS30,0
064570	071426	071744	072042	EMT20:	.WORD	EMS11,EMS24,EMS30,0
064600	071426	071773	072031	EMT21:	.WORD	EMS11,EMS25,EMS27,0
064610	071426	072010	072031	EMT22:	.WORD	EMS11,EMS26,EMS27,0
064620	071426	072052	072042	EMT23:	.WORD	EMS11,EMS31,EMS30,0
064630	071426	072101	072042	EMT24:	.WORD	EMS11,EMS32,EMS30,0
064640	071426	072130	072042	EMT25:	.WORD	EMS11,EMS33,EMS30,0
064650	071426	072156	072042	EMT26:	.WORD	EMS11,EMS34,EMS30,0
064660	071426	072227	072042	EMT27:	.WORD	EMS11,EMS35,EMS30,0
064670	071426	072256	072042	EMT30:	.WORD	EMS11,EMS36,EMS30,0
064700	071426	072305	072042	EMT31:	.WORD	EMS11,EMS37,EMS30,0
064710	071426	072333	072042	EMT32:	.WORD	EMS11,EMS40,EMS30,0
064720	071426	072362	072042	EMT33:	.WORD	EMS11,EMS41,EMS30,0
064730	071426	072410	072042	EMT34:	.WORD	EMS11,EMS42,EMS30,0
064740	071426	072437	072042	EMT35:	.WORD	EMS11,EMS43,EMS30,0
064750	071426	072466	072042	EMT36:	.WORD	EMS11,EMS44,EMS30,0
064760	071426	072541	072042	EMT37:	.WORD	EMS11,EMS45,EMS30,0
064770	073331	071634	000000	EMT40:	.WORD	EMS66,EMS20,0
064776	073517	075225	073525	EMT41:	.WORD	EMS75,EMS141,EMS76,0
065006	075316	075326	073453	EMT42:	.WORD	EMS144,EMS145,EMS72,EMS76,0
065020	072633	072751	073525	EMT43:	.WORD	EMS47,EMS53,EMS76,0
065030	073556	072751	073525	EMT44:	.WORD	EMS77,EMS53,EMS76,0
065040	073604	072751	073525	EMT45:	.WORD	EMS100,EMS53,EMS76,0
065050	073632	072751	073525	EMT46:	.WORD	EMS101,EMS53,EMS76,0
065060	073263	073245	000000	EMT47:	.WORD	EMS65,EMS64,0
065066	071500	071522	073217	EMT50:	.WORD	EMS13,EMS15,EMS63,0
065076	071426	072604	072042	EMT51:	.WORD	EMS11,EMS46,EMS30,EMS67,0
065110	072633	072751	073401	EMT52:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
065126	072633	072751	073401	EMT53:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
065144	072662	072751	073401	EMT54:	.WORD	EMS50,EMS53,EMS67,0
065154	075253	075270	072751	EMT55:	.WORD	EMS142,EMS143,EMS53,EMS67,0
065166	072711	073453	073401	EMT56:	.WORD	EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
065204	076057	073463	073401	EMT57:	.WORD	EMS167,EMS73,EMS67,0
065214	073034	073401	074213	EMT60:	.WORD	EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
065232	073440	073034	073401	EMT61:	.WORD	EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
065252	075204	073401	074213	EMT62:	.WORD	EMS140,EMS67,EMS115,EMS47,EMS70,0
065266	000000			EMT63:	.WORD	
065270	075411	075454	073426	EMT64:	.WORD	EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
065310	072737	076504	076665	EMT65:	.WORD	EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
065330	076016	073707	073453	EMT66:	.WORD	EMS165,EMS103,EMS72,EMS124,0

E05

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 263
ERROR MESSAGE TABLE

SEQ 0265

065342	076016	073707	073453	EMT67:	.WORD	EMS165, EMS103, EMS72, EMS171, 0
065354	072604	071634	075711	EMT70:	.WORD	EMS46, EMS20, EMS163, 0
065364	073440	073632	075711	EMT71:	.WORD	EMS71, EMS101, EMS163, 0
065374	072633	073453	075711	EMT72:	.WORD	EMS47, EMS72, EMS163, EMS115, EMS140, EMS141, 0
065412	072633	073453	075711	EMT73:	.WORD	EMS47, EMS72, EMS163, EMS115, EMS141, EMS72, 0
065430	073034	072751	075711	EMT74:	.WORD	EMS56, EMS53, EMS163, 0
065440	073440	073034	075711	EMT75:	.WORD	EMS71, EMS56, EMS163, EMS115, EMS150, EMS152, EMS72, 0
065460	072662	072751	075711	EMT76:	.WORD	EMS50, EMS53, EMS163, 0
065470	075253	075270	072751	EMT77:	.WORD	EMS142, EMS143, EMS53, EMS163, 0
065502	073517	075225	075711	EMT100:	.WORD	EMS75, EMS141, EMS163, EMS115, EMS47, EMS70, 0
065520	073517	075411	075711	EMT101:	.WORD	EMS75, EMS150, EMS163, EMS115, EMS56, EMS73, 0
065536	076057	073463	075711	EMT102:	.WORD	EMS167, EMS73, EMS163, 0
065546	075374	075430	073500	EMT103:	.WORD	EMS147, EMS151, EMS74, EMS163, 0
065560	072711	073500	075711	EMT104:	.WORD	EMS51, EMS74, EMS163, EMS115, EMS50, EMS70, 0
065576	076016	073604	075711	EMT105:	.WORD	EMS165, EMS100, EMS163, 0
065606	076016	073556	075711	EMT106:	.WORD	EMS165, EMS77, EMS163, 0
065616	075316	075326	072751	EMT107:	.WORD	EMS144, EMS145, EMS53, EMS163, EMS115, EMS143, EMS70, 0
065636	074013	074053	074220	EMT110:	.WORD	EMS110, EMS112, EMS116, EMS111, 0
065650	074137	071157	000000	EMT111:	.WORD	EMS113, EMS4, 0
065656	074137	071114	000000	EMT112:	.WORD	EMS113, EMS3, 0
065664	074013	074213	074220	EMT113:	.WORD	EMS110, EMS115, EMS116, EMS117, EMS114, 0
065700	074137	074272	000000	EMT114:	.WORD	EMS113, EMS120, 0
065706	074307	074747	074773	EMT115:	.WORD	EMS121, EMS132, EMS133, 0
065716	074352	074747	074773	EMT116:	.WORD	EMS122, EMS132, EMS133, 0
065726	074407	074747	074773	EMT117:	.WORD	EMS123, EMS132, EMS133, 0
065736	074452	074747	074773	EMT120:	.WORD	EMS124, EMS132, EMS133, 0
065746	074504	074747	074773	EMT121:	.WORD	EMS125, EMS132, EMS133, 0
065756	074547	074747	074773	EMT122:	.WORD	EMS126, EMS132, EMS133, 0
065766	075147	074747	074773	EMT123:	.WORD	EMS137, EMS132, EMS133, 0
065776	074651	074747	074773	EMT124:	.WORD	EMS130, EMS132, EMS133, 0
066006	074707	074747	074773	EMT125:	.WORD	EMS131, EMS132, EMS133, 0
066016	074307	074747	075016	EMT126:	.WORD	EMS121, EMS132, EMS134, EMS123, 0
066030	074352	074747	075016	EMT127:	.WORD	EMS122, EMS132, EMS134, EMS123, 0
066042	074407	074747	075016	EMT130:	.WORD	EMS123, EMS132, EMS134, EMS123, 0
066054	074452	074747	075016	EMT131:	.WORD	EMS124, EMS132, EMS134, EMS123, 0
066066	074504	074747	075016	EMT132:	.WORD	EMS125, EMS132, EMS134, EMS123, 0
066100	074547	074747	075016	EMT133:	.WORD	EMS126, EMS132, EMS134, EMS123, 0
066112	075147	074747	075016	EMT134:	.WORD	EMS137, EMS132, EMS134, EMS123, 0
066124	074651	074747	075016	EMT135:	.WORD	EMS130, EMS132, EMS134, EMS123, 0
066136	074707	074747	075016	EMT136:	.WORD	EMS131, EMS132, EMS134, EMS123, 0
066150	074307	074747	075060	EMT137:	.WORD	EMS121, EMS132, EMS135, 0
066160	074407	074747	075060	EMT140:	.WORD	EMS123, EMS132, EMS135, 0
066170	074307	074747	075122	EMT141:	.WORD	EMS121, EMS132, EMS136, 0
066200	075147	074747	075122	EMT142:	.WORD	EMS137, EMS132, EMS136, 0
066210	074452	074747	075122	EMT143:	.WORD	EMS124, EMS132, EMS136, 0
066220	074504	074747	075122	EMT144:	.WORD	EMS125, EMS132, EMS136, 0
066230	074547	074747	075122	EMT145:	.WORD	EMS126, EMS132, EMS136, 0
066240	074707	074747	075122	EMT146:	.WORD	EMS131, EMS132, EMS136, 0
066250	075656	074747	075122	EMT147:	.WORD	EMS162, EMS132, EMS136, 0
066260	074651	074747	075122	EMT150:	.WORD	EMS130, EMS132, EMS136, 0
066270	075204	074213	075225	EMT151:	.WORD	EMS140, EMS115, EMS141, EMS70, 0
066302	075253	074213	075270	EMT152:	.WORD	EMS142, EMS115, EMS143, EMS72, 0
066314	075316	075326	075355	EMT153:	.WORD	EMS144, EMS145, EMS146, EMS115, EMS143, EMS72, 0
066332	075316	075326	071634	EMT154:	.WORD	EMS144, EMS145, EMS20, EMS115, EMS143, EMS70, 0
066350	075411	075454	075522	EMT155:	.WORD	EMS150, EMS152, EMS154, EMS153, 0
066362	075374	075430	075522	EMT156:	.WORD	EMS147, EMS151, EMS154, EMS155, 0

