

RMBO

FCTNL TEST PT 2
CZRNEAO

AH-T113A-MC
FICHE 1 OF 2

JUL 1982
COPYRIGHT © 1982
MADE IN USA



A large grid of approximately 15 columns and 25 rows of data. Each cell contains a small table or set of data points, likely representing test results or performance metrics. The text is very small and difficult to read, but the layout is consistent across the entire page.

RM80

FCTNL TEST PT 2
CZRNEAO

AH-T113A-MC
FICHE 2 OF 2

JUL 1982
COPYRIGHT © 1982
MADE IN USA



A grid of approximately 15 columns and 15 rows of data. Each cell contains a small table or list of values, likely representing test results or system parameters. The data is organized in a structured, tabular format.



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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-T112A-MC
PRODUCT NAME: CZRNEAO RM80 FCTNL PT2
PRODUCT DATE: APRIL 1, 1982
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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 UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 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
 - 7. EXECUTION TIME
- 5. ENVIRONMENTAL SUPPORT
 - 1. PROCESSOR COMPATIBILITY
 - 2. DUAL PORT CONFIGURATIONS
 - 3. MEMORY PARITY HARDWARE
 - 4. MEMORY MANAGEMENT HARDWARE
 - 5. ACT, APT COMPATIBILITY
 - 6. XXDP COMPATIBILITY
 - 7. OPERATING SYSTEM COMPATIBILITY
- 6. TEST DESCRIPTION

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM80 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RM80 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 RM80 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

THE TEST IS COMPRISED OF 4 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; PART 4 TESTS MECHANICAL POSITIONING OPERATIONS AND VARIOUS TIMING PARAMETERS OF THE RM80 DISK DRIVE.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM80 DISK SUBSYSTEM WHICH CONSISTS OF THE RH70 MASSBUS CONTROLLER, THE RM80 MASSBUS ADAPTER, AND THE STORAGE MODULE DISK DRIVE. NOTE THAT A DISK 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 RM80 SUBSYSTEM FUNCTIONAL TEST:

PDP-11 PROCESSOR
20K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH70 CONTROLLER
1 TO 8 RM80 DISK DRIVES

2.2 MEDIA REQUIREMENTS

58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

EACH UNIT BEING TESTED MUST BE LOADED WITH A DISK BEFORE TESTING BEGINS ON THAT UNIT. THE DRIVE MUST HAVE A FORMATTED DISK AND CONTAIN A READABLE COPY OF THE MFG(DEC144), USR(DEC144) AND SSF BAD SECTOR FILES.

NOTE: TO ENSURE THAT ALL CUSTOMER DATA IS PRESERVED ON THE DISK, ALL DATA TRANSFERS WILL BE PERFORMED ON THE FE CYLINDERS ONLY.

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RM80 DISKLESS TEST, PART 1 & 2

RM80 FUNCTIONAL TEST, PART 1

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.

3.2 SWITCH OPTIONS

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

SW15 HALT ON ERROR
SW14 LOOP ON TEST (CURRENTLY BEING EXECUTED)
SW13 INHIBIT ERROR TYPEOUTS
SW12 UNUSED
SW11 INHIBIT TEST ITERATIONS
SW10 BELL ON ERROR
SW09 LOOP ON ERROR
SW08 LOOP ON TEST IN SW07-00

THE LOW ORDER 8 SWITCHES (SW07-SW00), ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY THE OCTAL NUMBER OF THE TEST WHICH THE PROGRAM WILL LOOP ON.

3.3 STARTING

THE PROGRAM MAY BE STARTED AT LOCATION 200 OR 204. STARTING AT 200 WILL BE THE NORMAL STARTING ADDRESS. STARTING AT 204 WILL ENABLE THE RH/RM BASE ADDRESS TO BE CHANGED. IF RUNNING IN A STAND-ALONE ENVIRONMENT, THE PROGRAM USES CONSOLE DIALOGUE TO ALLOW THE OPERATOR

115 TO CONTROL TEST CONDITIONS.

116
117
118
119 3.4 HALTING

120 THE PROGRAM SHOULD BE HALTED BY TYPING CONTROL C FROM THE CONSOLE.

121 NOTE: IF THE PROGRAM IS HALTED BY ANY OTHER MEANS, BAD HEADER
122 INFORMATION MAY BE LEFT ON THE DISK. THIS OF COURSE, DEPENDS
123 ON WHICH TEST WAS BEING PERFORMED AT THE TIME THE PROGRAM
124 WAS HALTED. ALSO, THIS ONLY APPLIES TO A NORMAL (ERROR FREE)
125 OPERATION OF THE PROGRAM.
126
127

128
129
130 3.5 RESTARTING

131 THE PROGRAM CAN BE RESTARTED AT ADDRESS 200 OR 204. (SEE SECTION 3.3)
132
133

134
135 4.0 OPERATOR INTERFACE

136 4.1 PROGRAM ID

137 THE PROGRAM TYPES ITS TITLE AND MAINDEC NUMBER THE FIRST TIME IT
138 IS STARTED AFTER BEING LOADED. ALSO, A WARNING MESSAGE IS TYPED,
139 NOTIFYING THE OPERATOR OF POSSIBLE HEADER CORRUPTION IF THE PROGRAM IS
140 HALTED IMPROPERLY. THE PROGRAM IDENTIFICATION AND THE WARNING DO NOT
141 OCCUR IF THE PROGRAM IS RESTARTED.
142
143

144
145
146 4.2 CONSOLE DIALOGUE

147 WHEN THE PROGRAM IS RUNNING IN STAND ALONE MODE, IT ENTERS A
148 CONSOLE DIALOGUE SEQUENCE AFTER TYPING THE PROGRAM I.D. AND WARNING
149 MESSAGE. (SEE SECTION 4.1)
150

151 THE FIRST QUESTION TYPED OUT IS: "TYPE HELP TEXT (L) N ?".
152 IF THE OPERATOR RESPONDS WITH A "Y", THE PROGRAM WILL TYPE A BRIEF
153 HELP MESSAGE WHICH WILL LIST SWITCH OPTIONS, ETC. ANY OTHER RESPONSE
154 TO THE QUESTION IS CONSIDERED A "N" AND NO HELP TEXT IS TYPED. THIS
155 QUESTION IS ONLY ASKED ON THE INITIAL PROGRAM START AND NOT ON
156 SUBSEQUENT START-UP'S.
157

158 ON THE PROGRAM INITIAL START AND WHEN RESTARTING AT LOCATION 204,
159 THE OPERATOR MAY CHANGE THE RH/RM BASE ADDRESSES WITH THE FOLLOWING
160 DIALOGUE.
161

162 EXAMPLE 1

163 RMCS1=176700 <CR> ;NO CHANGE IN ADDRESS
164 RMVEC=000254 <CR> ;NO CHANGE IN ADDRESS
165

166 EXAMPLE 2
167
168
169
170
171

172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228

RMCS1=176700 177200<CR> ;CHANGE BASE ADDRESS TO 177200
RMVEC=000254 260<CR> ;CHANGE VECTOR ADDRESS TO 260

ON THE INITIAL START, THE NEXT QUESTION TYPED IS, 'TYPE "A" TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S) AND TERMINATE INPUT WITH A CARRIAGE RETURN'. THEN, 'DRIVE(S):' IS TYPED AND WAITS FOR THE OPERATOR TO TYPE AN "A" TO TEST ALL POSSIBLE DRIVES OR TYPE ANY STRING OF DRIVE NUMBER(S) TO BE TESTED AND TERMINATE THE INPUT WITH A "CARRIAGE RETURN". NO COMMAS OR ANY OTHER SEPARATORS ARE NEEDED WHEN ENTERING THE DRIVE NUMBERS AS A STRING. THE PROGRAM ENTERS THE COMMA SEPARATOR AUTOMATICALLY AFTER TYPING EACH NUMBER. ON ALL SUBSEQUENT STARTS, ONLY THE 'DRIVE(S):' PROMPT IS TYPED.

THE DIAGNOSTIC THEN INITIALIZES AND REPORTS THE STATUS OF THE DRIVES WHICH WERE PREVIOUSLY SPECIFIED FOR TESTING. THE FOLLOWING IS AN EXAMPLE PRINTOUT:

```
'UNIT STATUS:
 0 ONLINE RM80
 1 LOAD DEVICE
 2 OFFLINE RM80
 3 NOT PRESENT
 4 NOT PRESENT
 5 NOT AN RM80
 6 NOT PRESENT
 7 NOT PRESENT'
```

THE ABOVE UNIT STATUS SHOWS THAT DRIVE 0 WILL BE TESTED, WHILE DRIVES 1 - 7 WILL NOT BE TESTED.

THE DIAGNOSTIC THEN TYPES THE FOLLOWING MESSAGE, BASED ON THE STATUS OF THE DRIVE:

'DRIVE(S) TO BE TESTED, 0'

IF NO DRIVES ARE AVAILABLE FOR TESTING, THE FOLLOWING MESSAGE WILL BE TYPED TO THE OPERATOR:

'DRIVE(S) TO BE TESTED, NONE'

THE PROGRAM WILL THEN, EITHER START TESTING THE DRIVES AVAILABLE FOR TESTING OR RETURN TO THE BEGINNING OF THE PROGRAM AND WAIT.

ONCE THE DRIVES START TESTING, THE FOLLOWING MESSAGE WILL OCCUR AS EACH DRIVE BEGINS TO BE TESTED:

'DRIVE 0'

AFTER ALL THE DRIVES ARE COMPLETELY TESTED, THE END OF PASS MESSAGE WILL BE TYPED (SEE SECTION 4.3) AND THE PROGRAM WILL START TESTING ALL THE DRIVES AGAIN. THIS WILL CONTINUE UNTIL THE PROGRAM IS HALTED BY THE OPERATOR.

NOTE: THE LETTER LOCATED WITHIN THE BRACKETS () INDICATES THE TYPE OF RESPONSE REQUIRED BY THE USER, D=DECIMAL, O=OCTAL AND L=LETTER.

229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285

4.3 PROGRESS REPORTS

AN END OF PASS REPORT OCCURS EACH TIME THE PROGRAM IS EXECUTED FOR ALL DEVICES IN THE TEST QUE. THE END OF PASS REPORT IS AS FOLLOWS.

'END OF PASS 1'

THE FOLLOWING MESSAGE WILL ALSO OCCUR IF THERE WERE ERRORS SINCE THE LAST END OF PASS REPORT.

'TOTAL ERRORS SINCE LAST REPORT 0'

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 (DRIVE) BEING TESTED, THE DRIVE TYPE, 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.

THE FOLLOWING PRINTOUT SHOWS A TYPICAL ERROR MESSAGE FOR THIS PROGRAM:

```

DRV# 0 - RM80, TEST# 14, ERR# 326, PC=016654
MASSBUS DATA BUS PARITY ERROR "MDPE" (RMCS2, BIT 8) DETECTED
DURING WRITE COMMAND
EXPECTD  RECEVD
040300    040700
RMCS1     RMCS2   RMDS   RMER1   RMER2   RMAS
144252    040700  010700  000000  000000  000000
RMWC      RMBA    RMDA   RMOF   RMDC   RMEC1   RMEC2
177403    104604  000002  010000  000000  004066  000000
RMMR1     RMMR2   RMDT   RMSN
000010    011777  024026  177777

```

286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342

4.7 EXECUTION TIME

PASS 1 OF THE PROGRAM TAKES ABOUT 20 SECONDS. PASS 2 AND SUBSEQUENT PASSES TAKE 55 SECONDS.

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM80 SUBSYSTEM FUNCTIONAL TEST IS EXECUTABLE ON A PDP-11/70 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 RM80 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RM30 ADAPTER BUT IS EXECUTABLE ON RM80 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 RM80 SUBSYSTEM FUNCTIONAL TEST.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE IS NOT USED DURING THE RM80 SUBSYSTEM FUNCTIONAL TEST.

5.5 ACT11, APT11 COMPATIBILITY

THE RM80 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 RM80 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RM80 IS THE XXDP LOADING DEVICE.

343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

6.0 TEST DESCRIPTION

TEST 1 CONTROLLER ACCESS TEST

PURPOSE:

TO VERIFY THAT THE UNIBUS ADDRESS OF THE RM80 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.

TEST 2 - 30 WRITE/READ HEADER AND DATA (FORMAT) TESTS

PURPOSE:

TO TEST WRITE HEADER AND DATA AND READ HEADER AND DATA FUNCTIONALITY OF THE RM80 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

400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456

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 THE SAME TYPE OF ERRORS AS IN THE WRITE COMMAND.

NOTE: 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.

TEST 2 FORMAT ZEROS TEST

THE TEST SEEKS TO CYLINDER 559., THEN WRITES HEADER AND DATA ON SECTOR 1 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. THE TEST IS REPEATED FOR 16 BIT FORMAT.

TEST 3 ZERO FILL TEST TEST

THE TEST EXECUTES A SEEK TO CYLINDER 559. 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 RH70 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.