066374	075374	075430	075556	EMT157: .WORD	EMS147,EMS151,EMS156,EMS157,0
066406	075626	075556	075611	EMT160: .WORD	EMS161,EMS156,EMS160,0
066416	071773	072031	075556	EMT161: .WORD	EMS25,EMS27,EMS156,EMS160,0
066430	072010	072031	075556	EMT162: .WORD	EMS26,EMS27,EMS156,EMS160,0
066442	072633	075711	075204	EMT163: .WORD	EMS47,EMS163,EMS140,0
066452	072633	071634	000000	EMT164: .WORD	EMS47,EMS20,0
066460	073034	071634	000000	EMT165: .WORD	EMS56,EMS20,0
066466	072604	071634	000000	EMT166: .WORD	EMS46,EMS20,0
066474	077205	071634	000000	EMT167: .WORD	EMS224,EMS20,0
066502	076057	075522	076032	EMT170: .WORD	EMS167,EMS154,EMS166,0
066512	076057	075522	075574	EMT171: .WORD	EMS167,EMS154,EMS157,0
066522	076057	075522	075536	EMT172: .WORD	EMS167,EMS154,EMS155,0
066532	076133	074747	074773	EMT173: .WORD	EMS171,EMS132,EMS133,0
066542	076133	074747	075016	EMT174: .WORD	EMS171,EMS132,EMS134,EMS123,0
066554	074137	076306	000000	EMT175: .WORD	EMS113,EMS177,0
066562	076330	076345	000000	EMT176: .WORD	EMS200,EMS201,0
066570	076443	075225	072042	EMT177: .WORD	EMS203,EMS141,EMS30,EMS202,0
066602	076443	072711	072042	EMT200: .WORD	EMS203,EMS51,EMS30,EMS202,0
066614	076443	076456	072042	EMT201: .WORD	EMS203,EMS204,EMS30,EMS202,0
066626	076443	072662	072042	EMT202: .WORD	EMS203,EMS50,EMS30,EMS202,0
066640	076443	075270	072042	EMT203: .WORD	EMS203,EMS143,EMS30,EMS202,0
066652	075411	073500	076413	EMT204: .WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
066670	072662	073707	073453	EMT205: .WORD	EMS50,EMS103,EMS72,0
066700	073716	073463	076413	EMT206: .WORD	EMS104,EMS73,EMS202,0
066710	073517	075225	074213	EMT207: .WORD	EMS75,EMS141,EMS115,EMS140,0
066722	073517	075411	000000	EMT210: .WORD	EMS75,EMS150,0
066730	072711	073453	074213	EMT211: .WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
066744	075253	072751	074213	EMT212: .WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
066760	072662	073707	072751	EMT213: .WORD	EMS50,EMS103,EMS53,0
066770	072737	076504	076032	EMT214: .WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
067010	072737	074250	073034	EMT215: .WORD	EMS52,EMS117,EMS56,EMS163,0
067022	073034	072042	073245	EMT216: .WORD	EMS56,EMS30,EMS64,0
067032	072604	072042	073245	EMT217: .WORD	EMS46,EMS30,EMS64,0
067042	072052	072042	073245	EMT220: .WORD	EMS31,EMS30,EMS64,0
067052	072633	072042	073245	EMT221: .WORD	EMS47,EMS30,EMS64,0
067062	072737	074250	073556	EMT222: .WORD	EMS52,EMS117,EMS77,0
067072	072737	074250	073217	EMT223: .WORD	EMS52,EMS117,EMS63,0
067102	000000			EMT224: .WORD	
067104	000000			EMT225: .WORD	
067106	000000			EMT226: .WORD	
067110	000000			EMT227: .WORD	
067112	000000			EMT230: .WORD	
067114	000000			EMT231: .WORD	
067116	000000			EMT232: .WORD	
067120	000000			EMT233: .WORD	
067122	000000			EMT234: .WORD	
067124	000000			EMT235: .WORD	
067126	000000			EMT236: .WORD	
067130	000000			EMT237: .WORD	
067132	000000			EMT240: .WORD	
067134	000000			EMT241: .WORD	
067136	000000			EMT242: .WORD	
067140	000000			EMT243: .WORD	
067142	000000			EMT244: .WORD	
067144	000000			EMT245: .WORD	
067146	076057	074747	076543	EMT246: .WORD	EMS167,EMS132,EMS207,0

067156	076057	074747	076570	ENT247: .WORD	EMS167, EMS132, EMS210, EMS125, 0
067170	076057	076601	076570	ENT250: .WORD	EMS167, EMS211, EMS210, EMS207, EMS206, 0
067204	076617	076642	000000	ENT251: .WORD	EMS212, EMS213, 0
067212	076016	076617	000000	ENT252: .WORD	EMS165, EMS212, 0
067220	075374	075430	075556	ENT253: .WORD	EMS147, EMS151, EMS156, EMS210, EMS26, EMS27, 0
067236	076443	073632	072042	ENT254: .WORD	EMS203, EMS101, EMS30, 0
067246	076443	076057	072042	ENT255: .WORD	EMS203, EMS167, EMS30, 0
067256	076443	073604	072042	ENT256: .WORD	EMS203, EMS100, EMS30, 0
067266	071426	072604	072042	ENT257: .WORD	EMS11, EMS46, EMS30, EMS102, 0
067300	072737	074250	073034	ENT260: .WORD	EMSS2, EMS117, EMSS6, EMS102, 0
067312	072737	076504	077007	ENT261: .WORD	EMSS2, EMS205, EMS220, EMS206, EMS115, EMSS1, EMS72, 0
067332	073716	073463	076413	ENT262: .WORD	EMS104, EMS73, EMS202, 0
067342	072662	072751	073660	ENT263: .WORD	EMSS0, EMSS3, EMS102, 0
067352	073034	072751	073660	ENT264: .WORD	EMSS6, EMSS3, EMS102, EMS115, EMS150, EMS152, EMS70, 0
067372	073440	073034	073660	ENT265: .WORD	EMS71, EMSS6, EMS102, EMS115, EMS150, EMS152, EMS72, 0
067412	075253	075270	072751	ENT266: .WORD	EMS142, EMS143, EMSS3, EMS102, 0
067424	072662	075355	074213	ENT267: .WORD	EMSS0, EMS146, EMS115, EMSS2, EMS117, EMS46, 0
067442	072633	072751	073660	ENT270: .WORD	EMS47, EMSS3, EMS102, EMS115, EMS140, 0
067456	072633	072751	073660	ENT271: .WORD	EMS47, EMSS3, EMS102, EMS115, EMS141, EMS72, 0
067474	073517	075225	073660	ENT272: .WORD	EMS75, EMS141, EMS102, EMS115, EMS47, EMS73, 0
067512	072711	073500	073660	ENT273: .WORD	EMSS1, EMS74, EMS102, EMS115, EMSS0, EMS70, 0
067530	073217	072751	073111	ENT274: .WORD	EMS63, EMSS3, EMSS7, EMS115, EMS41, EMS146, 0
067546	074220	072751	072362	ENT275: .WORD	EMS116, EMSS3, EMS41, EMSS7, 0
067560	072633	072751	073111	ENT276: .WORD	EMS47, EMSS3, EMSS7, EMS115, EMS140, 0
067574	072633	072751	073111	ENT277: .WORD	EMS47, EMSS3, EMSS7, EMS115, EMS141, EMS72, 0
067612	073034	072751	073111	ENT300: .WORD	EMSS6, EMSS3, EMSS7, EMS115, EMS150, EMS152, EMS70, 0
067632	073440	073034	072751	ENT301: .WORD	EMS71, EMSS6, EMSS3, EMSS7, EMS115, EMS150, EMS152, EMS72, 0
067654	076016	073604	073707	ENT302: .WORD	EMS165, EMS100, EMS103, EMSS7, 0
067666	076016	073632	073707	ENT303: .WORD	EMS165, EMS101, EMS103, EMSS7, 0
067700	076016	073556	073707	ENT304: .WORD	EMS165, EMS77, EMS103, EMSS7, 0
067712	072604	072042	073245	ENT305: .WORD	EMS46, EMS30, EMS64, EMSS7, 0
067724	072662	072751	073111	ENT306: .WORD	EMSS0, EMSS3, EMSS7, 0
067734	072662	075355	074213	ENT307: .WORD	EMSS0, EMS146, EMS115, EMSS2, EMS117, EMS46, EMSS7, 0
067754	075253	075270	072751	ENT310: .WORD	EMS142, EMS143, EMSS3, EMSS7, 0
067766	073716	073707	072751	ENT311: .WORD	EMS104, EMS103, EMSS3, EMSS7, 0
070000	073745	073707	072751	ENT312: .WORD	EMS105, EMS103, EMSS3, EMSS7, 0
070012	075316	075326	073707	ENT313: .WORD	EMS144, EMS145, EMS103, EMSS7, EMS115, EMS143, EMS70, 0
070032	072410	073707	072751	ENT314: .WORD	EMS42, EMS103, EMSS3, EMSS7, 0
070044	072052	073707	072751	ENT315: .WORD	EMS31, EMS103, EMSS3, EMSS7, 0
070056	073440	072052	073707	ENT316: .WORD	EMS71, EMS31, EMS103, EMSS7, 0
070070	072437	073707	073111	ENT317: .WORD	EMS43, EMS103, EMSS7, 0
070100	072541	073707	073111	ENT320: .WORD	EMS45, EMS103, EMSS7, 0
070110	072466	073707	073111	ENT321: .WORD	EMS44, EMS103, EMSS7, 0
070120	073774	071634	000000	ENT322: .WORD	EMS106, EMS20, 0
070126	072256	073707	073111	ENT323: .WORD	EMS36, EMS103, EMSS7, 0
070136	076206	072256	073707	ENT324: .WORD	EMS173, EMS36, EMS103, EMSS7, 0
070150	076166	072256	073707	ENT325: .WORD	EMS172, EMS36, EMS103, EMSS7, 0
070162	071500	076223	071522	ENT326: .WORD	EMS13, EMS174, EMS15, EMS35, EMSS3, EMS175, 0
070200	075374	075430	073500	ENT327: .WORD	EMS147, EMS151, EMS74, EMS175, 0
070212	073331	072751	076231	ENT330: .WORD	EMS66, EMSS3, EMS175, 0
070222	072130	073707	072751	ENT331: .WORD	EMS33, EMS103, EMSS3, EMS175, 0
070234	072333	073707	072751	ENT332: .WORD	EMS40, EMS103, EMSS3, EMSS7, 0
070246	072711	073500	073111	ENT333: .WORD	EMSS1, EMS74, EMSS7, EMS115, EMSS0, EMS70, 0
070264	073517	075225	073111	ENT334: .WORD	EMS75, EMS141, EMSS7, EMS115, EMS47, EMS73, 0
070302	073517	075411	075454	ENT335: .WORD	EMS75, EMS150, EMS152, EMSS7, EMS115, EMSS6, EMS73, 0
070322	073137	073152	073201	ENT336: .WORD	EMS60, EMS61, EMS62, 0

070332	074220	074250	072156	EMT337: .WORD	EMS116, EMS117, EMS34, 0
070342	072156	072751	072763	EMT340: .WORD	EMS34, EMS53, EMS54, EMS111, 0
070354	073005	072156	000000	EMT341: .WORD	EMS55, EMS34, 0
070362	072737	074250	073034	EMT342: .WORD	EMS52, EMS117, EMS56, EMS57, 0
070374	072737	074250	072541	EMT343: .WORD	EMS52, EMS117, EMS45, EMS57, 0
070406	072737	074250	072466	EMT344: .WORD	EMS52, EMS117, EMS44, EMS57, 0
070420	072737	074250	077027	EMT345: .WORD	EMS52, EMS117, EMS221, 0
070430	073774	071634	074213	EMT346: .WORD	EMS106, EMS20, EMS115, EMS223, EMS72, 0
070444	072737	076504	077077	EMT347: .WORD	EMS52, EMS205, EMS222, EMS206, 0
070456	073517	075411	073660	EMT350: .WORD	EMS75, EMS150, EMS102, EMS115, EMS56, EMS73, 0
070474	076057	073463	073660	EMT351: .WORD	EMS167, EMS73, EMS102, 0
070504	076703	000000		EMT352: .WORD	EMS215, 0
070510	076754	076443	076724	EMT353: .WORD	EMS217, EMS203, EMS216, 0
070520	077027	071634	000000	EMT354: .WORD	EMS221, EMS20, 0

070526	077257	100073	100154	EHT1:	.WORD	EH1,STSH1,STSH2,STSH4,0
070540	100073	100154	100307	EHT2:	.WORD	STSH1,STSH2,STSH4,0
070550	077276	000000		EHT110:	.WORD	EH110,0
070554	077305	000000		EHT111:	.WORD	EH111,0
070560	077324	000000		EHT114:	.WORD	EH114,0
070564	077353	100073	100154	EHT223:	.WORD	EH223,STSH1,STSH2,STSH4,0
070576	077401	100073	100154	EHT256:	.WORD	EH256,STSH1,STSH2,STSH4,0
070610	077456	100073	100154	EHT336:	.WORD	EH336,STSH1,STSH2,STSH4,0
070622	077515	100073	100154	EHT337:	.WORD	EH337,STSH1,STSH2,STSH4,0
070634	077654	100073	100154	EHT344:	.WORD	EH344,STSH1,STSH2,STSH4,0
070646	100014	000000		EHT353:	.WORD	EH353,0

070652	100350	100444	100462	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
070662	100444	100462	100514	EDT2:	.WORD	STSD1,STSD2,STSD4
070670	100356			EDT110:	.WORD	ED110
070672	100362			EDT111:	.WORD	ED111
070674	100370			EDT114:	.WORD	ED114
070676	100400	100444	100462	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
070706	100410	100444	100462	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
070716	100422	100444	100462	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
070726	100422	100444	100462	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
070740	100434			EDT353:	.WORD	ED353

K05

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 269
ERROR MESSAGE TABLE

SEQ 0271

070742	100527	100545	100545	EFT1:	.WORD	EF111,STSF,STSF,STSF
070752	100545	100545	100545	EFT2:	.WORD	STSF,STSF,STSF
070760	100526			EFT110:	.WORD	EF110
070762	100527			EFT111:	.WORD	EF111
070764	100531			EFT114:	.WORD	EF114
070766	100531	100545	100545	EFT223:	.WORD	EF114,STSF,STSF,STSF
070776	100534	100545	100545	EFT336:	.WORD	EF336,STSF,STSF,STSF
071006	100534	100545	100545	EFT337:	.WORD	EF336,STSF,STSF,STSF
071016	100534	100545	100545	EFT344:	.WORD	EF336,STSF,STSF,STSF
071026	100531			EFT353:	.WORD	EF114

.SBTTL ERROR MESSAGE STRINGS

071030	051127	047117	020107	EMS1:	.ASCIZ	WRONG UNIT SELECTED (RMCS2, BITS 0-2) a
071077	104	053105	041511	EMS2:	.ASCIZ	DEVICE WENT a
071114	047125	053101	044501	EMS3:	.ASCIZ	UNAVAILABLE "DVA" (RMCS1, BIT 11) a
071157	116	047117	054105	EMS4:	.ASCIZ	NONEXISTENT "NED" (RMCS2, BIT 12) a
071222	047503	046515	047101	EMS5:	.ASCIZ	COMMAND NOT COMPLETED, a
071252	047503	052116	047522	EMS6:	.ASCIZ	CONTROLLER NOT READY (RMCS1, BIT 7) a
071317	104	044522	042526	EMS7:	.ASCIZ	DRIVE NOT READY "DRY" (RMDS, BIT 7) a
071364	047507	047040	052117	EMS10:	.ASCIZ	GO NOT RESET "GO" (RMCS1, BIT 0) a
071426	047111	040526	044514	EMS11:	.ASCIZ	INVALID a
071437	106	047125	052103	EMS12:	.ASCIZ	FUNCTION CODE (RMCS1, BITS 1-5) a
071500	040515	051523	052502	EMS13:	.ASCIZ	MASSBUS a
071511	103	047117	051124	EMS14:	.ASCIZ	CONTROL a
071522	052502	020123	040520	EMS15:	.ASCIZ	BUS PARITY ERROR a
071544	046442	050103	021105	EMS16:	.ASCIZ	"MCPE" (RMCS1, BIT 13) a
071574	051124	047101	043123	EMS17:	.ASCIZ	TRANSFER ERROR (RMCS1, BIT 14) a
071634	044123	052517	042114	EMS20:	.ASCIZ	SHOULD NOT BE SET a
071657	127	051117	020104	EMS21:	.ASCIZ	WORD COUNT (RMWC) a
071702	052502	020123	051050	EMS22:	.ASCIZ	BUS (RMA) a
071716	046042	052102	020042	EMS23:	.ASCIZ	"LBT" (RMDS, BIT 10) a
071744	040442	042517	020042	EMS24:	.ASCIZ	"AOE" (RMER1, BIT 09) a
071773	104	051511	020113	EMS25:	.ASCIZ	DISK (RMDA) a
072010	054503	044514	042116	EMS26:	.ASCIZ	CYLINDER (RMDC) a
072031	101	042104	042522	EMS27:	.ASCIZ	ADDRESS a
072042	052123	052101	051525	EMS30:	.ASCIZ	STATUS a
072052	053442	042514	020042	EMS31:	.ASCIZ	"MLE" (RMER1, BIT 11) a
072101	042	050125	021105	EMS32:	.ASCIZ	"UPE" (RMCS2, BIT 13) a
072130	053442	043103	020042	EMS33:	.ASCIZ	"WCF" (RMER1, BIT 5) a
072156	051127	052111	020105	EMS34:	.ASCIZ	WRITE CHECK ERROR-"WCE" (RMCS2, BIT 14) a
072227	042	042115	042520	EMS35:	.ASCIZ	"MDPE" (RMCS2, BIT 8) a
072256	042042	045503	020042	EMS36:	.ASCIZ	"DCK" (RMER1, BIT 15) a
072305	042	041505	021110	EMS37:	.ASCIZ	"ECH" (RMER1, BIT 6) a
072333	042	046104	021124	EMS40:	.ASCIZ	"DLT" (RMCS2, BIT 15) a
072362	046442	043130	020042	EMS41:	.ASCIZ	"MXF" (RMCS2, BIT 9) a
072410	042042	042524	020042	EMS42:	.ASCIZ	"DTE" (RMER1, BIT 12) a
072437	042	041510	041522	EMS43:	.ASCIZ	"HCRC" (RMER1, BIT 8) a
072466	042510	042101	051105	EMS44:	.ASCIZ	HEADER COMPARE ERROR "HCE" (RMER1, BIT 7) a
072541	106	051117	040515	EMS45:	.ASCIZ	FORMAT ERROR "FER" (RMER1, BIT 4) a
072604	044442	042501	020042	EMS46:	.ASCIZ	"IAE" (RMER1, BIT 10) a
072633	042	050117	021111	EMS47:	.ASCIZ	"OPT" (RMER1, BIT 13) a
072662	051442	044513	020042	EMS50:	.ASCIZ	"SKI" (RMER2, BIT 14) a
072711	042	044520	021120	EMS51:	.ASCIZ	"PIP" (RMDS, BIT 13) a
072737	124	042510	051040	EMS52:	.ASCIZ	THE RM03 a
072751	104	052105	041505	EMS53:	.ASCIZ	DETECTED a
072763	101	020124	047101	EMS54:	.ASCIZ	AT AN UNEXPECTED a
073005	111	041516	051117	EMS55:	.ASCIZ	INCORRECT DATA DURING a
073034	047111	040526	044514	EMS56:	.ASCIZ	INVALID COMMAND ERROR "IVC" (RMER2, BIT 12) a
073111	104	051125	047111	EMS57:	.ASCIZ	DURING DATA TRANSFER a
073137	104	052101	020101	EMS60:	.ASCIZ	DATA READ a
073152	047504	051505	047040	EMS61:	.ASCIZ	DOES NOT COMPARE WITH a
073201	104	052101	020101	EMS62:	.ASCIZ	DATA WRITTEN a
073217	042	040520	021122	EMS63:	.ASCIZ	"PAR" (RMER1, BIT 3) a
073245	111	020123	047111	EMS64:	.ASCIZ	IS INCORRECT a
073263	103	046517	047520	EMS65:	.ASCIZ	COMPOSITE ERROR "ERR" (RMDS, BIT 14) a
073331	104	052101	020101	EMS66:	.ASCIZ	DATA PARITY ERROR "DPE" (RMER2, BIT 3) a

073401	104	051125	047111	EMS67:	.ASCIZ	2DURING SEEK COMMAND 2
073426	051511	051040	051505	EMS70:	.ASCIZ	2IS RESET 2
073440	051105	047522	042516	EMS71:	.ASCIZ	2ERRONEOUS 2
073453	111	020123	042523	EMS72:	.ASCIZ	2IS SET 2
073463	104	042111	047040	EMS73:	.ASCIZ	2DID NOT SET 2
073500	044504	020104	047516	EMS74:	.ASCIZ	2DID NOT RESET 2
073517	114	051517	020124	EMS75:	.ASCIZ	2LOST 2
073525	104	051125	047111	EMS76:	.ASCIZ	2DURING PACK ACK COMMAND 2
073556	051042	051115	020042	EMS77:	.ASCIZ	2"RM" (RMR1, BIT 2) 2
073604	044442	051114	020042	EMS100:	.ASCIZ	2"ILR" (RMR1, BIT 1) 2
073632	044442	043114	020042	EMS101:	.ASCIZ	2"ILF" (RMR1, BIT 0) 2
073660	052504	044522	043516	EMS102:	.ASCIZ	2DURING SEARCH COMMAND 2
073707	105	051122	051117	EMS103:	.ASCIZ	2ERROR 2
073716	046042	041502	020042	EMS104:	.ASCIZ	2"LBC" (RMR2, BIT 10) 2
073745	042	051514	021103	EMS105:	.ASCIZ	2"LSC" (RMR2, BIT 11) 2
073774	042510	042101	051105	EMS106:	.ASCIZ	2HEADER ERRORS 2
074013	102	051525	052040	EMS110:	.ASCIZ	2BUS TIMEOUT (04 TRAP) 2
074042	042101	051104	051505	EMS111:	.ASCIZ	2ADDRESS 2
074053	127	042510	020116	EMS112:	.ASCIZ	2WHEN READING/WRITING RH REGISTERS 2
074115	101	020124	044124		.ASCIZ	2AT THE FOLLOWING 2
074137	124	042510	051440	EMS113:	.ASCIZ	2THE SELECTED DEVICE IS 2
074167	116	047117	054105	EMS114:	.ASCIZ	2NONEXISTENT DEVICE 2
074213	040	006455	000012	EMS115:	.ASCIZ	2 -2<CR><LF>
074220	044124	020105	040515	EMS116:	.ASCIZ	2THE MASSBUS CONTROLLER 2
074250	040506	046111	042105	EMS117:	.ASCIZ	2FAILED TO DETECT 2
074272	047516	020124	047101	EMS120:	.ASCIZ	2NOT AN RMD3 2
074307	103	047117	051124	EMS121:	.ASCIZ	2CONTROL STATUS REGISTER 1, RMCS1, 2
074352	052502	020123	042101	EMS122:	.ASCIZ	2BUS ADDRESS REGISTER, RMB, 2
074407	103	047117	051124	EMS123:	.ASCIZ	2CONTROL STATUS REGISTER 2, RMCS2, 2
074452	051105	047522	020122	EMS124:	.ASCIZ	2ERROR REGISTER 1, RMR1, 2
074504	052101	042524	052116	EMS125:	.ASCIZ	2ATTENTION SUMMARY REGISTER, RMAS, 2
074547	115	044501	052116	EMS126:	.ASCIZ	2MAINTENANCE REGISTER #1, RMR #1, 2
074612	041505	020103	047520	EMS127:	.ASCIZ	2ECC POSITION REGISTER, RMEC1, 2
074651	105	041503	050040	EMS130:	.ASCIZ	2ECC PATTERN REGISTER, RMEC2, 2
074707	115	044501	052116	EMS131:	.ASCIZ	2MAINTENANCE REGISTER 2, RMR2, 2
074747	116	052117	044440	EMS132:	.ASCIZ	2NOT INITIALIZED BY 2
074773	125	044516	052502	EMS133:	.ASCIZ	2UNIBUS INITIALIZE 2
075016	047503	052116	047522	EMS134:	.ASCIZ	2CONTROLLER CLEAR, I.E. BIT 5 OF 2
075060	044122	030461	042440	EMS135:	.ASCIZ	2RH11 ERROR CLEAR (RMCS1, BIT 14) 2
075122	051104	053111	020105	EMS136:	.ASCIZ	2DRIVE CLEAR COMMAND 2
075147	104	044522	042526	EMS137:	.ASCIZ	2DRIVE STATUS REGISTER, RMD5 2
075204	042515	044504	046525	EMS140:	.ASCIZ	2MEDIUM OFF LINE 2
075225	042	047515	021114	EMS141:	.ASCIZ	2"ML" (RMD5, BIT 12) 2
075253	104	044522	042526	EMS142:	.ASCIZ	2DRIVE FAULT 2
075270	042042	041526	020042	EMS143:	.ASCIZ	2"DVC" (RMR2, BIT 7) 2
075316	047125	040523	042506	EMS144:	.ASCIZ	2UNSAFE 2
075326	052442	051516	020042	EMS145:	.ASCIZ	2"UNS" (RMR1, BIT 14) 2
075355	123	047510	046125	EMS146:	.ASCIZ	2SHOULD BE SET 2
075374	043117	051506	052105	EMS147:	.ASCIZ	2OFFSET MODE 2
075411	040	047526	052514	EMS150:	.ASCIZ	2 VOLUME VALID 2
075430	047442	021115	024040	EMS151:	.ASCIZ	2"OM" (RMD5, BIT 0) 2
075454	053042	021126	024040	EMS152:	.ASCIZ	2"VV" (RMD5, BIT 6) 2
075500	040520	045503	040440	EMS153:	.ASCIZ	2PACK ACK COMMAND 2
075522	047516	020124	042523	EMS154:	.ASCIZ	2NOT SET BY 2
075536	043117	051506	052105	EMS155:	.ASCIZ	2OFFSET COMMAND 2
075556	047516	020124	042522	EMS156:	.ASCIZ	2NOT RESET BY 2