TEST 4 FORMAT CHECK ZEROS TEST

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.

TEST 5 FORMAT CHECK ZEROS W/ WCE ERROR TEST

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

457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513

THE CORRECT WRITE CHECK ERROR IS DETECTED.

TEST 6 FORMAT ONES TEST

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.

TEST 7 FORMAT CHECK ONES TEST

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

TEST 10 FORMAT CHECK ONES W/ WCE ERROR TEST

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.

TEST 11 FORMAT MULTIPLE SECTORS TEST

THE TEST SEEKS TO CYLINDER 559. 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 SAME SECTORS ARE VERIFIED WITH A WRITE CHECK HEADER AND DATA COMMAND. THE TEST IS REPEATED FOR 16 BIT FORMAT WITH THE SSEI BIT SET.

TEST 12 READ HEADER & DATA W/ HEAD SWITCHING TEST

THE TEST SEEKS TO CYLINDER 559. TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE READ HEADER AND DATA COMMAND STARTS WITH CYLINDER 559., TRACK 2, SECTOR 30. THE WORD COUNT IS EQUAL TO MULTIPLE SECTORS WHICH CAUSES THE SUBSYSTEM TO SWITCH

514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570

FROM TRACK 2 TO TRACK 3 AFTER THE FIRST SECTOR IS READ. IF THE 'SSE' BIT IS SET AFTER EXECUTING A DATA COMMAND, THE 16 BIT MODE WILL BE TERMINATED AND THE TEST IS REPEATED FOR 16 BIT FORMAT WITH THE SSEI BIT SET, WHERE SECTOR 31. IS THE LAST SECTOR. IF NO 'SSE' WAS SET, THEN BOTH FORMAT MODES WILL BE EXECUTED.

TEST 13 READ HEADER & DATA W/ MID-TRANSFER SEEK TEST

THIS TEST READS MULTIPLE SECTORS STARTING WITH CYLINDER 559., LAST TRACK AND SECTOR 30., CAUSING A MID-TRANSFER SEEK AFTER THE FIRST OF THE MULTIPLE SECTORS IS READ. IF THE 'SSE' BIT IS SET AFTER EXECUTING A DATA COMMAND, THE 16 BIT MODE WILL BE TERMINATED AND THE TEST IS REPEATED FOR 16 BIT FORMAT WITH THE SSEI BIT SET, WHERE SECTOR 31. IS THE LAST SECTOR. IF NO 'SSE' WAS SET, THEN BOTH FORMAT MODES WILL BE EXECUTED.

TEST 14 FORMAT W/ IMPLIED SEEK TEST

THIS TEST SEEKS TO THE FIRST CYLINDER PRIOR TO WRITING HEADER AND DATA ON CYLINDER 559. 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.

TEST 15 FORMAT EACH SECTOR ADDRESS TEST

HEADERS AND DATA OF EACH SECTOR ON CYLINDER 559., TRACK 2 ARE FORMATTED AND READ WITH THE PROGRAM VERIFYING HEADERS AND DATA.

TEST 16 FORMAT EACH TRACK ADDRESS TEST

THIS TEST FORMATS SECTOR 0 OF EACH TRACK ON CYLINDER 560. AND READS EACH SECTOR WITH THE PROGRAM VERIFYING HEADERS AND DATA.

571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627

TEST 17 READ HEADER & DATA IN LAST SECTOR TEST

THIS TEST EXECUTES A READS HEADER AND DATA COMMAND ON THE LAST SECTOR (30.) OF THE DISK, IN 16 BIT FORMAT MODE AND VERIFIES THAT LBT STATUS SETS. IF THE 'SSE' BIT IS SET AFTER EXECUTING A DATA COMMAND, THE 16 BIT MODE WILL BE TERMINATED AND THE TEST IS REPEATED FOR 16 BIT FORMAT WITH THE SSEI BIT SET, WHERE SECTOR 31. IS THE LAST SECTOR. IF NO 'SSE' WAS SET, THEN BOTH FORMAT MODES WILL BE EXECUTED.

TEST 20 READ HEADER & DATA W/ AOE ERROR TEST

THIS TEST READS MULTIPLE SECTORS IN 16 BIT FORMAT, STARTING WITH THE LAST SECTOR 30. AND VERIFIES THAT AOE STATUS SETS. IF THE 'SSE' BIT IS SET AFTER EXECUTING A DATA COMMAND, THE 16 BIT MODE WILL BE TERMINATED AND THE TEST IS REPEATED FOR 16 BIT FORMAT WITH THE SSEI BIT SET, WHERE SECTOR 31. IS THE LAST SECTOR. IF NO 'SSE' WAS SET, THEN BOTH FORMAT MODES WILL BE EXECUTED.

TEST 21 READ INVALID SECTOR ADDRESS TEST

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

TEST 22 READ INVALID TRACK ADDRESS TEST

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

TEST 23 READ INVALID CYLINDER ADDRESS TEST

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

TEST 24 FORMAT AT OFFSET TEST

628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683

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

TEST 25 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.

TEST 26 FORMAT ERROR TEST

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.

TEST 27 & 30 FORMAT HCE TEST (1ST AND 2ND 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.

TEST 31 FORMAT FE CYLINDERS

THIS TEST FORMATS BOTH FE CYLINDERS TO RE-ESTABLISH THE CORRECT FORMAT ON THE SECTORS THAT WERE WRITTEN UPON BY PREVIOUS TESTS. THE TEST IS DONE IN 16 BIT FORMAT WITH THE SSEI BIT SET TO ALLOW ALL 32 SECTORS ON EACH TRACK TO BE FORMATTED.

1
487
488

:*LAST REVISION 19-OCT-81

.TITLE CZRNEAO RM80 FCTNL PT2

:*COPYRIGHT (C) 1982

:*DIGITAL EQUIPMENT CORPORATION

:*COLORADO SPGS., CO. 80919

:*

:*PROGRAM BY MIKE LEAVITT

:*

:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC

:*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

:*

489

.SBTTL OPERATIONAL SWITCH SETTINGS

:*

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	UNUSED
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<7:0>
7	TN128
6	TN64
5	TN32
4	TN16
3	TN8
2	TN4
1	TN2
0	TN1

490

491
492

.SBTTL BASIC DEFINITIONS

:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

001100
104000
000004

STACK = 1100

ERROR = EMT

SCOPE = IOT

::BASIC DEFINITION OF ERROR CALL

::BASIC DEFINITION OF SCOPE CALL

:*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

HT = 11

LF = 12

CR = 15

CRLF = 200

PS = 177776

PSW=PS

STKLMT = 177774

PIRQ = 177772

DSWR = 177570

DDISP = 177570

::CODE FOR HORIZONTAL TAB

::CODE FOR LINE FEED

::CODE FOR CARRIAGE RETURN

::CODE FOR CARRIAGE RETURN-LINE FEED

::PROCESSOR STATUS WORD

::STACK LIMIT REGISTER

::PROGRAM INTERRUPT REQUEST REGISTER

::HARDWARE SWITCH REGISTER

::HARDWARE DISPLAY REGISTER

:*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003

R0 = X0

R1 = X1

R2 = X2

R3 = X3

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

000004	R4	=	%4	::GENERAL REGISTER
000005	R5	=	%5	::GENERAL REGISTER
000006	R6	=	%6	::GENERAL REGISTER
000007	R7	=	%7	::GENERAL REGISTER
000006	SP	=	%6	::STACK POINTER
000007	PC	=	%7	::PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS

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

;*''SWITCH REGISTER'' SWITCH DEFINITIONS

100000	SW15	=	100000	
040000	SW14	=	40000	
020000	SW13	=	20000	
010000	SW12	=	10000	
004000	SW11	=	4000	
002000	SW10	=	2000	
001000	SW09	=	1000	
000400	SW08	=	400	
000200	SW07	=	200	
000100	SW06	=	100	
000040	SW05	=	40	
000020	SW04	=	20	
000010	SW03	=	10	
000004	SW02	=	4	
000002	SW01	=	2	
000001	SW00	=	1	
001000	SW9=SW09			
000400	SW8=SW08			
000200	SW7=SW07			
000100	SW6=SW06			
000040	SW5=SW05			
000020	SW4=SW04			
000010	SW3=SW03			
000004	SW2=SW02			
000002	SW1=SW01			
000001	SW0=SW00			

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

100000	BIT15	=	100000	
040000	BIT14	=	40000	
020000	BIT13	=	20000	
010000	BIT12	=	10000	
004000	BIT11	=	4000	
002000	BIT10	=	2000	
001000	BIT09	=	1000	
000400	BIT08	=	400	
000200	BIT07	=	200	
000100	BIT06	=	100	
000040	BIT05	=	40	

```

000020 BIT04 = 20
000010 BIT03 = 10
000004 BIT02 = 4
000002 BIT01 = 2
000001 BIT00 = 1
001000 BIT9=BIT09
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00
    
```

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC = 4          ;;TIME OUT AND OTHER ERRORS
000010 RESVEC = 10       ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14      ;;"T" BIT
000014 TRTVEC = 14       ;;TRACE TRAP
000014 BPTVEC = 14       ;;BREAKPOINT TRAP (BPT)
000020 IOTVEC = 20       ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC = 24       ;;POWER FAIL
000030 EMTVEC = 30       ;;EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC = 34      ;;"TRAP" TRAP
000060 TKVEC = 60        ;;TTY KEYBOARD VECTOR
000064 TPVEC = 64        ;;TTY PRINTER VECTOR
000240 PIRQVEC = 240     ;;PROGRAM INTERRUPT REQUEST VECTOR
    
```

.SBTTL RM80 REGISTER BIT DEFINITIONS

;*RMCS1 CONTROL STATUS REGISTER

```

493 004000 DVA = BIT11      ;DEVICE AVAILABLE-READ ONLY
494 000040 F4 = BIT05      ;FUNCTION CODE
495 000020 F3 = BIT04      ;FUNCTION CODE
496 000010 F2 = BIT03      ;FUNCTION CODE
497 000004 F1 = BIT02      ;FUNCTION CODE
500 000002 F0 = BIT01      ;FUNCTION CODE
501 000001 GO = BIT00     ;GO BIT
502 000077 FNCMSK = 000077 ;FUNCTION CODE MASK
    
```

;*FUNCTION CODES (BITS 01-05 OF RMCS1)

```

503 000000 NOP = 000000 ;NOP COMMAND
504 000002 ILF02 = 000002 ;ILLEGAL COMMAND
505 000004 SEEK = 000004 ;SEEK COMMAND
506 000006 RECAL = 000006 ;RECALIBRATE COMMAND
507 000010 DRVCLR = 000010 ;DRIVE CLEAR COMMAND
508 000012 RELEASE = 000012 ;RELEASE COMMAND
509 000014 OFFSET = 000014 ;OFFSET COMMAND
510 000016 RTC = 000016 ;RETURN TO CENTERLINE COMMAND
511 000020 RIP = 000020 ;READ IN PRESET COMMAND
512 000022 PAKACK = 000022 ;PACK ACKNOWLEDGE COMMAND
513 000022 PACACK = PACACK ;
514 000024 ILF24 = 000024 ;ILLEGAL COMMAND
515 000026 ILF26 = 000026 ;ILLEGAL COMMAND
    