075574	052122	020103	047503	EMS157:	.ASCIZ	2BIC COMMAND 2
075611	122	050111	041440	EMS160:	.ASCIZ	2RIP COMMAND 2
075626	043117	051506	052105	EMS161:	.ASCIZ	2OFFSET REGISTER (RMOF) 2
075656	051105	047522	020122	EMS162:	.ASCIZ	2ERROR REGISTER #2, RMR2, 2
075711	104	051125	047111	EMS163:	.ASCIZ	2DURING RECALIBRATE 2
075735	111	020123	047111	EMS164:	.ASCII	2IS INTERMITTENT OR DRIVE DIDNT DROP ON 2
076004	054503	044514	042116		.ASCIZ	2CYLINDER 2
076016	047125	054105	042520	EMS165:	.ASCIZ	2UNEXPECTED 2
076032	042522	040503	044514	EMS166:	.ASCIZ	2RECALIBRATE COMMAND 2
076057	042	052101	021101	EMS167:	.ASCIZ	2"ATA" (RMS, BIT15) 2
076104	044127	047105	051040	EMS170:	.ASCIZ	2WHEN READING REGISTER 2
076133	105	051122	051117	EMS171:	.ASCIZ	2ERROR REGISTER #2, RMR2, 2
076166	047516	051116	041505	EMS172:	.ASCIZ	2NONRECOVERABLE 2
076206	042522	047503	042526	EMS173:	.ASCIZ	2RECOVERABLE 2
076223	104	052101	020101	EMS174:	.ASCIZ	2DATA 2
076231	104	051125	047111	EMS175:	.ASCIZ	2DURING WRITE COMMAND 2
076257	042	050117	021105	EMS176:	.ASCIZ	2"OPE" (RMR2, BIT 13) 2
076306	047111	053440	044522	EMS177:	.ASCIZ	2IN WRITE PROTECT 2
076330	040503	020116	047516	EMS200:	.ASCIZ	2CAN NOT SET 2
076345	104	040511	047107	EMS201:	.ASCIZ	2DIAGNOSTIC MODE "DMD" (RMMR1, BIT 0) 2
076413	104	051125	047111	EMS202:	.ASCIZ	2DURING DIAGNOSTIC MODE 2
076443	111	041516	051117	EMS203:	.ASCIZ	2INCORRECT 2
076456	053442	046122	020042	EMS204:	.ASCIZ	2"ML" (RMS, BIT 11) 2
076504	054105	041505	052125	EMS205:	.ASCIZ	2EXECUTED 2
076516	044527	044124	041440	EMS206:	.ASCIZ	2WITH COMP ERROR SET 2
076543	042	047507	020042	EMS207:	.ASCIZ	2"GO" (RMCS1, BIT 0) 2
076570	051127	052111	047111	EMS210:	.ASCIZ	2WRITING 2
076601	127	051501	051040	EMS211:	.ASCIZ	2WAS RESET BY 2
076617	120	047522	051107	EMS212:	.ASCIZ	2PROGRAM INTERRUPT 2
076642	040527	020123	047516	EMS213:	.ASCIZ	2WAS NOT GENERATED 2
076665	123	042505	020113	EMS214:	.ASCIZ	2SEEK COMMAND 2
076703	120	047522	051107	EMS215:	.ASCIZ	2PROGRAM TIMEOUT 2
076724	052504	044522	043516	EMS216:	.ASCIZ	2DURING LOOK AHEAD TEST 2
076754	047514	045517	040440	EMS217:	.ASCIZ	2LOOK AHEAD REGISTER, RMLA, 2
077007	123	040505	041522	EMS220:	.ASCIZ	2SEARCH COMMAND 2
077027	102	042101	051440	EMS221:	.ASCIZ	2BAD SECTOR ERROR "BSE" (RMR2, BIT 15) 2
077077	101	042040	052101	EMS222:	.ASCIZ	2A DATA TRANSFER COMMAND 2
077130	042510	042101	051105	EMS223:	.ASCIZ	2HEADER COMPARE INHIBIT "HCI" (RMOF, BIT 10) 2
077205	116	047117	054105	EMS224:	.ASCIZ	2NONEXISTENT MEMORY "NEM" (RMCS2, BIT 11) 2

077257	105	050130	052103	EH1:	.ASCIZ	RECEVD	RECEVD				
077276	052502	040523	051104	EH110:	.ASCIZ	RECEVD	RECEVD				
077305	040	046522	051503	EH111:	.ASCIZ	RMCS2	RMCS1				
077324	042522	042503	042126	EH114:	.ASCIZ	RECEVD	SNGPRT	DULPRT			
077353	105	050130	052103	EH223:	.ASCIZ	RECEVD	RECEVD	DATA			
077401	105	050130	052103	EH256:	.ASCII	RECEVD	RECEVD	RGSTR	<CR><LF>		
077430	052123	052101	051525		.ASCIZ	STATUS	STATUS	INDEX			
077456	042107	042101	051522	EH336:	.ASCIZ	GDADR	GDADR	BDADR	BDDATA		
077515	122	041515	031123	EH337:	.ASCII	RMCS2	STATUS	FAILING	DATA	<CR><LF>	
077555	137	057537	057537		.ASCII	*****	*****	*****	*****	<CR><LF>	
077615	105	050130	052103		.ASCIZ	RECEVD	RECEVD	BIT	ADRESS		
077654	046522	051105	020061	EH344:	.ASCII	RMER1	STATUS	HEADER	FAILING	<CR><LF>	
077715	137	057537	057537		.ASCII	*****	*****	WORD	BIT	<CR><LF>	
077754	054105	041520	042124		.ASCIZ	RECEVD	RECEVD	NUMBER	POSITION		
100014	054105	041520	042124	EH353:	.ASCII	RECEVD	RECEVD	<CR><LF>			
100034	041474	037122	046074		.ASCIZ	<CR><LF>	RMLA	RMLA	RMOF		
100073	040	046522	051503	STSH1:	.ASCII	RMCS1	RMCS2	RMD5	RMER1	RMER2	<CR><LF>
100142	020040	051040	040515		.ASCIZ	RMAS					<CR><LF>
100154	051040	053515	020103	STSH2:	.ASCII	RMWC	RMBA	RMDA	RMOF	RMDCA	<CR><LF>
100223	040	020040	051040		.ASCIZ	RMEC1	RMEC2				<CR><LF>
100247	040	046522	040504	STSH3:	.ASCIZ	RMDA	RMDC	RMOF	RMLA		<CR><LF>
100307	040	046522	051115	STSH4:	.ASCIZ	RMMR1	RMMR2	RMDT	RMSNA		<CR><LF>

	100350			.EVEN		
100350	001140	001142	000000	ED1:	.WORD	SGDDAT, SBDDAT, 0
100356	001276	000000		ED110:	.WORD	SBASE, 0
100362	001174	001176	000000	ED111:	.WORD	STMP0, STMP1, 0
100370	001354	001176	001200	ED114:	.WORD	RMDTI, STMP1, STMP2, 0
100400	001140	001142	001174	ED223:	.WORD	SGDDAT, SBDDAT, STMP0, 0
100410	001134	001140	001136	ED336:	.WORD	SGDADR, SGDDAT, SBDADR, SBDDAT, 0
100422	001140	001142	001174	ED337:	.WORD	SGDDAT, SBDDAT, STMP0, STMP1, 0
100434	001140	001142	001430	ED353:	.WORD	SGDDAT, SBDDAT, RMOFO, 0
100444	001326	001336	001340	STSD1:	.WORD	RMCS1I, RMCS2I, RMDSI, RMER1I, RMER2I, RMASI, 0
100462	001330	001332	001334	STSD2:	.WORD	RMNCI, RMBAI, RMDAI, RMOFI, RMDCI, RMECI
100476	001374	000000			.WORD	RMEC2I, 0
100502	001334	001362	001360	STSD3:	.WORD	RMDAI, RMDCI, RMOFI, RMLAI, 0
100514	001352	001366	001354	STSD4:	.WORD	RMMR1I, RMMR2I, RMDTI, RMSNI, 0

006

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 275
ERROR MESSAGE STRINGS

SEQ 0277

100526	000			EF110:	.BYTE	0
100527	000	000		EF111:	.BYTE	0,0
100531	000	000	000	EF114:	.BYTE	0,0,0
100534	000	000	000	EF336:	.BYTE	0,0,0,0
100540	000	000	000	EF337:	.BYTE	0,0,0,0,0
100545	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0

E06

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 276
ERROR MESSAGE STRINGS

SEQ 0278

100554 000402
101560 177777
101562 177777
101564 000402
102570 177777
102572 177777

.EVEN
MFGFIL: .BLKW 258.
 .WORD -1
 .WORD -1
USRFIL: .BLKW 258.
 .WORD -1
 .WORD -1

.EVEN

102574
102574 000402
103600 000402

BUFFER:
BUFO1E: .BLKW 258.
BUFTWO: .BLKW 258.

102574

= BUFFER

102574
102574
102576
102664
102754
102770
103010
103030
103063
103115
103141
103167
103215
103240
103274
103341
103367
103423
103470
103525
103563
103624
103660
103720
103757
104013
104043
104074
104137
104201
104246
104274
104321
104355
104411
104454
104520
104522
104560
104616
104620
104636
104674
104721
104745
105003
105013
105045
105072
105117

005015
040503 052125 047511
040502 020104 042510
044127 047105 042040
046011 051511 020124
057411 057537 057537
030524 041411 047117
124 004462 042504
124 004463 051104
124 004464 047506
124 004465 047506
124 004466 042532
033524 043011 051117
030524 004460 047506
124 030461 043011
124 031061 043011
124 031461 043011
030524 004464 047506
124 032461 043011
124 033061 043011
030524 004467 047506
031124 004460 047506
031124 004461 047506
124 031062 043011
124 031462 043011
124 032062 043011
031124 004465 047506
124 033062 043011
124 033462 043011
031524 004460 047506
031524 004461 053111
124 031063 043011
124 031463 043011
124 032063 043011
031524 004465 047506
005015
047411 042520 040522
057411 057537 057537
005015
053523 052111 044103
026455 026455 026455
020040 032461 004411
040 030440 004464
040 030440 004463
040 030440 004462
040 030440 004461
040 030440 004460
020040 034440 004411
040 020040 004470

HELP:
.ASCII <CR><LF>
.ASCII @CAUTION: PARTS 2 AND 3 OF THE FUNCTIONAL TEST LEAVE @<CR><LF>
.ASCII @BAD HEADERS ON THE PACK SO BE SURE TO FORMAT THE PACKS@<CR><LF>
.ASCII @WHEN DONE.@<CR><LF>
.ASCII @ LIST OF TESTS@<CR><LF>
.ASCII @ *****@<CR><LF>
.ASCII @T1 CONTROLLER ACCESS TEST@<CR><LF>
.ASCII @T2 DEVICE AVAILABLE TEST@<CR><LF>
.ASCII @T3 DRIVE TYPE TEST@<CR><LF>
.ASCII @T4 FORMAT ZEROS - 18@<CR><LF>
.ASCII @T5 FORMAT ZEROS - 16@<CR><LF>
.ASCII @T6 ZERO FILL TEST@<CR><LF>
.ASCII @T7 FORMAT CHECK ZEROS - 16@<CR><LF>
.ASCII @T10 FORMAT CHECK ZEROS W/ WCE ERROR@<CR><LF>
.ASCII @T11 FORMAT ONES - 16@<CR><LF>
.ASCII @T12 FORMAT CHECK ONES - 16@<CR><LF>
.ASCII @T13 FORMAT CHECK ONES W/ WCE ERROR@<CR><LF>
.ASCII @T14 FORMAT MULTIPLE SECTORS@<CR><LF>
.ASCII @T15 FORMAT W/ HEAD SWITCHING@<CR><LF>
.ASCII @T16 FORMAT W/ MID TRANSFER SEEK@<CR><LF>
.ASCII @T17 FORMAT W/ IMPLIED SEEK@<CR><LF>
.ASCII @T20 FORMAT EACH SECTOR ADDRESS@<CR><LF>
.ASCII @T21 FORMAT EACH TRACK ADDRESS@<CR><LF>
.ASCII @T22 FORMAT PRIME CYLINDERS@<CR><LF>
.ASCII @T23 FORMAT LAST SECTOR@<CR><LF>
.ASCII @T24 FORMAT W/ AOE ERROR@<CR><LF>
.ASCII @T25 FORMAT INVALID SECTOR ADDRESS@<CR><LF>
.ASCII @T26 FORMAT INVALID TRACK ADDRESS@<CR><LF>
.ASCII @T27 FORMAT INVALID CYLINDER ADDRESS@<CR><LF>
.ASCII @T30 FORMAT AT OFFSET@<CR><LF>
.ASCII @T31 IVC FORMAT TEST@<CR><LF>
.ASCII @T32 FORMAT ERROR TEST - 18@<CR><LF>
.ASCII @T33 FORMAT ERROR TEST - 16@<CR><LF>
.ASCII @T34 FORMAT HCE, FIRST HEADER WORD@<CR><LF>
.ASCII @T35 FORMAT HCE, SECOND HEADER WORD@<CR><LF>
.ASCII <CR><LF>
.ASCII @ OPERATIONAL SWITCH SETTINGS@<CR><LF>
.ASCII @ *****@<CR><LF>
.ASCII <CR><LF>
.ASCII @SWITCH USE@<CR><LF>
-----@<CR><LF>
@ 15 HALT ON ERROR@<CR><LF>
@ 14 LOOP ON TEST@<CR><LF>
@ 13 INHIBIT ERROR TYPEOUTS@<CR><LF>
@<CR><LF>
@ 12 INHIBIT ITERATIONS@<CR><LF>
@ 11 BELL ON ERROR@<CR><LF>
@ 10 LOOP ON ERROR@<CR><LF>
@ 9 LOOP ON TEST IN SWR(7:0)@<CR><LF>
@ 8

G06

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 278
ERROR MESSAGE STRINGS

SEQ 0280

105157	040	020040	004467	.ASCII	2	7	TN1282<CR><LF>
105174	020040	033040	004411	.ASCII	2	6	TN642<CR><LF>
105210	020040	032440	004411	.ASCII	2	5	TN322<CR><LF>
105224	020040	032040	004411	.ASCII	2	4	TN162<CR><LF>
105240	020040	031440	004411	.ASCII	2	3	TN82<CR><LF>
105253	040	020040	004462	.ASCII	2	2	TN42<CR><LF>
105266	020040	030440	004411	.ASCII	2	1	TN22<CR><LF>
105301	040	020040	004460	.ASCII	2	0	TN12<CR><LF>
105314	005015			.ASCII		<CR><LF>	
105316	005015	000		.ASCIZ		<CR><LF>	
	000001			.END			

ARGS = 000004

3638#	3675#	3707#	3714#	3721#	3750#	3757#	3764#	3773#	3806#	3843#	3875#	3882#
3889#	3918#	3925#	3932#	3941#	3974#	4011#	4044#	4051#	4058#	4088#	4095#	4102#
4111#	4144#	4181#	4213#	4220#	4227#	4255#	4262#	4269#	4304#	4341#	4373#	4380#
4387#	4418#	4429#	4492#	4529#	4561#	4568#	4575#	4604#	4611#	4618#	4627#	4660#
4697#	4729#	4736#	4743#	4771#	4778#	4785#	4820#	4857#	4889#	4896#	4903#	4934#
4945#	5008#	5045#	5077#	5084#	5091#	5119#	5126#	5133#	5167#	5204#	5236#	5243#
5250#	5278#	5285#	5292#	5326#	5363#	5395#	5402#	5409#	5437#	5444#	5451#	5485#
5524#	5557#	5564#	5571#	5599#	5606#	5613#	5650#	5688#	5695#	5702#	5731#	5738#
5745#	5754#	5795#	5833#	5840#	5847#	5876#	5883#	5890#	5899#	5940#	5978#	5985#
5992#	6021#	6028#	6035#	6044#	6073#	6121#	6128#	6135#	6157#	6203#	6210#	6217#
6252#	6290#	6297#	6304#	6313#	6344#	6351#	6358#	6399#	6437#	6444#	6451#	6460#
6491#	6498#	6505#	6546#	6584#	6591#	6598#	6607#	6638#	6645#	6652#	6692#	6742#
6749#	6756#	6804#	6811#	6818#	6827#	6867#	6918#	6932#	6941#	6972#	6986#	7020#
7058#	7065#	7072#	7102#	7113#	7124#	7158#	7196#	7203#	7210#	7240#	7251#	7262#
7297#	7344#	7351#	7358#	7387#	7402#	7417#	7437#	7455#	7501#	7548#	7555#	7562#
7591#	7602#	7620#										
1490#	8367*	8396	8442*	8443	8445*	8446*	8447	8449*	8494			
1489#	8366*	8395	8450*	8451	8453*	8493						
1367	1380											
1019#	9229	9230	9232	9964	9965	10000	10003	10495	10496	10535	10538	12456
1367	1371											
1056#	10692											
3440	3458	12456#										
1367	1374											
1367	1381											
1260#	1367	1406										
1367	1407											
1209#												
1208#												
3638#	3643	3675#	3679	3707#	3711	3714#	3718	3721#	3725	3750#	3754	3757#
3761	3764#	3768	3773#	3779	3806#	3811	3843#	3847	3875#	3879	3882#	3886
3889#	3893	3918#	3922	3925#	3929	3932#	3936	3941#	3947	3974#	3979	4011#
4015	4044#	4048	4051#	4055	4058#	4062	4088#	4092	4095#	4099	4102#	4106
4111#	4117	4144#	4149	4181#	4185	4213#	4217	4220#	4224	4227#	4231	4255#
4259	4262#	4266	4269#	4273	4304#	4309	4341#	4345	4373#	4377	4380#	4384
4387#	4391	4418#	4422	4429#	4433	4492#	4497	4529#	4533	4561#	4565	4568#
4572	4575#	4579	4604#	4608	4611#	4615	4618#	4622	4627#	4633	4660#	4665
4697#	4701	4729#	4733	4736#	4740	4743#	4747	4771#	4775	4778#	4782	4785#
4789	4820#	4825	4857#	4861	4889#	4893	4896#	4900	4903#	4907	4934#	4938
4945#	4949	5008#	5013	5045#	5049	5077#	5081	5084#	5088	5091#	5095	5119#
5123	5126#	5130	5133#	5137	5167#	5172	5204#	5208	5236#	5240	5243#	5247
5250#	5254	5278#	5282	5285#	5289	5292#	5296	5326#	5331	5363#	5367	5395#
5399	5402#	5406	5409#	5413	5437#	5441	5444#	5448	5451#	5455	5485#	5490
5524#	5528	5557#	5561	5564#	5568	5571#	5575	5599#	5603	5606#	5610	5613#
5617	5650#	5655	5688#	5692	5695#	5699	5702#	5706	5731#	5735	5738#	5742
5745#	5749	5754#	5760	5795#	5800	5833#	5837	5840#	5844	5847#	5851	5876#
5880	5883#	5887	5890#	5894	5899#	5905	5940#	5945	5978#	5982	5985#	5989
5992#	5996	6021#	6025	6028#	6032	6035#	6039	6044#	6050	6073#	6078	6121#
6125	6128#	6132	6135#	6139	6157#	6162	6203#	6207	6210#	6214	6217#	6221
6252#	6257	6290#	6294	6297#	6301	6304#	6308	6313#	6318	6344#	6348	6351#
6355	6358#	6362	6399#	6404	6437#	6441	6444#	6448	6451#	6455	6460#	6465
6491#	6495	6498#	6502	6505#	6509	6546#	6551	6584#	6588	6591#	6595	6598#
6602	6607#	6612	6638#	6642	6645#	6649	6652#	6656	6692#	6697	6742#	6746
6749#	6753	6756#	6760	6804#	6808	6811#	6815	6818#	6822	6827#	6833	6867#
6872	6918#	6922	6932#	6936	6941#	6946	6972#	6976	6986#	6990	7020#	7025
7058#	7062	7065#	7069	7072#	7076	7102#	7106	7113#	7117	7124#	7128	7158#

ASNDA 001476
ASNDC 001474
ASWREG= 000000
ATA = 100000
ATESTN= 000000
ATNMSK= 000377
ATNTBL 063776
AUNIT = 000000
AUSWR = 000000
AVECT1= 120254
AVECT2= 000000
A16 = 000400
A17 = 001000
BACK = 000001

J06

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 282
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0283