```

521	000030	SEARCH	= 000030	:SEARCH COMMAND
524	000030	ILF30	= 000030	:ILLEGAL COMMAND
	000032	ILF32	= 000032	:ILLEGAL COMMAND
	000034	ILF34	= 000034	:ILLEGAL COMMAND
	000036	ILF36	= 000036	:ILLEGAL COMMAND
	000040	ILF40	= 000040	:ILLEGAL COMMAND
	000042	ILF42	= 000042	:ILLEGAL COMMAND
	000044	ILF44	= 000044	:ILLEGAL COMMAND
	000046	ILF46	= 000046	:ILLEGAL COMMAND
525	000050	WCD	= 000050	:WRITE CHECK DATA COMMAND
526	000052	WCH	= 000052	:WRITE CHECK HEADER AND DATA
527	000054	ILF54	= 000054	:ILLEGAL COMMAND
528	000056	ILF56	= 000056	:ILLEGAL COMMAND
529	000060	WD	= 000060	:WRITE DATA COMMAND
530	000062	WH	= 000062	:WRITE HEADER AND DATA COMMAND
531	000064	ILF64	= 000064	:ILLEGAL COMMAND
532	000066	ILF66	= 000066	:ILLEGAL COMMAND
533	000070	RD	= 000070	:READ DATA COMMAND
534	000072	RH	= 000072	:READ HEADER AND DATA COMMAND
535	000074	ILF74	= 000074	:ILLEGAL COMMAND
536	000076	ILF76	= 000076	:ILLEGAL COMMAND
537				
538		:*RMDA	DISK ADDRESS REGISTER	
539				
540		:TRACK	ADDRESS DEFINITIONS	
541	004000	TA8	= BIT11	:TRACK ADDRESS 8.
542	002000	TA4	= BIT10	:TRACK ADDRESS 4
543	001000	TA2	= BIT09	:TRACK ADDRESS 2
544	000400	TA1	= BIT08	:TRACK ADDRESS 1
545				
546		:SECTOR	ADDRESS DEFINITIONS	
547	000020	SA16	= BIT04	:SECTOR ADDRESS 16.
548	000010	SA8	= BIT03	:SECTOR ADDRESS 8.
549	000004	SA4	= BIT02	:SECTOR ADDRESS 4
550	000002	SA2	= BIT01	:SECTOR ADDRESS 2
551	000001	SA1	= BIT00	:SECTOR ADDRESS 1
552				
553		:TRACK &	SECTOR MASKS	
554	177400	TADMSK	= 177400	:TRACK ADDRESS MASK
555	000377	SADMSK	= 000377	:SECTOR ADDRESS MASK
556				
557		:*RMDS	DRIVE STATUS REGISTER	
558				
559	100000	ATA	= BIT15	:ATTENTION ACTIVE
560	040000	ERR	= BIT14	:COMPOSITE ERROR
561	020000	PIP	= BIT13	:POSITIONING IN PROGRESS
562	010000	MOL	= BIT12	:MEDIUM ON LINE
563	004000	WRL	= BIT11	:WRITE LOCK
564	002000	LBT	= BIT10	:LAST BLOCK TRANSFERRED
565	001000	PGM	= BIT09	:PROGRAMMABLE
566	000400	DPR	= BIT08	:DRIVE PRESENT
567	000200	DRY	= BIT07	:DRIVE READY
568	000100	VV	= BIT06	:VOLUME VALID
569	000001	OM	= BIT00	:OFFSET MODE ACTIVE
570				
571		:*RMER1	ERROR REGISTER #1	
572				

573	100000	DCK	= BIT15	:DATA CHECK ERROR
574	040000	UNS	= BIT14	:DRIVE UNSAFE
575	020000	OPI	= BIT13	:OPERATION INCOMPLETE
576	010000	DTE	= BIT12	:DRIVE TIMING ERROR
577	004000	WLE	= BIT11	:WRITE LOCK ERROR
578	002000	IAE	= BIT10	:INVALID ADDRESS ERROR
579	001000	AOE	= BIT09	:ADDRESS OVERFLOW ERROR
580	000400	HCRC	= BIT08	:HEADER CRC ERROR
581	000200	HCE	= BIT07	:HEADER COMPARE ERROR
582	000100	ECH	= BIT06	:ECC "HARD" ERROR
583	000040	WCF	= BIT05	:WRITE CLOCK FAILURE
584	000020	FER	= BIT04	:FORMAT ERROR
585	000010	PAR	= BIT03	:PARITY ERROR
586	000004	RMR	= BIT02	:REGISTER MODIFICATION REFUSED
587	000002	ILR	= BIT01	:ILLEGAL REGISTER
588	000001	ILF	= BIT00	:ILLEGAL FUNCTION
589				
590	115760	NDTMSK	= DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER	
591			;'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA	
592			:COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS	
593				
594			;*RMAS ATTENTION SUMMARY REGISTER	
595				
596	000377	ATNMSK	= 377	:MASK FOR ATTENTION BITS
597				
598			;*RMLA LOOK AHEAD REGISTER	
599				
600	002000	SC4	= BIT10	:SECTOR COUNT = 16
601	001000	SC3	= BIT09	:SECTOR COUNT = 8
602	000400	SC2	= BIT08	:SECTOR COUNT = 4
603	000200	SC1	= BIT07	:SECTOR COUNT = 2
604	000100	SC0	= BIT06	:SECTOR COUNT = 1
605				
606	003700	SCTMSK	= 003700	:SECTOR COUNT MASK
607				
608			;*RMMR1 MAINTENANCE REGISTER #1	
609				
610			:WRITE ONLY BITS	
611	100000	DBCK	= BIT15	:DEBUG CLOCK
612	040000	DBEN	= BIT14	:DEBUG CLOCK ENABLE
613	020000	DEBL	= BIT13	:DIAGNOSTIC END OF BLOCK
614	010000	DTO	= BIT12	:DIAGNOSTIC TIMEOUT
615	004000	MCLK	= BIT11	:MAINTENANCE CLOCK
616	002000	MRD	= BIT10	:READ DATA
617	001000	MUR	= BIT09	:UNIT READY
618	000400	MOC	= BIT08	:ON CYLINDER
619	000200	MSER	= BIT07	:SEEK ERROR
620	000100	MDF	= BIT06	:DRIVE FAULT
621	000040	MS	= BIT05	:SECTOR PULSE
622	000010	MWP	= BIT03	:WRITE PROTECT
623	000004	MI	= BIT02	:INDEX PULSE
624	000002	MSC	= BIT01	:SECTOR COMPARE
625	000001	DMD	= BIT00	:DIAGNOSTIC MODE
626				
627			:READ ONLY BITS	
628	100000	OCC	= BIT15	:OCCUPIED
629	040000	RG	= BIT14	:RUN AND GO

630	020000	EBL	= BIT13	:END OF BLOCK
631	010000	REX	= BIT12	:EXCEPTION
632	004000	ESRC	= BIT11	:ENABLE SEARCH
633	002000	PLFS	= BIT10	:LOOKING FOR SYNC
634	001000	ECRC	= BIT09	:ENABLE CRC OUT
635	000400	PDA	= BIT08	:DATA AREA
636	000200	PHA	= BIT07	:HEADER AREA
637	000100	CONT	= BIT06	:CONTINUE
638	000040	WC	= BIT05	:WORD CLOCK
639	000020	EECC	= BIT04	:ENABLE ECC OUT
640	000010	MWD	= BIT03	:WRITE DATA BIT
641	000004	LS	= BIT02	:LAST SECTOR
642	000002	LST	= BIT01	:LAST SECTOR AND TRACK
643	000001	DMD	= BIT00	:DIAGNOSTIC MODE
644				
645		:*RMDT DRIVE TYPE REGISTER		
646				
647	100000	NSA	= BIT15	:NOT SECTOR ADDRESSED = 0
648	040000	TAP	= BIT14	:TAPE DRIVE = 0
649	020000	MOH	= BIT13	:MOVING HEAD = 1
650	004000	DRQ	= BIT11	:DRIVE REQUEST REQUIRED
651				
652	020026	SNGPRT	= 020026	:SINGLE PORT DRIVE TYPE
653	024026	DULPRT	= 024026	:DUAL PORT DRIVE TYPE
654				
655		:*RMOF OFFSET REGISTER		
656				
657	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
658	004000	ECI	= BIT11	:ECC INHIBIT
659	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
660	001000	SSEI	= BIT09	:SKIP SECTOR ERROR INHIBIT
661	000200	OFD	= BIT07	:OFFSET FORWARD
662				
663		:*RMDC DESIRED CYLINDER ADDRESS REGISTER		
664				
665	001777	CYLSK	= 001777	:MASK FOR CYLINDER ADDRESS
666				
667		:*RMMR2 MAINTENANCE REGISTER #2		
668				
669		:READ ONLY BITS		
670	100000	RQA	= BIT15	:PORT A REQUEST
671	040000	RQB	= BIT14	:PORT B REQUEST
672	020000	TAG	= BIT13	:TAG CONTROL
673	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
674	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
675	002000	CH	= BIT10	:CONTROL OR HEAD TAG
676	001000	BB09	= BIT09	:TAG BUS
677	000400	BB08	= BIT08	:TAG BUS
678	000200	BB07	= BIT07	:TAG BUS
	000100	BB06	= BIT06	:TAG BUS
	000040	BB05	= BIT05	:TAG BUS
	000020	BB04	= BIT04	:TAG BUS
	000010	BB03	= BIT03	:TAG BUS
	000004	BB02	= BIT02	:TAG BUS
	000002	BB01	= BIT01	:TAG BUS
	000001	BB00	= BIT00	:TAG BUS

```

680          ;*RMER2 ERROR REGISTER 2
681
682          100000      BSE      = BIT15      ;BAD SECTOR ERROR
683          040000      SKI      = BIT14      ;SEEK INCOMPLETE
684          020000      OPE      = BIT13      ;OPERATOR PLUG ERROR
685          010000      IVC      = BIT12      ;INVALID COMMAND ERROR
686          004000      LSC      = BIT11      ;LOSS OF SYSTEM CLOCK
687          002000      LBC      = BIT10      ;LOSS OF BIT CLOCK
688          000200      DVC      = BIT07      ;DEVICE CHECK
689          000040      SSE      = BIT05      ;SKIP SECTOR ERROR
690          000010      DPE      = BIT03      ;DATA PARITY ERROR
691
692          .SBTTL  PROGRAM MNEMONICS
693
694          100000      MSE      = BIT15      ;MANUFACTURING DETECTED SECTOR ERROR
695          040000      USE      = BIT14      ;USER DETECTED SECTOR ERROR
696          020000      SSF      = BIT13      ;SKIP SECTOR FAILURE
697
698          .SBTTL  RM80 REGISTER INDEX VALUES
699
700          000000      RMCS1    = 00        ;CONTROL STATUS REGISTER #1
701          000006      RMDA     = 06        ;DISK ADDRESS REGISTER
702          000012      RMDS     = 12        ;DRIVE STATUS REGISTER
703          000014      RMER1    = 14        ;ERROR REGISTER #1
704          000016      RMAS     = 16        ;ATTENTION SUMMARY REGISTER
705          000020      RMLA     = 20        ;LOOK AHEAD REGISTER
706          000024      RMMR1    = 24        ;MAINTENANCE REGISTER
707          000026      RMDT     = 26        ;DRIVE TYPE REGISTER
708          000030      RMSN     = 30        ;SERIAL NUMBER REGISTER
709          000032      RMOF     = 32        ;OFFSET REGISTER
710          000034      RMDC     = 34        ;DESIRED CYLINDER REGISTER
711          000036      RMHR     = 36        ;HOLDING REGISTER
712          000040      RMMR2    = 40        ;MAINTENANCE REGISTER #2
713          000042      RMER2    = 42        ;ERROR REGISTER #2
714          000044      RMEC1    = 44        ;ECC POSITION REGISTER
715          000046      RMEC2    = 46        ;ECC PATTERN REGISTER
716          000050      ILRG50    = 50        ;ILLEGAL REGISTER 50
717          000052      ILRG52    = 52        ;ILLEGAL REGISTER 52
718          000054      ILRG54    = 54        ;ILLEGAL REGISTER 54
719          000056      ILRG56    = 56        ;ILLEGAL REGISTER 56
720          000060      ILRG60    = 60        ;ILLEGAL REGISTER 60
721          000062      ILRG62    = 62        ;ILLEGAL REGISTER 62
722          000064      ILRG64    = 64        ;ILLEGAL REGISTER 64
723          000066      ILRG66    = 66        ;ILLEGAL REGISTER 66
724          000070      ILRG70    = 70        ;ILLEGAL REGISTER 70
725          000072      ILRG72    = 72        ;ILLEGAL REGISTER 72
726          000074      ILRG74    = 74        ;ILLEGAL REGISTER 74
727          000076      ILRG76    = 76        ;ILLEGAL REGISTER 76
728
729          000077      IDXMSK    = 77        ;MASK FOR REGISTER INDEX NUMBER
730
731          .SBTTL  RH CONTROLLER REGISTER BIT DEFINITIONS
732
733          ;*RMCS1 CONTROL STATUS REGISTER #1
734
735          100000      SC        = BIT15      ;SPECIAL CONDITION-READ ONLY
736          040000      TRE        = BIT14      ;TRANSFER ERROR
  
```

728	020000	MCPE	= BIT13	:MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
729	002000	PSEL	= BIT10	:PORT B SELECT
730	001000	A17	= BIT09	:ADDRESS EXTENSION
731	000400	A16	= BIT08	:ADDRESS EXTENSION
732	000200	RDY	= BIT07	:READY-READ ONLY
733	000100	IE	= BIT06	:INTERRUPT ENABLE
734				
735		;*RMCS2 RH CONTROL STATUS REGISTER #2		
736				
737	100000	DLT	= BIT15	:DATA LATE-READ ONLY
738	040000	WCE	= BIT14	:WRITE CHECK ERROR-READ ONLY
739	020000	UPE	= BIT13	:UNIBUS PARITY ERROR
740	010000	NED	= BIT12	:NONEXISTANT DRIVE-READ ONLY
741	004000	NEM	= BIT11	:NONEXISTANT MEMORY-READ ONLY
742	002000	PGE	= BIT10	:PROGRAM ERROR-READ ONLY
743	001000	MXF	= BIT09	:MISSED TRANSFER
744	000400	MDPE	= BIT08	:MASSBUS DATA BUS PARITY ERROR-READ ONLY
745	000200	OR	= BIT07	:OUTPUT READY-READ ONLY
746	000100	IR	= BIT06	:INPUT READY-READ ONLY
747	000040	CLR	= BIT05	:CONTROLLER CLEAR
748	000020	PAT	= BIT04	:PARITY TEST
749	000010	BAI	= BIT03	:UNIBUS ADDRESS INCREMENT INHIBIT
752	000004	U2	= BIT02	:UNIT SELECT
	000002	U1	= BIT01	:UNIT SELECT
	000001	U0	= BIT00	:UNIT SELECT
753				
754		:UNIT SELECT MASK		
755				
756	000007	UNTMSK	= 7	:UNIT SELECT MASK
757				
758		;*RMCS3 RH70 CONTROL STATUS REGISTER #3		
759				
760	100000	APE	= BIT15	:ADDRESS PARITY ERROR
761	040000	DPEHI	= BIT14	:DATA PARITY ERROR HIGH WORD
762	020000	DPELO	= BIT13	:DATA PARITY ERROR LOW WORD
763	010000	WCEHI	= BIT12	:WRITE CHECK ERROR HIGH WORD
764	004000	WCELO	= BIT11	:WRITE CHECK ERROR LOW WORD
765	002000	DBL	= BIT10	:DOUBLE WORD TRANSFER
766	000100	IE	= BIT06	:INTERRUPT ENABLE
767	000010	IPCK3	= BIT03	:INVERT PARITY CHECK
768	000004	IPCK2	= BIT02	:INVERT PARITY CHECK
769	000002	IPCK1	= BIT01	:INVERT PARITY CHECK
770	000001	IPCK0	= BIT00	:INVERT PARITY CHECK
771				
772		.SBTTL RH CONTROLLER REGISTER INDEX VALUES		
773				
774	000000	RMCS1	= 00	:CONTROL, STATUS REGISTER #1
775	000002	RMWC	= 02	:WORD COUNT REGISTER
776	000004	RMBA	= 04	:BUS ADDRESS REGISTER
777	000010	RMCS2	= 10	:CONTROL, STATUS REGISTER #2
778	000022	RMDB	= 22	:DATA BUFFER
779	000050	RMBAE	= 50	:BUS ADDRESS EXTENSION
780	000052	RMCS3	= 52	:CONTROL, STATUS REGISTER #3
781				
782	176700	ABASE	= 176700	:UNIBUS ADDRESS
783	120254	AVECT1	= 120254	:UNIBUS VECTOR ADDRESS AND PRIORITY
784				

1

.SBTTL TRAP CATCHER

000000

. =0
 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000174 000174
 000176 000000
 000176 000000

. =174
 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)

000200 000137 005434
 2
 3 000204 000137 005424
 4
 5

JMP @#START ;:JUMP TO STARTING ADDRESS OF PROGRAM
 JMP @#START1 ;CHANGE RH/RM BUS ADDRESS

.SBTTL ACT11 HOOKS

 ;HOOKS REQUIRED BY ACT11

000046 000210
 000046 000046
 000052 031402
 000052 000052
 000052 000000
 000210 000210

\$SVPC= . ;SAVE PC
 =46
 \$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .SEOP
 =52
 .WORD 0 ;:2)SET LOC.52 TO ZERO
 =\$SVPC ;: RESTORE PC

6
 7 001100
 8

. =1100
 .SBTTL APT PARAMETER BLOCK

 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
 ;*****

000024 001100
 000024 000024
 000044 000200
 000044 000044
 000044 001100
 001100 001100

.\$X= . ;:SAVE CURRENT LOCATION
 =24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
 200 ;:FOR APT START UP
 =44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
 \$APTHDR ;:POINT TO APT HEADER BLOCK
 =.\$X ;:RESET LOCATION COUNTER

 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
 ;INTERFACE SPEC.

001100
 001100 000000
 001102 001222
 001104 000024
 001106 000024
 001110 000024
 001112 000042
 9 001114

\$APTHD:
 \$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
 \$MADR: .WORD \$MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
 \$STM: .WORD 20. ;:RUN TIM OF LONGEST TEST
 \$PSTM: .WORD 20. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 \$UNITM: .WORD 20. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
 .WORD \$ETEND-\$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
 TAGADR=.

001230	000000	\$PASS:	.WORD	APASS	::PASS COUNT
001232	000000	\$DEVCT:	.WORD	ADEVCT	::DEVICE COUNT
001234	000000	\$UNIT:	.WORD	AUNIT	::I/O UNIT NUMBER
001236	000000	\$MSGAD:	.WORD	AMSGAD	::MESSAGE ADDRESS
001240	000000	\$MSGLG:	.WORD	AMSGLG	::MESSAGE LENGTH
001242		\$ETABLE:			::APT ENVIRONMENT TABLE
001242	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
001243	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
001244	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
001246	000000	\$USWR:	.WORD	AUSWR	::USER SWITCHES
001250	000000	\$CPUOP:	.WORD	ACPUOP	::CPU TYPE,OPTIONS
		*			BITS 15-11=CPU TYPE
		*			11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
		*			11/70=06,PDQ=07,Q=10
		*			BIT 10=REAL TIME CLOCK
		*			BIT 9=FLOATING POINT PROCESSOR
		*			BIT 8=MEMORY MANAGEMENT
001252	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001253	000	\$MTYP1:	.BYTE	AMTYP1	::MEM. TYPE,BLK#1
		*			MEM.TYPE BYTE -- (HIGH BYTE)
		*			900 NSEC CORE=001
		*			300 NSEC BIPOLAR=002
		*			500 NSEC MOS=003
001254	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
		*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001256	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001257	000	\$MTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
001260	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001262	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001263	000	\$MTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
001264	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001266	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001267	000	\$MTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
001270	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001272	120254	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001274	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001276	176700	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001300	000000	\$DEVW:	.WORD	ADEVW	::DEVICE MAP
001302	000000	\$CDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001304	000000	\$CDW2:	.WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001306	000000	\$DDW0:	.WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
001310	000000	\$DDW1:	.WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
001312	000000	\$DDW2:	.WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
001314	000000	\$DDW3:	.WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
001316	000000	\$DDW4:	.WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
001320	000000	\$DDW5:	.WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
001322	000000	\$DDW6:	.WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
001324	000000	\$DDW7:	.WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
001326		\$ETEND:			
		.MEXIT			

.SBTTL USER DEFINED TAGS

001326	000000	CTLFG:	.WORD	0	:CONTAINS CONTROL-C FLAG
001330	000000	CHGADR:	.WORD	0	:CHANGE RH/RM BUS ADDRESS = -1, NO CHANGE = 0
001332	000000	XXDP:	.WORD	0	:THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH :THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE : 'XXDP' DEVICE CODE FOR THE RM80.
001334	000	LSTRK:	.BYTE	0	:LO BYTE = 0
001335	015		.BYTE	13.	:HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT :UNDER TEST. RM80 = 13.

:THE REGISTER INPUT BUFFER IS USED FOR
 :STORING DRIVE STATUS

001336

GETBUF:

		:REGISTER INPUT BUFFER			
001336	000000	RMCS1I:	.WORD	0	:CONTROL, STATUS REGISTER #1
001340	000000	RMWCI:	.WORD	0	:WORD COUNT REGISTER
001342	000000	RMBAI:	.WORD	0	:BUS ADDRESS REGISTER
001344	000000	RMDAI:	.WORD	0	:DISK ADDRESS REGISTER
001346	000000	RMCS2I:	.WORD	0	:CONTROL, STATUS REGISTER #2
001350	000000	RMDSI:	.WORD	0	:DRIVE STATUS REGISTER
001352	000000	RMER1I:	.WORD	0	:ERROR REGISTER #1
001354	000000	RMASI:	.WORD	0	:ATTENTION SUMMARY REGISTER
001356	000000	RMLAI:	.WORD	0	:LOOK AHEAD REGISTER
001360	000000	RMDBI:	.WORD	0	:DATA BUFFER
001362	000000	RMMR1I:	.WORD	0	:MAINTENANCE REGISTER #1
001364	000000	RMDTI:	.WORD	0	:DRIVE TYPE REGISTER
001366	000000	RMSNI:	.WORD	0	:SERIAL NUMBER REGISTER
001370	000000	RMOFI:	.WORD	0	:OFFSET REGISTER
001372	000000	RMDCI:	.WORD	0	:DESIRED CYLINDER REGISTER
001374	000000	RMHRI:	.WORD	0	:HOLDING REGISTER
001376	000000	RMMR2I:	.WORD	0	:MAINTENANCE REGISTER #2
001400	000000	RMER2I:	.WORD	0	:ERROR REGISTER #2
001402	000000	RMEC1I:	.WORD	0	:ECC POSITION REGISTER
001404	000000	RMEC2I:	.WORD	0	:ECC PATTERN REGISTER
001406	000000	RMBAEI:	.WORD	0	:BUS ADDRESS EXTENSION REGISTER
001410	000000	RMCS3I:	.WORD	0	:CONTROL, STATUS REGISTER #3

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

001412

PUTBUF:

		:REGISTER OUTPUT BUFFER			
001412	000000	RMCS1O:	.WORD	0	:CONTROL, STATUS REGISTER #1
001414	000000	RMWCO:	.WORD	0	:WORD COUNT REGISTER
001416	000000	RMBAO:	.WORD	0	:BUS ADDRESS REGISTER
001420	000000	RMDAO:	.WORD	0	:DISK ADDRESS REGISTER
001422	000000	RMCS2O:	.WORD	0	:CONTROL, STATUS REGISTER #2
001424	000000	RMDSO:	.WORD	0	:DRIVE STATUS REGISTER
001426	000000	RMER1O:	.WORD	0	:ERROR REGISTER #1
001430	000000	RMASO:	.WORD	0	:ATTENTION SUMMARY REGISTER
001432	000000	RMLAO:	.WORD	0	:LOOK AHEAD REGISTER
001434	000000	RMDBO:	.WORD	0	:DATA BUFFER
001436	000000	RMMR1O:	.WORD	0	:MAINTENANCE REGISTER #1

001440	000000	RMDTO: .WORD	0	;DRIVE TYPE REGISTER
001442	000000	RMSNO: .WORD	0	;SERIAL NUMBER REGISTER
001444	000000	RMOFO: .WORD	0	;OFFSET REGISTER