	7163	7196#	7200	7203#	7207	7210#	7214	7240#	7244	7251#	7255	7262#	7266	
	7297#	7302	7344#	7348	7351#	7355	7358#	7362	7387#	7391	7402#	7406	7417#	
	7421	7437#	7441	7455#	7459	7501#	7506	7548#	7552	7555#	7559	7562#	7566	
	7591#	7595	7602#	7606	7620#	7624								
BAOSCT	034210													
BAI	= 000010	8083#												
BB00	= 000001	1228#	9618											
BB01	= 000002	1146#												
BB02	= 000004	1145#												
BB03	= 000010	1144#												
BB04	= 000020	1143#												
BB05	= 000040	1142#												
BB06	= 000100	1141#												
BB07	= 000200	1140#												
BB08	= 000400	1139#												
BB09	= 001000	1138#												
BIT0	= 000001	1137#												
BIT00	= 000001	935#	8905											
BIT01	= 000002	925#	935	961	1010	1029	1048	1086	1105	1146	1232	1248		
BIT02	= 000004	924#	934	960	1009	1047	1085	1104	1145	1231	1247			
BIT03	= 000010	923#	933	959	1008	1046	1084	1103	1144	1230	1246			
BIT04	= 000020	922#	932	958	1007	1045	1083	1102	1143	1158	1228	1245		
BIT05	= 000040	921#	931	957	1006	1044	1081	1101	1142	1227				
BIT06	= 000100	920#	930	956	1043	1082	1100	1141	1226					
BIT07	= 000200	919#	929	1028	1042	1064	1081	1099	1140	1211	1225	1244		
BIT08	= 000400	918#	928	1027	1041	1063	1080	1098	1122	1139	1157	1210	1224	
BIT09	= 001000	917#	927	1005	1026	1040	1062	1079	1097	1138	1209	1222	11883	
BIT1	= 000002	916#	926	1004	1025	1039	1061	1078	1096	1137	1208	1221	11901	
BIT10	= 002000	934#	8373	8531	9635								11999	
		915#	1003	1024	1038	1060	1077	1095	1121	1136	1156	1207	1220	1243
BIT11	= 004000	11976												
		914#	955	1023	1037	1076	1094	1112	1120	1135	1155	1219	1242	8001
		11908												
BIT12	= 010000	913#	1022	1036	1075	1093	1119	1134	1154	1218	1241	7951		
BIT13	= 020000	912#	1021	1035	1074	1092	1111	1133	1153	1205	1217	1240	11983	
BIT14	= 040000	911#	1020	1034	1073	1091	1110	1132	1152	1163	1204	1216	1239	7916
		11869												
BIT15	= 100000	910#	1019	1033	1072	1090	1109	1131	1151	1162	1203	1215	1238	7903
		8616												
BIT2	= 000004	933#	8670											
BIT3	= 000010	932#	8905											
BIT4	= 000020	931#	8029	11077	11164									
BIT5	= 000040	930#	7979	11183										
BIT6	= 000100	929#												
BIT7	= 000200	928#	7928											
BIT8	= 000400	927#	3309	8911										
BIT9	= 001000	926#												
BOTADR	033232	7767#	7785#	7788	7803	7848#								
BOTFLG	033234	7753#	7795#	7798	7801#	7849#								
BPTVEC	= 001014	942#												
BSE	= 101000	1151#	7415	7426	8279	9522								
BUFFER	102574	8098#	8347	12456#										
BUFONE	102574	3630	3774	3798	3942	3966	4112	4136	4296	4484	4628	4652	4812	5000
		5159	5318	5477	5642	5755	5787	5900	5932	6045	6244	6391	6538	6684
		6828	6838	6859	7012	7150	7289	7306	7308#	7310#	7427	7446	7493	7510
		7512#	7514#	7611	12456#									
BUFTWO	103600	3732	3775	3900	3943	4070	4113	4396#	4443	4456	4586	4629	4912#	4958

K06

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
 DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 283
 CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0284

	4971	5713	5756	5858	5901	6003	6046	6086	6170	6323	6470	6617	6781
	6829	6951	7084	7222	7369	7428	7447	7573	7612	12456#			
CC = 004000	1135#												
CH = 002000	1136#												
CHRCNT 033235	7754#	7772*	7778*	7779	7782*	7786	7791*	7802*	7850#				
CKSWR = 104410	11867	11972	11998	12349#									
CLEAN = ***** U	7431	12456											
CLKADR 001500	1491#	8867*	8869	8873*	8875	8900							
CLKVCT 001502	1492#	8868*	8874*										
CLR = 000040	1226#	10061											
CLRSTS 046272	7938	10097#											
CLSPRN 063044	3361	3377	3398	12456#									
CMNSTA 006632	3315	3447#											
CMPBUF 036360	3773	3941	4111	4627	5754	5899	6044	6827	8596#				
CMPERR 056416	11476#												
CNSL00 063204	3325	12456#											
CNSL01 063243	3358	12456#											
CNSL02 063270	3367	12456#											
CNSL03 063351	3372	12456#											
CNSL04 063401	3383	12456#											
CNSL05 063455	3393	12456#											
CNSL06 063511	3404	12456#											
CNSL07 063536	3417	12456#											
CNTCLR 046154	3546	3590	7918	10053#									
CONT = 000100	1099#												
CR = 000015	850#	3430	7770	11555	11842	11852	12456						
CRLF = 000200	851#	3298	11813	11852									
CYLSK= 001777	1126#	8534											
DBCK = 100000	1072#												
DBEN = 040000	1073#												
DBL = 002000	1243#												
DCK = 100000	1033#	1050	8272	8604	9560	9580	11170	11173					
DDISP = 177570	857#	1343	3262										
DEBL = 020000	1074#												
DEVSEL 044502	7905	9763#											
DISPLA 001156	1343#	3262*	3270*	11923*	11975*								
DISPRE 000174	1270#	3270											
DLT = 100000	1215#	9404	11249	11252									
DMD = 000001	1086#	1105#	6876										
DPE = 000010	1158#	8273	8279	9066	9346	9842	10381	10802	11223	11226			
DPEHI = 040000	1239#												
DPELO = 020000	1240#												
DPR = 000400	1026#	10003	10200	10201	10667								
DRQ = 004000	1112#												
DRVCLR= 000010	970#	8126	10655										
DRVSTS 051510	8144	10646#											
DRY = 000200	1027#	9033	9129	9132	9133	10003	10200	10201	10667				
DSWR = 177570	856#	1342	3261										
DTASTS 052312	3714	3757	3882	3925	4051	4095	4220	4262	4380	4429	4568	4611	4736
	4778	4896	4945	5084	5126	5243	5285	5402	5444	5564	5606	5695	5738
	5840	5883	5985	6028	6128	6210	6297	6351	6444	6498	6591	6645	6749
	6811	7065	7113	7203	7251	7351	7402	7417	7437	7555	7602	8247	10777#
DTE = 010000	1036#	1050	11036	11054	11057								
DTO = 010000	1075#												
DULPRT= 024024	1115#	3605	3608										
DVA = 004000	955#	3565	8756	8829	8994	8997	9788	10106	10654	10655			

DVC = 000200	1157#	9935	10190	10442	10480	10483	10616	10989	10993	10996	11041	11415	11420
	11423												
EARLY 064322	12456#												
EAL = 020000	1092#												
ECH = 000100	1042#	1050	8272	8606	9543	9558	9576	9582	11180	12456			
ECI = 004000	1120#	8608	9578	11177									
ECRC = 001000	1096#												
EDT1 070652	1526	1533	1540	1547	1554	1561	1575	1582	1589	1596	1603	1610	1617
	1624	1631	1638	1645	1652	1659	1666	1673	1680	1687	1694	1701	1708
	1715	1722	1729	1736	1743	1750	1757	1764	1771	1778	1785	1792	1799
	1806	1813	1821	1828	1835	1842	1849	1857	1865	1873	1887	1894	1901
	1908	1915	1922	1929	1937	1944	1951	1958	1965	1972	1979	1986	1993
	2000	2007	2014	2021	2063	2070	2077	2084	2091	2098	2105	2112	2119
	2126	2133	2140	2147	2154	2161	2168	2175	2182	2189	2196	2203	2210
	2217	2224	2231	2238	2245	2252	2259	2266	2273	2280	2287	2294	2301
	2308	2315	2322	2337	2344	2351	2358	2372	2379	2386	2393	2400	2407
	2414	2421	2428	2435	2442	2449	2456	2463	2470	2477	2484	2491	2498
	2519	2526	2533	2540	2547	2687	2694	2701	2722	2729	2736	2750	2757
	2764	2771	2778	2785	2792	2799	2807	2814	2822	2829	2836	2844	2852
	2860	2869	2877	2885	2892	2899	2906	2913	2920	2927	2934	2941	2948
	2955	2962	2969	2976	2983	2990	2997	3004	3011	3018	3025	3032	3039
	3046	3053	3060	3067	3074	3081	3116	3123	3137	3144	3151	3158	3165
	3186	12456#											
EDT110 070670	2028	12456#											
EDT111 070672	2035	2042	12456#										
EDT114 070674	2056	12456#											
EDT2 070662	2505	2512	2708	2715	12456#								
EDT223 070676	2554	2743	12456#										
EDT336 070706	2330	3088	3102	3109	12456#								
EDT337 070716	3095	12456#											
EDT344 070726	3130	12456#											
EDT353 070740	3179	12456#											
ED1 100350	12456#												
ED110 100356	12456#												
ED111 100362	12456#												
ED114 100370	12456#												
ED223 100400	12456#												
ED336 100410	12456#												
ED337 100422	12456#												
ED353 100434	12456#												
EECC = 000020	1101#												
EFT1 070742	1527	1534	1541	1548	1555	1562	1576	1583	1590	1597	1604	1611	1618
	1625	1632	1639	1646	1653	1660	1667	1674	1681	1688	1695	1702	1709
	1716	1723	1730	1737	1744	1751	1758	1765	1772	1779	1786	1793	1800
	1807	1814	1822	1829	1836	1843	1850	1858	1866	1874	1888	1895	1902
	1909	1916	1923	1930	1938	1945	1952	1959	1966	1973	1980	1987	1994
	2001	2008	2015	2022	2064	2071	2078	2085	2092	2099	2106	2113	2120
	2127	2134	2141	2148	2155	2162	2169	2176	2183	2190	2197	2204	2211
	2218	2225	2232	2239	2246	2253	2260	2267	2274	2281	2288	2295	2302
	2309	2316	2323	2338	2345	2352	2359	2373	2380	2387	2394	2401	2408
	2415	2422	2429	2436	2443	2450	2457	2464	2471	2478	2485	2492	2499
	2520	2527	2534	2541	2548	2688	2695	2702	2723	2730	2737	2751	2758
	2765	2772	2779	2786	2793	2800	2808	2815	2823	2830	2837	2845	2853
	2861	2870	2878	2886	2893	2900	2907	2914	2921	2928	2935	2942	2949
	2956	2963	2970	2977	2984	2991	2998	3005	3012	3019	3026	3033	3040
	3047	3054	3061	3068	3075	3082	3117	3124	3138	3145	3152	3159	3166