001446	000000	RMDCO: .WORD	0	;DESIRED CYLINDER REGISTER
001450	000000	RMHRO: .WORD	0	;HOLDING REGISTER
001452	000000	RMMR20: .WORD	0	;MAINTENANCE REGISTER #2
001454	000000	RMER20: .WORD	0	;ERROR REGISTER #2
001456	000000	RMEC10: .WORD	0	;ECC POSITION REGISTER
001460	000000	RMEC20: .WORD	0	;ECC PATTERN REGISTER
001462	000000	RMBAE0: .WORD	0	;BUS ADDRESS EXTENSION REGISTER
001464	000000	RMCS30: .WORD	0	;CONTROL, STATUS REGISTER #3

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

001466	000000	TSTQUE: .WORD	0	;CONTAINS DEVICE POINTER
001470		.BLKW	8.	;TEST QUE FOR DEVICES UNDER TEST
001510	000000	.WORD	0	;TABLE TERMINATOR GOES HERE WHEN ;ALL 8. DEVICES ARE UNDER TEST.

;MEDIA ENABLE IS SET IF THE BAD SECTOR FILES HAVE BEEN RECOVERED
 ;FOR THE UNIT UNDER TEST, OTHERWISE IT IS ZERO.
 MEDENB: .WORD 0 ;MEDIA ENABLE

001512	000000			
001514	000000	SSFENB: .WORD	0	;SSF ENABLE

;SKIP SECTOR FAILURE ENABLE IS SET IF THE "SSF" BIT SHOULD BE
 ;SET IN THE FIRST HEADER WORD WHEN THE FE CYLINDERS ARE BEING
 ;REFORMATTED.

001516	000000	ASNDC: .WORD	0	;ASSIGNED DESIRED CYLINDER
001520	000000	ASNDA: .WORD	0	;ASSIGNED TRACK, AND SECTOR
001522	000000	CLKADR: .WORD	0	;UNIBUS ADDRESS OF KW11 CLOCK
001524	000000	CLKVCT: .WORD	0	;VECTOR ADDRESS OF KW11 CLOCK

;LOCATIONS "ASNDC" AND "ASNDC" CONTAIN THE CYLINDER, TRACK AND SECTOR
 ;ADDRESS ASSIGNED BY THE BAD SECTOR MODULE.

001526		GETINX: .BLKB	23.	;GET INDEX TABLE
--------	--	---------------	-----	------------------

;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.

001555		PUTINX: .BLKB	23.	;PUT 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.

;PUT TAGS HERE

.SBTTL ERROR POINTER TABLE

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

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

	001604	\$ERRTB:	
1		;ERROR 1	WRONG UNIT SELECTED
2			
3	001604 065442	EMT1	
	001606 071600	EHT1	
	001610 071724	EDT1	
	001612 072014	EFT1	
4		;ERROR 2	DEVICE WENT UNAVAILABLE
5			
6	001614 065446	EMT2	
	001616 071600	EHT1	
	001620 071724	EDT1	
	001622 072014	EFT1	
7		;ERROR 3	DEVICE WENT NONEXISTENT
8			
9	001624 065454	EMT3	
	001626 071600	EHT1	
	001630 071724	EDT1	
	001632 072014	EFT1	
10		;ERROR 4	CONTROLLER NOT READY
11			
12	001634 065462	EMT4	
	001636 071600	EHT1	
	001640 071724	EDT1	
	001642 072014	EFT1	
13		;ERROR 5	DRIVE NOT READY AND GO NOT RESET
14			
15	001644 065470	EMT5	
	001646 071600	EHT1	
	001650 071724	EDT1	
	001652 072014	EFT1	
16		;ERROR 6	UNEXPECTED VALUE FOR "ATA" STATUS
17			

18	001654	065476	EMT6	
	001656	071600	EHT1	
	001660	071724	EDT1	
	001662	072014	EFT1	
19				
20			:ERROR 7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER
21	001664	065504	EMT7	
	001666	000000	0	
	001670	000000	0	
	001672	000000	0	
22				
23			:ERROR 10	DRIVE NOT READY BUT GO IS RESET
24	001674	065512	EMT10	
	001676	071600	EHT1	
	001700	071724	EDT1	
	001702	072014	EFT1	
25				
26			:ERROR 11	GO NOT RESET BUT DRIVE IS READY
27	001704	065516	EMT11	
	001706	071600	EHT1	
	001710	071724	EDT1	
	001712	072014	EFT1	
28				
29			:ERROR 12	INCORRECT FUNCTION CODE
30	001714	065522	EMT12	
	001716	071600	EHT1	
	001720	071724	EDT1	
	001722	072014	EFT1	
31				
32			:ERROR 13	PARITY ERROR READING REMOTE REGISTERS
33	001724	065530	EMT13	
	001726	071600	EHT1	
	001730	071724	EDT1	
	001732	072014	EFT1	
34				
35			:ERROR 14	TRANSFER ERROR IS INCORRECT
36	001734	065542	EMT14	
	001736	071600	EHT1	
	001740	071724	EDT1	
	001742	072014	EFT1	
37				
38			:ERROR 15	INCORRECT WORD COUNT
39				

001744	065550	EMT15
001746	071600	EHT1
001750	071724	EDT1
001752	072014	EFT1
40		
41	:ERROR 16	INCORRECT BUS ADDRESS
42		
001754	065556	EMT16
001756	071600	EHT1
001760	071724	EDT1
001762	072014	EFT1
43		
44	:ERROR 17	INCORRECT LBT STATUS
45		
001764	065566	EMT17
001766	071600	EHT1
001770	071724	EDT1
001772	072014	EFT1
46		
47	:ERROR 20	INCORRECT AOE
48		
001774	065576	EMT20
001776	071600	EHT1
002000	071724	EDT1
002002	072014	EFT1
49		
50	:ERROR 21	INCORRECT DISK ADDRESS
51		
002004	065606	EMT21
002006	071600	EHT1
002010	071724	EDT1
002012	072014	EFT1
52		
53	:ERROR 22	INCORRECT CYLINDER ADDRESS
54		
002014	065616	EMT22
002016	071600	EHT1
002020	071724	EDT1
002022	072014	EFT1
55		
56	:ERROR 23	INCORRECT WLE STATUS
57		
002024	065626	EMT23
002026	071600	EHT1
002030	071724	EDT1
002032	072014	EFT1
58		
59	:ERROR 24	INCORRECT UPE STATUS
60		
002034	065636	EMT24

	002036 071600	EHT1	
	002040 071724	EDT1	
	002042 072014	EFT1	
61			
62		:ERROR 25	INCORRECT WCF STATUS
63			
	002044 065646	EMT25	
	002046 071600	EHT1	
	002050 071724	EDT1	
	002052 072014	EFT1	
64			
65		:ERROR 26	INCORRECT WCE STATUS
66			
	002054 065656	EMT26	
	002056 071600	EHT1	
	002060 071724	EDT1	
	002062 072014	EFT1	
67			
68		:ERROR 27	INCORRECT MDPE STATUS
69			
	002064 065666	EMT27	
	002066 071600	EHT1	
	002070 071724	EDT1	
	002072 072014	EFT1	
70			
71		:ERROR 30	INCORRECT DCK STATUS
72			
	002074 065676	EMT30	
	002076 071600	EHT1	
	002100 071724	EDT1	
	002102 072014	EFT1	
73			
74		:ERROR 31	INCORRECT ECH STATUS
75			
	002104 065706	EMT31	
	002106 071600	EHT1	
	002110 071724	EDT1	
	002112 072014	EFT1	
76			
77		:ERROR 32	DLT SHOULD NOT BE SET
78			
	002114 065716	EMT32	
	002116 071600	EHT1	
	002120 071724	EDT1	
	002122 072014	EFT1	
79			
80		:ERROR 33	MXF SHOULD NOT BE SET
81			
	002124 065726	EMT33	
	002126 071600	EHT1	

	002130 071724	EDT1	
	002132 072014	EFT1	
82			
83			
84			:ERROR 34 DTE SHOULD NOT BE SET
	002134 065736	EMT34	
	002136 071600	EHT1	
	002140 071724	EDT1	
	002142 072014	EFT1	
85			
86			:ERROR 35 INCORRECT HCRC STATUS
87			
	002144 065746	EMT35	
	002146 071600	EHT1	
	002150 071724	EDT1	
	002152 072014	EFT1	
88			
89			:ERROR 36 INCORRECT HCE STATUS
90			
	002154 065756	EMT36	
	002156 071600	EHT1	
	002160 071724	EDT1	
	002162 072014	EFT1	
91			
92			:ERROR 37 INCORRECT FER STATUS
93			
	002164 065766	EMT37	
	002166 071600	EHT1	
	002170 071724	EDT1	
	002172 072014	EFT1	
94			
95			:ERROR 40 DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
96			
	002174 065776	EMT40	
	002176 071600	EHT1	
	002200 071724	EDT1	
	002202 072014	EFT1	
97			
98			:ERROR 41 LOST 'MOL' DURING PACK ACKNOWLEDGE
99			
	002204 066004	EMT41	
	002206 071600	EHT1	
	002210 071724	EDT1	
	002212 072014	EFT1	
100			
101			:ERROR 42 UNSAFE ERROR DURING PACK ACKNOWLEDGE
102			
	002214 066014	EMT42	
	002216 071600	EHT1	
	002220 071724	EDT1	

	002222	072014	EFT1	
103				
104			:ERROR 43	'OPI' ERROR DURING PACK ACKNOWLEDGE
105				
	002224	066026	EMT43	
	002226	071600	EHT1	
	002230	071724	EDT1	
	002232	072014	EFT1	
106				
107			:ERROR 44	'RMR' ERROR DURING PACK ACKNOWLEDGE
108				
	002234	066036	EMT44	
	002236	071600	EHT1	
	002240	071724	EDT1	
	002242	072014	EFT1	
109				
110			:ERROR 45	'ILR' ERROR DURING PACK ACKNOWLEDGE
111				
	002244	066046	EMT45	
	002246	071600	EHT1	
	002250	071724	EDT1	
	002252	072014	EFT1	
112				
113			:ERROR 46	'ILF' ERROR DURING PACK ACKNOWLEDGE
114				
	002254	066056	EMT46	
	002256	071600	EHT1	
	002260	071724	EDT1	
	002262	072014	EFT1	
115				
116			:ERROR 47	COMPOSITE ERROR STATUS IS INCORRECT
117				
	002264	066066	EMT47	
	002266	071600	EHT1	
	002270	071724	EDT1	
	002272	072014	EFT1	
118				
119			:ERROR 50	PARITY ERROR WRITING REMOTE REGISTERS
120				
	002274	066074	EMT50	
	002276	071600	EHT1	
	002300	071724	EDT1	
	002302	072014	EFT1	
121				
122			:ERROR 51	INCORRECT IAE STATUS DURING SEEK COMMAND
123				
	002304	066104	EMT51	
	002306	071600	EHT1	
	002310	071724	EDT1	
	002312	072014	EFT1	

124				
125			:ERROR 52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE
126				
	002314	066116	EMT52	
	002316	071600	EHT1	
	002320	071724	EDT1	
	002322	072014	EFT1	
127				
128			:ERROR 53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE, ASSUME
129			:	ON CYLINDER LATCH DIDN'T RESET
130				
	002324	066134	EMT53	
	002326	071600	EHT1	
	002330	071724	EDT1	
	002332	072014	EFT1	
131				
132			:ERROR 54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND
133				
	002334	066152	EMT54	
	002336	071600	EHT1	
	002340	071724	EDT1	
	002342	072014	EFT1	
134				
135			:ERROR 55	DEVICE CHECK DURING SEEK COMMAND
136				
	002344	066162	EMT55	
	002346	071600	EHT1	
	002350	071724	EDT1	
	002352	072014	EFT1	
137				
138			:ERROR 56	PIP IS STILL SET AFTER SEEK - SKI IS RESET
139				
	002354	066174	EMT56	
	002356	071600	EHT1	
	002360	071724	EDT1	
	002362	072014	EFT1	
140				
141			:ERROR 57	ATA DID NOT SET DURING SEEK COMMAND
142				
	002364	066212	EMT57	
	002366	071600	EHT1	
	002370	071724	EDT1	
	002372	072014	EFT1	
143				
144			:ERROR 60	IVC ERROR DURING SEEK COMMAND - LOST
145			:	VOLUME VALID
146				
	002374	066222	EMT60	
	002376	071600	EHT1	
	002400	071724	EDT1	

ERROR POINTER TABLE

	002402	072014	EFT1	
147				
148			:ERROR 61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
149			:	VOLUME VALID IS STIL SET
150				
	002404	066240	EMT61	
	002406	071600	EHT1	
	002410	071724	EDT1	
	002412	072014	EFT1	
151				
152			:ERROR 62	MOL IS ZERO, BUT OPI WAS NOT
153			:	REPORTED DURING SEEK COMMAND
154				
	002414	066260	FMT62	
	002416	071600	EHT1	
	002420	071724	EDT1	
	002422	072014	EFT1	
155				
156			:ERROR 63	SSE SHOULD NOT BE SET
157				
	002424	066274	EMT63	
	002426	071600	EHT1	
	002430	071724	EDT1	
	002432	072014	EFT1	
158				
159			:ERROR 64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
160				
	002434	066304	EMT64	
	002436	071600	EHT1	
	002440	071724	EDT1	
	002442	072014	EFT1	
161				
162			:ERROR 65	DRIVE EXECUTED A SEEK WITH ERROR SET
163				
	002444	066324	EMT65	
	002446	071600	EHT1	
	002450	071724	EDT1	
	002452	072014	EFT1	
164				
165			:ERROR 66	UNEXPECTED ERROR SET IN RMER1
166				
	002454	066344	EMT66	
	002456	071600	EHT1	
	002460	071724	EDT1	
	002462	072014	EFT1	
167				
168			:ERROR 67	UNEXPECTED ERROR SET IN RMER2
169				
	002464	066356	EMT67	
	002466	071600	EHT1	

	002470 071724	EDT1	
	002472 072014	EFT1	
170			
171		;ERROR 70	ERRONEOUS "IAE" ERROR DURING RECALIBRATE
172			
	002474 066370	EMT70	
	002476 071600	EHT1	
	002500 071724	EDT1	
	002502 072014	EFT1	
173			
174		;ERROR 71	"ILF" ERROR DURING RECALIBRATE
175			
	002504 066400	EMT71	
	002506 071600	EHT1	
	002510 071724	EDT1	
	002512 072014	EFT1	
176			
177		;ERROR 72	"OPI" ERROR DURING RECALIBRATE DUE TO "MOL" = 0
178			
	002514 066410	EMT72	
	002516 071600	EHT1	
	002520 071724	EDT1	
	002522 072014	EFT1	
179			
180		;ERROR 73	"OPI" ERROR DURING RECALIBRATE BECAUSE ON
181		:	CYLINDER DIDNT DROP
182			
	002524 066426	EMT73	
	002526 071600	EHT1	
	002530 071724	EDT1	
	002532 072014	EFT1	
183			
184		;ERROR 74	"IVC" ERROR DURING RECALIBRATE - "VV" = 0
185			
	002534 066444	EMT74	
	002536 071600	EHT1	
	002540 071724	EDT1	
	002542 072014	EFT1	
186			
187		;ERROR 75	ERRONEOUS "IVC" ERROR DURING RECALIBRATE - "VV" = 1
188			
	002544 066454	EMT75	
	002546 071600	EHT1	
	002550 071724	EDT1	
	002552 072014	EFT1	
189			
190		;ERROR 76	"SKI" ERROR DURING RECALIBRATE
191			
	002554 066474	EMT76	
	002556 071600	EHT1	

	002560 071724	EDT1	
	002562 072014	EFT1	
192			
193		:ERROR 77	"DVC" OCCURRED DURING RECALIBRATE
194			
	002564 066504	EMT77	
	002566 071600	EHT1	
	002570 071724	EDT1	
	002572 072014	EFT1	
195			
196		:ERROR 100	LOST "MOL" DURING RECALIBRATE - "GPI" = 0
197			
	002574 066516	EMT100	
	002576 071600	EHT1	
	002600 071724	EDT1	
	002602 072014	EFT1	
198			
199		:ERROR 101	LOST "VV" DURING RECALIBRATE - "IVC" = 0
200			
	002604 066534	EMT101	
	002606 071600	EHT1	
	002610 071724	EDT1	
	002612 072014	EFT1	
201			
202		:ERROR 102	"ATA" DID NOT SET DURING RECALIBRATE
203			
	002614 066552	EMT102	
	002616 071600	EHT1	
	002620 071724	EDT1	
	002622 072014	EFT1	
204			
205		:ERROR 103	"OM" DID NOT RESET DURING RECALIBRATE
206			
	002624 066562	EMT103	
	002626 071600	EHT1	
	002630 071724	EDT1	
	002632 072014	EFT1	
207			
208		:ERROR 104	"PIP" IS STIL SET AFTER RECALIBRATE
209			
	002634 066574	EMT104	
	002636 071600	EHT1	
	002640 071724	EDT1	
	002642 072014	EFT1	
210			
211		:ERROR 105	UNEXPECTED "ILR" ERROR DURING RECALIBRATE
212			
	002644 066612	EMT105	
	002646 071600	EHT1	
	002650 071724	EDT1	

	002652	072014	EFT1	
213				
214			:ERROR 106	UNEXPECTED 'RMR' ERROR DURING RECALIBRATE
215				
	002654	066622	EMT106	
	002656	071600	EHT1	
	002660	071724	EDT1	
	002662	072014	EFT1	
216				
217			:ERROR 107	'UNS' ERROR DURING RECALIBRATE - AC POWER IS LOW
218				
	002664	066632	EMT107	
	002666	071600	EHT1	
	002670	071724	EDT1	
	002672	072014	EFT1	
219				
220			:ERROR 110	CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
221				
	002674	066652	EMT110	
	002676	071622	EHT110	
	002700	071742	EDT110	
	002702	072032	EFT110	
222				
223			:ERROR 111	NONEXISTENT DEVICE
224				
	002704	066664	EMT111	
	002706	071626	EHT111	
	002710	071744	EDT111	
	002712	072034	EFT111	
225				
226			:ERROR 112	DEVICE NOT AVAILABLE
227				
	002714	066672	EMT112	
	002716	071626	EHT111	
	002720	071744	EDT111	
	002722	072034	EFT111	
228				
229			:ERROR 113	BUS TIMEOUT-NED STATUS FAILURE
230				
	002724	066700	EMT113	
	002726	000000	0	
	002730	000000	0	
	002732	000000	0	
231				
232			:ERROR 114	UNUSED
233				
	002734	000000	0	
	002736	000000	0	
	002740	000000	0	
	002742	000000	0	

234				
235			;ERROR	115 RMCS1 NOT INITIALIZED BY UNIBUS
236	002744	066716		EMT115
	002746	071600		EHT1
	002750	071724		EDT1
	002752	072014		EFT1
237				
238			;ERROR	116 RMBA NOT INITIALIZED BY UNIBUS
239	002754	066726		EMT116
	002756	071600		EHT1
	002760	071724		EDT1
	002762	072014		EFT1
240				
241			;ERROR	117 RMCS2 NOT INITIALIZED BY UNIBUS
242	002764	066736		EMT117
	002766	071600		EHT1
	002770	071724		EDT1
	002772	072014		EFT1
243				
244			;ERROR	120 RMER1 NOT INITIALIZED BY UNIBUS
245	002774	066746		EMT120
	002776	071600		EHT1
	003000	071724		EDT1
	003002	072014		EFT1
246				
247			;ERROR	121 RMAS NOT INITIALIZED BY UNIBUS
248	003004	066756		EMT121
	003006	071600		EHT1
	003010	071724		EDT1
	003012	072014		EFT1
249				
250			;ERROR	122 RMMR1 NOT INITIALIZED BY UNIBUS
251	003014	066766		EMT122
	003016	071600		EHT1
	003020	071724		EDT1
	003022	072014		EFT1
252				
253			;ERROR	123 RMDS NOT INITIALIZED BY UNIBUS
254	003024	066776		EMT123
	003026	071600		EHT1
	003030	071724		EDT1
	003032	072014		EFT1

255			
256			
257			
	003034	067006	
	003036	071600	
	003040	071724	
	003042	072014	
			EMT124
			EHT1
			EDT1
			EFT1
258			
259			
260			
	003044	067016	
	003046	071600	
	003050	071724	
	003052	072014	
			EMT125
			EHT1
			EDT1
			EFT1
261			
262			
263			
	003054	067026	
	003056	071600	
	003060	071724	
	003062	072014	
			EMT126
			EHT1
			EDT1
			EFT1
264			
265			
266			
	003064	067040	
	003066	071600	
	003070	071724	
	003072	072014	
			EMT127
			EHT1
			EDT1
			EFT1
267			
268			
269			
	003074	067052	
	003076	071600	
	003100	071724	
	003102	072014	
			EMT130
			EHT1
			EDT1
			EFT1
270			
271			
272			
	003104	067064	
	003106	071600	
	003110	071724	
	003112	072014	
			EMT131
			EHT1
			EDT1
			EFT1
273			
274			
275			
	003114	067076	
	003116	071600	
	003120	071724	
	003122	072014	
			EMT132
			EHT1
			EDT1
			EFT1

276

277		:ERROR 133	RMMR1 NOT CLEARED BY CONTROLLER CLEAR
278	003124 067110	EMT133	
	003126 071600	EHT1	
	003130 071724	EDT1	
	003132 072014	EFT1	
279		:ERROR 134	RMDS NOT CLEARED BY CONTROLLER CLEAR
280			
281	003134 067122	EMT134	
	003136 071600	EHT1	
	003140 071724	EDT1	
	003142 072014	EFT1	
282		:ERROR 135	RMEC2 NOT CLEARED BY CONTROLLER CLEAR
283			
284	003144 067134	EMT135	
	003146 071600	EHT1	
	003150 071724	EDT1	
	003152 072014	EFT1	
285		:ERROR 136	RMMR2 NOT CLEARED BY CONTROLLER CLEAR
286			
287	003154 067146	EMT136	
	003156 071600	EHT1	
	003160 071724	EDT1	
	003162 072014	EFT1	
288		:ERROR 137	RMCS1 NOT CLEARED BY ERROR CLEAR
289			
290	003164 067160	EMT137	
	003166 071600	EHT1	
	003170 071724	EDT1	
	003172 072014	EFT1	
291		:ERROR 140	RMCS2 NOT CLEARED BY ERROR CLEAR
292			
293	003174 067170	EMT140	
	003176 071600	EHT1	
	003200 071724	EDT1	
	003202 072014	EFT1	
294		:ERROR 141	RMCS1 NOT CLEARED BY DRIVE CLEAR
295			
296	003204 067200	EMT141	
	003206 071600	EHT1	
	003210 071724	EDT1	
	003212 072014	EFT1	
297		:ERROR 142	RMDS NOT CLEARED BY DRIVE CLEAR
298			

299	003214 067210	EMT142	
	003216 071600	EHT1	
	003220 071724	EDT1	
	003222 072014	EFT1	
300			
301		:ERROR 143	RMER1 NOT CLEARED BY DRIVE CLEAR
302	003224 067220	EMT143	
	003226 071600	EHT1	
	003230 071724	EDT1	
	003232 072014	EFT1	
303			
304		:ERROR 144	RMAS NOT CLEARED BY DRIVE CLEAR
305	003234 067230	EMT144	
	003236 071600	EHT1	
	003240 071724	EDT1	
	003242 072014	EFT1	
306			
307		:ERROR 145	RMMR1 NOT CLEARED BY DRIVE CLEAR
308	003244 067240	EMT145	
	003246 071600	EHT1	
	003250 071724	EDT1	
	003252 072014	EFT1	
309			
310		:ERROR 146	RMMR2 NOT CLEARED BY DRIVE CLEAR
311	003254 067250	EMT146	
	003256 071600	EHT1	
	003260 071724	EDT1	
	003262 072014	EFT1	
312			
313		:ERROR 147	RMER2 NOT CLEARED BY DRIVE CLEAR
314	003264 067260	EMT147	
	003266 071600	EHT1	
	003270 071724	EDT1	
	003272 072014	EFT1	
315			
316		:ERROR 150	RMEC2 NOT CLEARED BY DRIVE CLEAR
317	003274 067270	EMT150	
	003276 071600	EHT1	
	003300 071724	EDT1	
	003302 072014	EFT1	
318			
319		:ERROR 151	MEDIUM NOT ON LINE
320			

003304	067300	EMT151
003306	071600	EHT1
003310	071724	EDT1
003312	072014	EFT1
321		
322		
323		:ERROR 152 DRIVE FAULT
003314	067312	EMT152
003316	071600	EHT1
003320	071724	EDT1
003322	072014	EFT1
324		
325		
326		:ERROR 153 UNSAFE SHOULD BE SET BECAUSE DVC IS SET
003324	067324	EMT153
003326	071600	EHT1
003330	071724	EDT1
003332	072014	EFT1
327		
328		
329		:ERROR 154 UNSAFE SHOULD NOT BE SET, AC IS LOW
003334	067342	EMT154
003336	071600	EHT1
003340	071724	EDT1
003342	072014	EFT1
330		
331		
332		:ERROR 155 VOLUME VALID NOT SET BY PACK ACK
003344	067360	EMT155
003346	071600	EHT1
003350	071724	EDT1
003352	072014	EFT1
333		
334		
335		:ERROR 156 OFFSET MODE NOT SET BY OFFSET COMMAND
003354	067372	EMT156
003356	071600	EHT1
003360	071724	EDT1
003362	072014	EFT1
336		
337		
338		:ERROR 157 OFFSET MODE NOT RESET BY RTC COMMAND
003364	067404	EMT157
003366	071600	EHT1
003370	071724	EDT1
003372	072014	EFT1
339		
340		
341		:ERROR 160 RMOF NOT RESET BY RIP COMMAND
003374	067416	EMT160

	003376 071600	EHT1	
	003400 071724	EDT1	
	003402 072014	EFT1	
342			
343			
344			:ERROR 161 RMDA NOT RESET BY RIP COMMAND
	003404 067426	EMT161	
	003406 071600	EHT1	
	003410 071724	EDT1	
	003412 072014	EFT1	
345			
346			
347			:ERROR 162 RMDC NOT RESET BY RIP COMMAND
	003414 067440	EMT162	
	003416 071600	EHT1	
	003420 071724	EDT1	
	003422 072014	EFT1	
348			
349			
350			:ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
351			: WRITE BUFFER
	003424 071374	EMT336	
	003426 071662	EHT336	
	003430 071760	EDT336	
	003432 072050	EFT336	
352			
353			
354			:ERROR 164 OPI SHOULD NOT BE SET
	003434 067462	EMT164	
	003436 071600	EHT1	
	003440 071724	EDT1	
	003442 072014	EFT1	
355			
356			
357			:ERROR 165 IVC SHOULD NOT BE SET
	003444 067470	EMT165	
	003446 071600	EHT1	
	003450 071724	EDT1	
	003452 072014	EFT1	
358			
359			
360			:ERROR 166 IAE SHOULD NOT BE SET
	003454 067476	EMT166	
	003456 071600	EHT1	
	003460 071724	EDT1	
	003462 072014	EFT1	
361			
362			
363			:ERROR 167 NEM SHOULD NOT BE SET
	003464 067504	EMT167	

	003466 071600	EHT1	
	003470 071724	EDT1	
	003472 072014	EFT1	
364			
365			
366			:ERROR 170 INCORRECT 'MOL' STATUS DURING DIAGNOSTIC MODE
	003474 067512	MT170	
	003476 071600	EHT1	
	003500 071724	EDT1	
	003502 072014	EFT1	
367			
368			
369			:ERROR 171 'ATA' NOT SET DURING RETURN TO CENTERLINE
	003504 067524	EMT171	
	003506 071600	EHT1	
	003510 071724	EDT1	
	003512 072014	EFT1	
370			
371			
372			:ERROR 172 'ATA' NOT SET BY OFFSET COMMAND
	003514 067534	EMT172	
	003516 071600	EHT1	
	003520 071724	EDT1	
	003522 072014	EFT1	
373			
374			
375			:ERROR 173 RMER2 NOT INITIALIZED BY UNIBUS INIT
	003524 067544	EMT173	
	003526 071600	EHT1	
	003530 071724	EDT1	
	003532 072014	EFT1	
376			
377			
378			:ERROR 174 RMER2 NOT INITIALIZED BY CONTROLLER CLEAR
	003534 067554	EMT174	
	003536 071600	EHT1	
	003540 071724	EDT1	
	003542 072014	EFT1	
379			
380			
381			:ERROR 175 SELECTED DEVICE IS IN WRITE PROTECT
	003544 067566	EMT175	
	003546 071600	EHT1	
	003550 071724	EDT1	
	003552 072014	EFT1	
382			
383			
384			:ERROR 176 CANNOT SET DIAGNOSTIC MODE
	003554 067574	EMT176	
	003556 071600	EHT1	

	003560 071724	EDT1	
	003562 072014	EFT1	
385			
386		:ERROR 177	--RESERVED FOR POWER MONITOR BIT FAILURE--
387			
	003564 000000	0	
	003566 000000	0	
	003570 000000	0	
	003572 000000	0	
388			
389		:ERROR 200	INCORRECT 'PIP' STATUS DURING DIAGNOSTIC MODE
390			
	003574 067604	EMT200	
	003576 071600	EHT1	
	003600 071724	EDT1	
	003602 072014	EFT1	
391			
392		:ERROR 201	INCORRECT 'WRL' STATUS DURING DIAGNOSTIC MODE
393			
	003604 067616	EMT201	
	003606 071600	EHT1	
	003610 071724	EDT1	
	003612 072014	EFT1	
394			
395		:ERROR 202	INCORRECT 'SKI' STATUS DURING DIAGNOSTIC MODE
396			
	003614 067630	EMT202	
	003616 071600	EHT1	
	003620 071724	EDT1	
	003622 072014	EFT1	
397			
398		:ERROR 203	INCORRECT 'DVC' STATUS DURING DIAGNOSTIC MODE
399			
	003624 067642	EMT203	
	003626 071600	EHT1	
	003630 071724	EDT1	
	003632 072014	EFT1	
400			
401		:ERROR 204	'VV' WAS NOT RESET BY MAINTENANCE UNIT READY
402			
	003634 067654	EMT204	
	003636 071600	EHT1	
	003640 071724	EDT1	
	003642 072014	EFT1	
403			
404		:ERROR 205	SELECTED DEVICE HAS A PERSISTENT 'SKI' ERROR
405			
	003644 067672	EMT205	
	003646 071600	EHT1	
	003650 071724	EDT1	

	003652	072014	EFT1	
406				
407			:ERROR 206	'LBC' DID NOT SET DURING DIAGNOSTIC MODE
408				
	003654	067702	EMT206	
	003656	071600	EHT1	
	003660	071724	EDT1	
	003662	072014	EFT1	
409				
410			:ERROR 207	UNEXPECTED LOSS OF 'MOL' - MEDIUM IS OFF LINE
411				
	003664	067712	EMT207	
	003666	071600	EHT1	
	003670	071724	EDT1	
	003672	072014	EFT1	
412				
413			:ERROR 210	UNEXPECTED LOSS OF VOLUME VALID - 'VV' = 0
414				
	003674	067724	EMT210	
	003676	071600	EHT1	
	003700	071724	EDT1	
	003702	072014	EFT1	
415				
416			:ERROR 211	UNEXPECTED MECHANICAL MOTION - 'PIP' = 1
417				
	003704	067732	EMT211	
	003706	071600	EHT1	
	003710	071724	EDT1	
	003712	072014	EFT1	
418				
419			:ERROR 212	UNEXPECTED DEVICE FAULT - 'DVC' = 1
420				
	003714	067746	EMT212	
	003716	071600	EHT1	
	003720	071724	EDT1	
	003722	072014	EFT1	
421				
422			:ERROR 213	UNEXPECTED SEEK INCOMPLETE ERROR - 'SKI' = 1
423				
	003724	067762	EMT213	
	003726	071600	EHT1	
	003730	071724	EDT1	
	003732	072014	EFT1	
424				
425			:ERROR 214	DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
426				
	003734	067772	EMT214	
	003736	071612	EHT2	
	003740	071734	EDT2	
	003742	072024	EFT2	

427			
428		:ERROR 215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
429	003744 070012	EMT215	
	003746 071612	EHT2	
	003750 071734	EDT2	
	003752 072024	EFT2	
430			
431		:ERROR 216	INCORRECT "IVC" STATUS
432	003754 070024	EMT216	
	003756 071600	EHT1	
	003760 071724	EDT1	
	003762 072014	EFT1	
433			
434		:ERROR 217	INCORRECT "IAE" STATUS
435	003764 070034	EMT217	
	003766 071600	EHT1	
	003770 071724	EDT1	
	003772 072014	EFT1	
436			
437		:ERROR 220	INCORRECT "WLE" STATUS
438	003774 070044	EMT220	
	003776 071600	EHT1	
	004000 071724	EDT1	
	004002 072014	EFT1	
439			
440		:ERROR 221	INCORRECT "OPI" STATUS
441	004004 070054	EMT221	
	004006 071600	EHT1	
	004010 071724	EDT1	
	004012 072014	EFT1	
442			
443		:ERROR 222	RM80 DID NOT DETECT RMR ERROR
444	004014 070064	EMT222	
	004016 071600	EHT1	
	004020 071724	EDT1	
	004022 072014	EFT1	
445			
446		:ERROR 223	RM80 DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
447	004024 070074	EMT223	
	004026 071636	EHT223	
	004030 071750	EDT223	
	004032 072040	EFT223	

448				
449			;ERROR 224	SSE NOT DETECTED DURING DATA TRANSFER
450				
	004034	070104	EMT224	
	004036	071600	EHT1	
	004040	071724	EDT1	
	004042	072014	EFT1	
451				
452			;ERROR 225	"SSEI" NOT RESET BY END OF TRACK DURING DATA TRANSFER
453				
	004044	070120	EMT225	
	004046	071600	EHT1	
	004050	071724	EDT1	
	004052	072014	EFT1	
454				
455			;ERROR 226	"SSEI" SHOULD BE SET AFTER DATA TRANSFER
456				
	004054	070140	EMT226	
	004056	071600	EHT1	
	004060	071724	EDT1	
	004062	072014	EFT1	
457				
458			;ERROR 227	SSE WAS DETECTED W/ SSEI SET
459				
	004064	070150	EMT227	
	004066	071600	EHT1	
	004070	071724	EDT1	
	004072	072014	EFT1	
460				
461			;ERROR 230	UNUSED
462				
	004074	070164	EMT230	
	004076	000000	0	
	004100	000000	0	
	004102	000000	0	
463				
464			;ERROR 231	UNUSED
465				
	004104	070166	EMT231	
	004106	000000	0	
	004110	000000	0	
	004112	000000	0	
466				
467			;ERROR 232	UNUSED
468				
	004114	070170	EMT232	
	004116	000000	0	
	004120	000000	0	
	004122	000000	0	

469

470			:ERROR 233	UNUSED
471	004124	070172	EMT233	
	004126	000000	0	
	004130	000000	0	
	004132	000000	0	
472				
473			:ERROR 234	UNUSED
474	004134	000000	0	
	004136	000000	0	
	004140	000000	0	
	004142	000000	0	
475				
476			:ERROR 235	UNUSED
477	004144	000000	0	
	004146	000000	0	
	004150	000000	0	
	004152	000000	0	
478				
479			:ERROR 236	UNUSED
480	004154	000000	0	
	004156	000000	0	
	004160	000000	0	
	004162	000000	0	
481				
482			:ERROR 237	UNUSED
483	004164	000000	0	
	004166	000000	0	