EMS103	073707	124568
EMS104	073716	124568
EMS105	073745	124568
EMS106	073774	124568
EMS11	071426	124568
EMS110	074013	124568
EMS111	074042	124568
EMS112	074053	124568
EMS113	074137	124568
EMS114	074167	124568
EMS115	074213	124568
EMS116	074220	124568
EMS117	074250	124568
EMS12	071437	124568
EMS120	074272	124568
EMS121	074307	124568
EMS122	074352	124568
EMS123	074407	124568
EMS124	074452	124568
EMS125	074504	124568
EMS126	074547	124568
EMS127	074612	124568
EMS13	071500	124568
EMS130	074651	124568
EMS131	074707	124568
EMS132	074747	124568
EMS133	074773	124568
EMS134	075016	124568
EMS135	075060	124568
EMS136	075122	124568
EMS137	075147	124568
EMS14	071511	124568
EMS140	075204	124568
EMS141	075225	124568
EMS142	075253	124568
EMS143	075270	124568
EMS144	075316	124568
EMS145	075326	124568
EMS146	075355	124568
EMS147	075374	124568
EMS15	071522	124568
EMS150	075411	124568
EMS151	075430	124568
EMS152	075454	124568
EMS153	075500	124568
EMS154	075522	124568
EMS155	075536	124568
EMS156	075556	124568
EMS157	075574	124568
EMS16	071544	124568
EMS160	075611	124568
EMS161	075626	124568
EMS162	075656	124568
EMS163	075711	124568
EMS164	075735	124568
EMS165	076016	124568

EMS166	076032	12456#
EMS167	076057	12456#
EMS17	071574	12456#
EMS170	076104	12456#
EMS171	076133	12456#
EMS172	076166	12456#
EMS173	076206	12456#
EMS174	076223	12456#
EMS175	076231	12456#
EMS176	076257	12456#
EMS177	076306	12456#
EMS2	071077	12456#
EMS20	071634	12456#
EMS200	076330	12456#
EMS201	076345	12456#
EMS202	076413	12456#
EMS203	076443	12456#
EMS204	076456	12456#
EMS205	076504	12456#
EMS206	076516	12456#
EMS207	076543	12456#
EMS21	071657	12456#
EMS210	076570	12456#
EMS211	076601	12456#
EMS212	076617	12456#
EMS213	076642	12456#
EMS214	076665	12456#
EMS215	076703	12456#
EMS216	076724	12456#
EMS217	076754	12456#
EMS22	071702	12456#
EMS220	077007	12456#
EMS221	077027	12456#
EMS222	077077	12456#
EMS223	077130	12456#
EMS224	077205	12456#
EMS23	071716	12456#
EMS24	071744	12456#
EMS25	071773	12456#
EMS26	072010	12456#
EMS27	072031	12456#
EMS3	071114	12456#
EMS30	072042	12456#
EMS31	072052	12456#
EMS32	072101	12456#
EMS33	072130	12456#
EMS34	072156	12456#
EMS35	072227	12456#
EMS36	072256	12456#
EMS37	072305	12456#
EMS4	071157	12456#
EMS40	072333	12456#
EMS41	072362	12456#
EMS42	072410	12456#
EMS43	072437	12456#
EMS44	072466	12456#

EMS45	072541	12456#	
EMS46	072604	12456#	
EMS47	072633	12456#	
EMS5	071222	12456#	
EMS50	072662	12456#	
EMS51	072711	12456#	
EMS52	072737	12456#	
EMS53	072751	12456#	
EMS54	072763	12456#	
EMS55	073005	12456#	
EMS56	073034	12456#	
EMS57	073111	12456#	
EMS6	071252	12456#	
EMS60	073137	12456#	
EMS61	073152	12456#	
EMS62	073201	12456#	
EMS63	073217	12456#	
EMS64	073245	12456#	
EMS65	073263	12456#	
EMS66	073331	12456#	
EMS67	073401	12456#	
EMS7	071317	12456#	
EMS70	073426	12456#	
EMS71	073440	12456#	
EMS72	073453	12456#	
EMS73	073463	12456#	
EMS74	073500	12456#	
EMS75	073517	12456#	
EMS76	073525	12456#	
EMS77	073556	12456#	
EMTVEC=	000030	945#	3245# 3246#
EMT1	064434	1524	12456#
EMT10	064504	1573	12456#
EMT100	065502	1970	12456#
EMT101	065520	1977	12456#
EMT102	065536	1984	12456#
EMT103	065546	1991	12456#
EMT104	065560	1998	12456#
EMT105	065576	2005	12456#
EMT106	065606	2012	12456#
EMT107	065616	2019	12456#
EMT11	064510	1580	12456#
EMT110	065636	2026	12456#
EMT111	065650	2033	12456#
EMT112	065656	2040	12456#
EMT113	065664	2047	12456#
EMT114	065700	2054	12456#
EMT115	065706	2061	12456#
EMT116	065716	2068	12456#
EMT117	065726	2075	12456#
EMT12	064514	1587	12456#
EMT120	065736	2082	12456#
EMT121	065746	2089	12456#
EMT122	065756	2096	12456#
EMT123	065766	2103	12456#
EMT124	065776	2110	12456#

D07

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 289
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0290

ENT125	066006	2117	12456#
ENT126	066016	2124	12456#
ENT127	066030	2131	12456#
ENT13	064522	1594	12456#
ENT130	066042	2138	12456#
ENT131	066054	2145	12456#
ENT132	066066	2152	12456#
ENT133	066100	2159	12456#
ENT134	066112	2166	12456#
ENT135	066124	2173	12456#
ENT136	066136	2180	12456#
ENT137	066150	2187	12456#
ENT14	064534	1601	12456#
ENT140	066160	2194	12456#
ENT141	066170	2201	12456#
ENT142	066200	2208	12456#
ENT143	066210	2215	12456#
ENT144	066220	2222	12456#
ENT145	066230	2229	12456#
ENT146	066240	2236	12456#
ENT147	066250	2243	12456#
ENT15	064542	1608	12456#
ENT150	066260	2250	12456#
ENT151	066270	2257	12456#
ENT152	066302	2264	12456#
ENT153	066314	2271	12456#
ENT154	066332	2278	12456#
ENT155	066350	2285	12456#
ENT156	066362	2292	12456#
ENT157	066374	2299	12456#
ENT16	064550	1615	12456#
ENT160	066406	2306	12456#
ENT161	066416	2313	12456#
ENT162	066430	2320	12456#
ENT163	066442	12456#	
ENT164	066452	2335	12456#
ENT165	066460	2342	12456#
ENT166	066466	2349	12456#
ENT167	066474	2356	12456#
ENT17	064560	1622	12456#
ENT170	066502	12456#	
ENT171	066512	2370	12456#
ENT172	066522	2377	12456#
ENT173	066532	2384	12456#
ENT174	066542	2391	12456#
ENT175	066554	2398	12456#
ENT176	066562	2405	12456#
ENT177	066570	2412	12456#
ENT2	064440	1531	12456#
ENT20	064570	1629	12456#
ENT200	066602	2419	12456#
ENT201	066614	2426	12456#
ENT202	066626	2433	12456#
ENT203	066640	2440	12456#
ENT204	066652	2447	12456#
ENT205	066670	2454	12456#

ENT206	066700	2461	12456#
ENT207	066710	2468	12456#
ENT21	064600	1636	12456#
ENT210	066722	2475	12456#
ENT211	066730	2482	12456#
ENT212	066744	2489	12456#
ENT213	066760	2496	12456#
ENT214	066770	2503	12456#
ENT215	067010	2510	12456#
ENT216	067022	2517	12456#
ENT217	067032	2524	12456#
ENT22	064610	1643	12456#
ENT220	067042	2531	12456#
ENT221	067052	2538	12456#
ENT222	067062	2545	12456#
ENT223	067072	2552	12456#
ENT224	067102	12456#	
ENT225	067104	12456#	
ENT226	067106	12456#	
ENT227	067110	12456#	
ENT23	064620	1650	12456#
ENT230	067112	12456#	
ENT231	067114	12456#	
ENT232	067116	12456#	
ENT233	067120	12456#	
ENT234	067122	12456#	
ENT235	067124	12456#	
ENT236	067126	12456#	
ENT237	067130	12456#	
ENT24	064630	1657	12456#
ENT240	067132	12456#	
ENT241	067134	12456#	
ENT242	067136	12456#	
ENT243	067140	12456#	
ENT244	067142	12456#	
ENT245	067144	12456#	
ENT246	067146	2685	12456#
ENT247	067156	2692	12456#
ENT25	064640	1664	12456#
ENT250	067170	2699	12456#
ENT251	067204	2706	12456#
ENT252	067212	2713	12456#
ENT253	067220	2720	12456#
ENT254	067236	2727	12456#
ENT255	067246	2734	12456#
ENT256	067256	2741	12456#
ENT257	067266	2748	12456#
ENT26	064650	1671	12456#
ENT260	067300	2755	12456#
ENT261	067312	2762	12456#
ENT262	067332	2769	12456#
ENT263	067342	2776	12456#
ENT264	067352	2783	12456#
ENT265	067372	2790	12456#
ENT266	067412	2797	12456#
ENT267	067424	2805	12456#

EMT27	064660	1678	12456#
EMT270	067442	2812	12456#
EMT271	067456	2820	12456#
EMT272	067474	2827	12456#
EMT273	067512	2834	12456#
EMT274	067530	2842	12456#
EMT275	067546	2850	12456#
EMT276	067560	2858	12456#
EMT277	067574	2867	12456#
EMT3	064446	1538	12456#
EMT30	064670	1685	12456#
EMT300	067612	2875	12456#
EMT301	067632	2883	12456#
EMT302	067654	2890	12456#
EMT303	067666	2897	12456#
EMT304	067700	2904	12456#
EMT305	067712	2911	12456#
EMT306	067724	2918	12456#
EMT307	067734	2925	12456#
EMT31	064700	1692	12456#
EMT310	067754	2932	12456#
EMT311	067766	2939	12456#
EMT312	070000	2946	12456#
EMT313	070012	2953	12456#
EMT314	070032	2960	12456#
EMT315	070044	2967	12456#
EMT316	070056	2974	12456#
EMT317	070070	2981	12456#
EMT32	064710	1699	12456#
EMT320	070100	2988	12456#
EMT321	070110	2995	12456#
EMT322	070120	3002	12456#
EMT323	070126	3009	12456#
EMT324	070136	3016	12456#
EMT325	070150	3023	12456#
EMT326	070162	3030	12456#
EMT327	070200	3037	12456#
EMT33	064720	1706	12456#
EMT330	070212	3044	12456#
EMT331	070222	3051	12456#
EMT332	070234	3058	12456#
EMT333	070246	3065	12456#
EMT334	070264	3072	12456#
EMT335	070302	3079	12456#
EMT336	070322	2328	3086
EMT337	070332	3093	12456#
EMT34	064730	1713	12456#
EMT340	070342	3100	12456#
EMT341	070354	3107	12456#
EMT342	070362	3114	12456#
EMT343	070374	3121	12456#
EMT344	070406	3128	12456#
EMT345	070420	3135	12456#
EMT346	070430	3142	12456#
EMT347	070444	3149	12456#
EMT35	064740	1720	12456#

12456#

H07

DZRM0A - RMO3 FUNCTIONAL TEST, PART 2
 DZRM0A.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 293
 CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0294

ERTY03 033263
 ERTY04 033271
 ESRC = 004000
 FER = 000020

 FIND = 000001

	7743	7857												
	7839	7859												
	1094													
	1044	1050	7111	7121	7249	7259	7400	7410	8272	9507	11105	11122	11125	
	11148	11151												
	3638	3640	3675	3676	3707	3708	3714	3715	3721	3722	3750	3751	3757	
	3758	3764	3765	3773	3776	3806	3808	3843	3844	3875	3876	3882	3883	
	3889	3890	3918	3919	3925	3926	3932	3933	3941	3944	3974	3976	4011	
	4012	4044	4045	4051	4052	4058	4059	4088	4089	4095	4096	4102	4103	
	4111	4114	4144	4146	4181	4182	4213	4214	4220	4221	4227	4228	4255	
	4256	4262	4263	4269	4270	4304	4306	4341	4342	4373	4374	4380	4381	
	4387	4388	4418	4419	4429	4430	4492	4494	4529	4530	4561	4562	4568	
	4569	4575	4576	4604	4605	4611	4612	4618	4619	4627	4630	4660	4662	
	4697	4698	4729	4730	4736	4737	4743	4744	4771	4772	4778	4779	4785	
	4786	4820	4822	4857	4858	4889	4890	4896	4897	4903	4904	4934	4935	
	4945	4946	5008	5010	5045	5046	5077	5078	5084	5085	5091	5092	5119	
	5120	5126	5127	5133	5134	5167	5169	5204	5205	5236	5237	5243	5244	
	5250	5251	5278	5279	5285	5286	5292	5293	5326	5328	5363	5364	5395	
	5396	5402	5403	5409	5410	5437	5438	5444	5445	5451	5452	5485	5487	
	5524	5525	5557	5558	5564	5565	5571	5572	5599	5600	5606	5607	5613	
	5614	5650	5652	5688	5689	5695	5696	5702	5703	5731	5732	5738	5739	
	5745	5746	5754	5757	5795	5797	5833	5834	5840	5841	5847	5848	5876	
	5877	5883	5884	5890	5891	5899	5902	5940	5942	5978	5979	5985	5986	
	5992	5993	6021	6022	6028	6029	6035	6036	6044	6047	6073	6075	6121	
	6122	6128	6129	6135	6136	6157	6159	6203	6204	6210	6211	6217	6218	
	6252	6254	6290	6291	6297	6298	6304	6305	6313	6315	6344	6345	6351	
	6352	6358	6359	6399	6401	6437	6438	6444	6445	6451	6452	6460	6462	
	6491	6492	6498	6499	6505	6506	6546	6548	6584	6585	6591	6592	6598	
	6599	6607	6609	6638	6639	6645	6646	6652	6653	6692	6694	6742	6743	
	6749	6750	6756	6757	6804	6805	6811	6812	6818	6819	6827	6830	6867	
	6869	6918	6919	6932	6933	6941	6943	6972	6973	6986	6987	7020	7022	
	7058	7059	7065	7066	7072	7073	7102	7103	7113	7114	7124	7125	7158	
	7160	7196	7197	7203	7204	7210	7211	7240	7241	7251	7252	7262	7263	
	7297	7299	7344	7345	7351	7352	7358	7359	7387	7388	7402	7403	7417	
	7418	7437	7438	7455	7456	7501	7503	7548	7549	7555	7556	7562	7563	
	7591	7592	7602	7603	7620	7621								
	1119	3796	3964	4134	4294	4482	4650	4810	4998	5157	5316	5475	5640	
	5785	5930	6084	6168	6389	6536	6682	6857	7082	7148	7287	7396	7491	
	8108	8537	8539	8610	9867	10943								
	9221	12456												
	962	10654	11098											
	960	9120												
	959	9120												
	958	9120	9694											
	957	9120												
	956	9120												
	3634	3802	3970	4140	4300	4488	4656	4816	5004	5163	5322	5481	5646	
	5791	5936	6248	6395	6542	6688	6863	7016	7154	7293	7497	8521		
	3599	3667	3699	3742	3835	3867	3910	4003	4036	4080	4173	4205	4247	
	4333	4365	4410	4450	4521	4553	4596	4689	4721	4763	4849	4881	4926	
	4965	5037	5069	5111	5196	5228	5270	5355	5387	5429	5516	5549	5591	
	5680	5723	5825	5868	5970	6013	6113	6195	6282	6336	6429	6483	6576	
	6630	6734	6796	6910	6964	7050	7094	7188	7232	7336	7379	7540	7583	
	7931	7955	7982	8005	8031	8135	8168	8204	8238	8738				
	1426	8750												
	1497	3596*	3597*	4448*	4449*	4963*	4964*	7953*	7954*	8003*	8004*	8696	8751	

FMT16 = 010000

FNCOTB 063676
 FNCMSK= 000077
 FO = 000002
 F1 = 000004
 F2 = 000010
 F3 = 000020
 F4 = 000040
 GENBUF 036114

GET 037012

GETBUF 001326
 GETINX 001504

SINTAG	001151	1340#	12114#	12133	12153	12266												
SITENB	001130	1330#	7735	11982#	11990	12011												
SLF	001220	1362#	11852	12011	12249	12259												
SLFLG	062731	12442#	12448#															
SLPADR	001122	1327#	3255#	3498#	11894#	11903#	11919#	11924	11926									
SLPERR	001124	1328#	3256#	3499#	11903	11920#	11926	12001										
SMAOR1	001254	1395#																
SMAOR2	001260	1399#																
SMAOR3	001264	1402#																
SMAOR4	001270	1405#																
SMAIL	001222	1309	1313	1368#	3273	3288	3477	3504	3544	3588	3622	3790	3958	4128				
		4288	4476	4644	4804	4992	5151	5310	5469	5634	5779	5924	6069	6154				
		6236	6383	6530	6676	6851	7004	7142	7280	7484	11796	11918	11988					
SMAMS1	001252	1389#																
SMAMS2	001256	1397#																
SMAMS3	001262	1400#																
SMAMS4	001266	1403#																
SMBADR	001102	1309#																
SMFLG	062730	12402#	12408#	12443#	12447#													
SMNEW	062066	12120	12264#															
SMSCAD	001236	1375#	12418#	12421														
SMGLG	001240	1376#	12429#															
SMSTY	001222	1369#	12416	12424#	12436	12440#												
SMSTR	062055	12117	12262#															
SMTYP1	001253	1390#																
SMTYP2	001257	1398#																
SMTYP3	001263	1401#																
SMTYP4	001267	1404#																
SMXCNT	060346	11916	11926#															
SNLL	001170	1348#	11823	11852														
SNWTST=	000001	3493#	3495	3534#	3536	3578#	3580	3613#	3781#	3949#	4119#	4279#	4467#	4635#				
		4795#	4983#	5142#	5301#	5460#	5625#	5770#	5915#	6060#	6145#	6227#	6374#	6521#				
		6667#	6842#	6995#	7133#	7271#	7475#											
SOCNT	057544	11728#	11757#	11770#														
SOMODE	057546	11723#	11727#	11732	11735#	11746#	11772#											
SOVER	060332	11870	11895	11904	11914	11923#												
SPASS	001230	1372#	3273#	7662#	7663#	7674	7696	11910	11927									
SPASTM	001106	1311#																
SPOWER	062456	12390	12395#															
SPWRDN	062310	3249	12359#	12387														
SPWRMG	062444	12390#																
SPWRUP	062362	12369	12375#															
SQUES	001216	1360#	11852	12011	12171	12242	12259											
SRDCHR	061502	12184#	12350															
SRDDEC=	##### U	12353																
SRDLIN	061572	12207#	12351															
SRDOCT	062100	12277#	12352															
SRDSZ =	000010	12200#																
SRESRE	056764	11593#	12354															
SRTNAD	032464	7695#																
SR2A =	##### U	12355																
SSAVRE	056726	11577#	12353															
SSAVR6	062454	12368#	12376	12377#	12378#	12394#												
SSCOPE	060032	3243	11866#															
SSETUP=	000137	1262#	3242	3243	3245	3247	3249	3251	3252	3253	3255	3282	3285	3474				
		7660	11867	11972	11998	12006	12054	12059	12060	12090	12266							

K08

DZRMDA - RMO3 FUNCTIONAL TEST, PART 2
DZRMDA.P11 29-JUL-77 14:07

MACY11 30(1046) 29-JUL-77 15:01 PAGE 310
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0310

	1942	1949	1956	1963	1970	1977	1984	1991	1998	2005	2012	2019	2026	2033	2040
	2047	2054	2061	2068	2075	2082	2089	2096	2103	2110	2117	2124	2131	2138	2145
	2152	2159	2166	2173	2180	2187	2194	2201	2208	2215	2222	2229	2236	2243	2250
	2257	2264	2271	2278	2285	2292	2299	2306	2313	2320	2328	2335	2342	2349	2356
	2363	2370	2377	2384	2391	2398	2405	2412	2419	2426	2433	2440	2447	2454	2461
	2468	2475	2482	2489	2496	2503	2510	2517	2524	2531	2538	2545	2552	2559	2566
	2573	2580	2587	2594	2601	2608	2615	2622	2629	2636	2643	2650	2657	2664	2671
	2678	2685	2692	2699	2706	2713	2720	2727	2734	2741	2748	2755	2762	2769	2776
	2783	2790	2797	2805	2812	2820	2827	2834	2842	2850	2858	2867	2875	2883	2890
	2897	2904	2911	2918	2925	2932	2939	2946	2953	2960	2967	2974	2981	2988	2995
	3002	3009	3016	3023	3030	3037	3044	3051	3058	3065	3072	3079	3086	3093	3100
	3107	3114	3121	3128	3135	3142	3149	3156	3163	3170	3177	3184			
SSCMRE	1315#														
SSCMTM	1315#	1352	1353	1354	1355	1356									
SSESCA	950#														
SSNEWT	950#	3493	3534	3578	3613	3781	3949	4119	4279	4467	4635	4795	4983	5142	5301
	5460	5625	5770	5915	6060	6145	6227	6374	6521	6667	6842	6995	7133	7271	7475
SSSET	12332#	12341	12342	12343	12344	12345	12347	12349	12350	12351	12352	12353	12354		
SSSETH	3273#														
SSSKIP	950#														
.EQUAT	808#	840													
.HEADE	808#	809													
.SETUP	808#	1262													
.SWRHI	808#	820													
.SWRLO	808#	832#													
.SACT1	808#	1274													
.SAPT8	808#	1364#													
.SAPTH	808#	1292													
.SAPTY	808#	12398													
.SCATC	808#	1263													
.SCMTA	808#	1315													
.SDIV	808#														
.SEOP	808#	7649													
.SERRO	808#	11957													
.SERRT	808#														
.SMULT	808#														
.SPOWE	808#	12355													
.SRAND	808#														
.SRDDE	808#														
.SRDOC	808#	12267													
.SREAD	808#	12011													
.SSAVE	808#	11560													
.SSCOP	808#	11852													
.SSIZE	808#														
.STRAP	808#	12305													
.STYP8	808#	11605													
.STYPD	808#	11629													
.STYPE	808#	11773													
.STYP0	808#	11696													

. ABS. 105321 000

ERRORS DETECTED: 0

L08

DZRMDA - RMD3 FUNCTIONAL TEST, PART 2 MACY11 30(1046) 29-JUL-77 15:01 PAGE 311
DZRMDA.P11 29-JUL-77 14:07 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0311

DSKZ:DZRMDA.BIN, DSKZ:DZRMDA.SEQ/DOC/SOL/CRF=DSKM:DZRMDA.P11
RUN-TIME: 64 60 4 SECONDS
RUN-TIME RATIO: 1299/129=10.0
CORE USED: 33K (65 PAGES)

DOCUMENT PAGES: 311

EOF1DZRMDASEQ

00010000

770804

PDP10 411