	004170	000000	0	
	004172	000000	0	
484				
485			:ERROR 240	UNUSED
486	004174	000000	0	
	004176	000000	0	
	004200	000000	0	
	004202	000000	0	
487				
488			:ERROR 241	UNUSED
489	004204	000000	0	
	004206	000000	0	
	004210	000000	0	
	004212	000000	0	
490				
491			:ERROR 242	UNUSED

492	004214	000000	0	
	004216	000000	0	
	004220	000000	0	
	004222	000000	0	
493				
494			:ERROR 243	UNUSED
495	004224	000000	0	
	004226	000000	0	
	004230	000000	0	
	004232	000000	0	
496				
497			:ERROR 244	UNUSED
498	004234	000000	0	
	004236	000000	0	
	004240	000000	0	
	004242	000000	0	
499				
500			:ERROR 245	UNUSED
501	004244	000000	0	
	004246	000000	0	
	004250	000000	0	
	004252	000000	0	
502				
503			:ERROR 246	"ATA" NOT RESET BY GO WHEN "ERR" = 0
504	004254	070220	EMT246	
	004256	071600	EHT1	
	004260	071724	EDT1	
	004262	072014	EFT1	
505				
506			:ERROR 247	"ATA" NOT RESET BY WRITING RMAS
507	004264	070230	EMT247	
	004266	071600	EHT1	
	004270	071724	EDT1	
	004272	072014	EFT1	
508				
509			:ERROR 250	"ATA" WAS RESET BY GO WHEN "ERR" = 1
510	004274	070242	EMT250	
	004276	071600	EHT1	
	004300	071724	EDT1	
	004302	072014	EFT1	
511				
512			:ERROR 251	PROGRAM INTERRUPT WAS NOT GENERATED
513				

004304	070256	EMT251
004306	071612	EHT2
004310	071734	EDT2
004312	072024	EFT2
514		
515	:ERROR	252 PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
516		
004314	070264	EMT252
004316	071612	EHT2
004320	071734	EDT2
004322	072024	EFT2
517		
518	:ERROR	253 OFFSET MODE WAS NOT RESET BY WRITING RMDC
519		
004324	070272	EMT253
004326	071600	EHT1
004330	071724	EDT1
004332	072014	EFT1
520		
521	:ERROR	254 INCORRECT "ILF" STATUS
522		
004334	070310	EMT254
004336	071600	EHT1
004340	071724	EDT1
004342	072014	EFT1
523		
524	:ERROR	255 INCORRECT "ATA" STATUS
525		
004344	070320	EMT255
004346	071600	EHT1
004350	071724	EDT1
004352	072014	EFT1
526		
527	:ERROR	256 INCORRECT "ILR" STATUS
528		
004354	070330	EMT256
004356	071650	EHT256
004360	071750	EDT223
004362	072040	EFT223
529		
530	:ERROR	257 INVALID IAE STATUS DURING SEARCH COMMAND
531		
004364	070340	EMT257
004366	071600	EHT1
004370	071724	EDT1
004372	072014	EFT1
532		
533	:ERROR	260 "IVC" WAS NOT DETECTED DURING SEARCH COMMAND
534		
004374	070352	EMT260

	004376 071600	EHT1	
	004400 071724	EDT1	
	004402 072014	EFT1	
535			
536			:ERROR 261 DRIVE EXECUTED SEARCH WITH ERROR SET
537			
	004404 070364	EMT261	
	004406 071600	EHT1	
	004410 071724	EDT1	
	004412 072014	EFT1	
538			
539			:ERROR 262 "LBC" ERROR NOT SET DURING DIAGNOSTIC MODE
540			
	004414 070404	EMT262	
	004416 071600	EHT1	
	004420 071724	EDT1	
	004422 072014	EFT1	
541			
542			:ERROR 263 "SKI" ERROR DURING SEARCH COMMAND
543			
	004424 070414	EMT263	
	004426 071600	EHT1	
	004430 071724	EDT1	
	004432 072014	EFT1	
544			
545			:ERROR 264 "IVC" ERROR DURING SEARCH - LOST VOLUME VALID
546			
	004434 070424	EMT264	
	004436 071600	EHT1	
	004440 071724	EDT1	
	004442 072014	EFT1	
547			
548			:ERROR 265 ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
549			
	004444 070444	EMT265	
	004446 071600	EHT1	
	004450 071724	EDT1	
	004452 072014	EFT1	
550			
551			:ERROR 266 DEVICE FAULT (DVC) DURING SEARCH
552			
	004454 070464	EMT266	
	004456 071600	EHT1	
	004460 071724	EDT1	
	004462 072014	EFT1	
553			
554			:ERROR 267 SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER
555			ADDRESS IS TOO LARGE
556			
	004464 070476	EMT267	

	004466 071600	EHT1	
	004470 071724	EDT1	
	004472 072014	EFT1	
557			
558		:ERROR 270	OPI ERROR DURING SEARCH BECAUSE MOL = 0
559			
	004474 070514	EMT270	
	004476 071600	EHT1	
	004500 071724	EDT1	
	004502 072014	EFT1	
560			
561		:ERROR 271	OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
562		:	DIDN'T DROP
563			
	004504 070530	EMT271	
	004506 071600	EHT1	
	004510 071724	EDT1	
	004512 072014	EFT1	
564			
565		:ERROR 272	LOST MOL DURING SEARCH, OPI IS NOT SET
566			
	004514 070546	EMT272	
	004516 071600	EHT1	
	004520 071724	EDT1	
	004522 072014	EFT1	
567			
568		:ERROR 273	PIP STIL SET AFTER SEARCH
569			
	004524 070564	EMT273	
	004526 071600	EHT1	
	004530 071724	EDT1	
	004532 072014	EFT1	
570			
571		:ERROR 274	PARITY ERROR OCCURRED WHILE WRITING REMOTE
572		:	REGISTERS BUT MXF DID NOT SET
573			
	004534 070602	EMT274	
	004536 071600	EHT1	
	004540 071724	EDT1	
	004542 072014	EFT1	
574			
575		:ERROR 275	MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
576		:	COMMAND STARTED
577			
	004544 070620	EMT275	
	004546 071600	EHT1	
	004550 071724	EDT1	
	004552 072014	EFT1	
578			
579		:ERROR 276	'OPI' ERROR DURING DATA TRANSFER BECAUSE 'MOL' WAS

580	:	ZERO	
581			
004554	070632	EMT276	
004556	071600	EHT1	
004560	071724	EDT1	
004562	072014	EFT1	
582			
583	:ERROR	277	'OPI' ERROR DURING DATA TRANSFER BECAUSE 1) ON
584	:		CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
585	:		3) RUN TIMED OUT
586			
004564	070646	EMT277	
004566	071600	EHT1	
004570	071724	EDT1	
004572	072014	EFT1	
587			
588	:ERROR	300	'IVC' ERROR DURING DATA TRANSFER BECAUSE VOLUME
589	:		WAS NOT VALID
590			
004574	070664	EMT300	
004576	071600	EHT1	
004600	071724	EDT1	
004602	072014	EFT1	
591			
592	:ERROR	301	ERRONEOUS 'IVC' ERROR DURING DATA TRANSFER - VOLUME
593	:		IS VALID
594			
004604	070704	EMT301	
004606	071600	EHT1	
004610	071724	EDT1	
004612	072014	EFT1	
595			
596	:ERROR	302	'ILR' ERROR DURING DATA TRANSFER
597			
004614	070726	EMT302	
004616	071600	EHT1	
004620	071724	EDT1	
004622	072014	EFT1	
598			
599	:ERROR	303	'ILF' ERROR DURING DATA TRANSFER
600			
004624	070740	EMT303	
004626	071600	EHT1	
004630	071724	EDT1	
004632	072014	EFT1	
601			
602	:ERROR	304	'RMR' ERROR DURING DATA TRANSFER
603			
004634	070752	EMT304	
004636	071600	EHT1	
004640	071724	EDT1	

	004642 072014	EFT1	
604			
605		:ERROR 305	INCORRECT "IAE" STATUS DURING DATA TRANSFER
606			
	004644 070764	EMT305	
	004646 071600	EHT1	
	004650 071724	EDT1	
	004652 072014	EFT1	
607			
608		:ERROR 306	"SKI" ERROR DURING DATA TRANSFER
609			
	004654 070776	EMT306	
	004656 071600	EHT1	
	004660 071724	EDT1	
	004662 072014	EFT1	
610			
611		:ERROR 307	DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
612			
	004664 071006	EMT307	
	004666 071600	EHT1	
	004670 071724	EDT1	
	004672 072014	EFT1	
613			
614		:ERROR 310	DEVICE FAULT DURING DATA TRANSFER
615			
	004674 071026	EMT310	
	004676 071600	EHT1	
	004700 071724	EDT1	
	004702 072014	EFT1	
616			
617		:ERROR 311	LOSS OF BIT CLOCK DURING DATA TRANSFER
618			
	004704 071040	EMT311	
	004706 071600	EHT1	
	004710 071724	EDT1	
	004712 072014	EFT1	
619			
620		:ERROR 312	LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
621			
	004714 071052	EMT312	
	004716 071600	EHT1	
	004720 071724	EDT1	
	004722 072014	EFT1	
622			
623		:ERROR 313	UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
624			
	004724 071064	EMT313	
	004726 071600	EHT1	
	004730 071724	EDT1	
	004732 072014	EFT1	

625				
626			:ERROR 314	DRIVE TIMING ERROR DURING DATA TRANSFER
627				
	004734	071104	EMT314	
	004736	071600	EHT1	
	004740	071724	EDT1	
	004742	072014	EFT1	
628				
629			:ERROR 315	WRITE LOCK ERROR
630				
	004744	071116	EMT315	
	004746	071600	EHT1	
	004750	071724	EDT1	
	004752	072014	EFT1	
631				
632			:ERROR 316	ERRONEOUS WRITE LOCK ERROR
633				
	004754	071130	EMT316	
	004756	071600	EHT1	
	004760	071724	EDT1	
	004762	072014	EFT1	
634				
635			:ERROR 317	HEADER CRC ERROR DURING DATA TRANSFER
636				
	004764	071142	EMT317	
	004766	071600	EHT1	
	004770	071724	EDT1	
	004772	072014	EFT1	
637				
638			:ERROR 320	FORMAT ERROR DURING DATA TRANSFER
639				
	004774	071152	EMT320	
	004776	071600	EHT1	
	005000	071724	EDT1	
	005002	072014	EFT1	
640				
641			:ERROR 321	HEADER COMPARE ERROR DURING DATA TRANSFER
642				
	005004	071162	EMT321	
	005006	071600	EHT1	
	005010	071724	EDT1	
	005012	072014	EFT1	
643				
644			:ERROR 322	HEADER ERRORS SHOULD NOT BE SET
645				
	005014	071172	EMT322	
	005016	071600	EHT1	
	005020	071724	EDT1	
	005022	072014	EFT1	

646				
647			:ERROR 323	DATA CHECK ERROR DURING DATA TRANSFER
648				
	005024	071200	EMT323	
	005026	071600	EHT1	
	005030	071724	EDT1	
	005032	072014	EFT1	
649				
650			:ERROR 324	CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
651				
	005034	071210	EMT324	
	005036	071600	EHT1	
	005040	071724	EDT1	
	005042	072014	EFT1	
652				
653			:ERROR 325	UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
654				
	005044	071222	EMT325	
	005046	071600	EHT1	
	005050	071724	EDT1	
	005052	072014	EFT1	
655				
656			:ERROR 326	DATA PARITY ERROR DURING READ COMMAND
657				
	005054	071234	EMT326	
	005056	071600	EHT1	
	005060	071724	EDT1	
	005062	072014	EFT1	
658				
659			:ERROR 327	OFFSET MODE NOT RESET BY WRITE COMMAND
660				
	005064	071252	EMT327	
	005066	071600	EHT1	
	005070	071724	EDT1	
	005072	072014	EFT1	
661				
662			:ERROR 330	DATA PARITY ERROR DURING WRITE COMMAND
663				
	005074	071264	EMT330	
	005076	071600	EHT1	
	005100	071724	EDT1	
	005102	072014	EFT1	
664				
665			:ERROR 331	WRITE CLOCK FAILURE DURING WRITE COMMAND
666				
	005104	071274	EMT331	
	005106	071600	EHT1	
	005110	071724	EDT1	
	005112	072014	EFT1	

667

668		:ERROR 332	DATA LATE ERROR DURING DATA TRANSFER
669			EMT332
	005114 071306		EHT1
	005116 071600		EDT1
	005120 071724		EFT1
	005122 072014		
670		:ERROR 333	PIP STIL SET AFTER DATA TRANSFER - SKI = 0
671			EMT333
672			EHT1
	005124 071320		EDT1
	005126 071600		EFT1
	005130 071724		
	005132 072014		
673		:ERROR 334	LOST MOL DURING DATA TRANSFER - OPI = 0
674			EMT334
675			EHT1
	005134 071336		EDT1
	005136 071600		EFT1
	005140 071724		
	005142 072014		
676		:ERROR 335	LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
677			EMT335
678			EHT1
	005144 071354		EDT1
	005146 071600		EFT1
	005150 071724		
	005152 072014		
679		:ERROR 336	DATA READ DOES NOT COMPARE WITH DATA WRITTEN
680			EMT336
681			EHT336
	005154 071374		EDT336
	005156 071662		EFT336
	005160 071760		
	005162 072050		
682		:ERROR 337	WRITE CHECK ERROR NOT DETECTED
683			EMT337
684			EHT337
	005164 071404		EDT337
	005166 071674		EFT337
	005170 071770		
	005172 072060		
685		:ERROR 340	WRITE CHECK ERROR AT UNEXPECTED ADDRESS
686			EMT340
687			EHT336
	005174 071414		EDT336
	005176 071662		EFT336
	005200 071760		
	005202 072050		
688		:ERROR 341	INCORRECT DATA DURING WRITE CHECK ERROR
689			

690	005204 071426	EMT341	
	005206 071662	EHT336	
	005210 071760	EDT336	
	005212 072050	EFT336	
691			
692		:ERROR 342	"IVC" ERROR NOT DETECTED DURING DATA TRANSFER
693			
	005214 071434	EMT342	
	005216 071600	EHT1	
	005220 071724	EDT1	
	005222 072014	EFT1	
694			
695		:ERROR 343	"FER" NOT DETECTED DURING DATA TRANSFER
696			
	005224 071446	EMT343	
	005226 071600	EHT1	
	005230 071724	EDT1	
	005232 072014	EFT1	
697			
698		:ERROR 344	"HCE" NOT DETECTED DURING DATA TRANSFER
699			
	005234 071460	EMT344	
	005236 071706	EHT344	
	005240 072000	EDT344	
	005242 072070	EFT344	
700			
701		:ERROR 345	"BSE" NOT DETECTED DURING DATA TRANSFER
702			
	005244 071472	EMT345	
	005246 071600	EHT1	
	005250 071724	EDT1	
	005252 072014	EFT1	
703			
704		:ERROR 346	HEADER ERROR WAS DETECTED W/ HCI SET
705			
	005254 071502	EMT346	
	005256 071600	EHT1	
	005260 071724	EDT1	
	005262 072014	EFT1	
706			
707		:ERROR 347	DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
708			
	005264 071516	EMT347	
	005266 071600	EHT1	
	005270 071724	EDT1	
	005272 072014	EFT1	
709			
710		:ERROR 350	LOST VOLUME VALID DURING SEARCH - "IVC" = 0
711			

005274	071530	EMT350
005276	071600	EHT1
005300	071724	EDT1
005302	072014	EFT1
712		
713		
714		
005304	071546	EMT351
005306	071600	EHT1
005310	071724	EDT1
005312	072014	EFT1
715		
716		
717		
005314	071556	EMT352
005316	000000	0
005320	000000	0
005322	000000	0
718		
719		
720		
005324	071562	EMT353
005326	071720	EHT353
005330	072012	EDT353
005332	072100	EFT353
721		
722		
723		
005334	071572	EMT354
005336	071600	EHT1
005340	071724	EDT1
005342	072014	EFT1
724		
725		

;ERROR 351 'ATA' DID NOT SET DURING SEARCH

;ERROR 352 PROGRAM TIMEOUT WHILE TESTING RMLA

;ERROR 353 LOOK AHEAD TEST FAILS

;ERROR 354 BSE SHOULD NOT BE SET

;PUT ERROR TABLE HERE

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

.SBTTL ERROR TABLE USAGE

:THE ERROR TABLE ABOVE CONSISTS OF FOUR WORD ENTRIES FOR EACH ERROR
:NUMBER, I.E.,

:
: EMT - ERROR MESSAGE TABLE ADDRESS
: EHT - ERROR HEADER TABLE ADDRESS
: EDT - ERROR DATA TABLE ADDRESS
: EFT - ERROR FORMAT TABLE ADDRESS
:

:THE EMT ENTRY IS THE ADDRESS OF THE TABLE OF ERROR MESSAGE STRINGS
:FOR THE PARTICULAR ERROR. EACH ERROR MESSAGE TABLE LISTS THE ADDRESS
: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.
:


```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3      005344 011600      BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
4      005346 005740      TST      -(R0)      ;ADJUST PC -2
5      005350 022626      CMP      (SP)+,(SP)+    ;RESTORE STACK POINTER
6      005352 104401 005360  TYPE      ,65$      ;:TYPE ASCIZ STRING
        005356 000417      BR       64$      ;:GET OVER THE ASCIZ
        ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
        64$:
7      005416 010046      MOV      R0,-(SP)      ;SETUP FOR TYPING OUT PC
8      005420 104402      TYPOC
9      005422 000240      NOP
        ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
        ;TO STOP ON UNEXPECTED TIMEOUT.
10
11
12     .SBTTL  START OF PROGRAM
13
14     005424 012737 177777 001330  START1: MOV      #-1,CHGADR    ;CHANGE RH/RM BUS ADDRESS
15     005432 000402      BR       START2
16
17     005434 005037 001330  START:  CLR      CHGADR    ;NO CHANGE IN ADDRESS
18     005440 000240  START2:  NOP
19     005442 005227 000000  INC      #0      ;TTY LOOP, WAIT FOR INCREMENT
20     005446 001375      BNE     .-4      ;OF WORD
21     005450 000005      RESET    ;RESET THE WORLD
22
23     .SBTTL  INITIALIZE THE COMMON TAGS
        ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
        MOV      #SCMTAG,R6    ;:FIRST LOCATION TO BE CLEARED
        CLR      (R6)+        ;:CLEAR MEMORY LOCATION
        CMP      #SWR,R6      ;:DONE?
        BNE     .-6          ;:LOOP BACK IF NO
        MOV      #STACK,SP    ;:SETUP THE STACK POINTER
        ;:INITIALIZE A FEW VECTORS
        MOV      #SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
        MOV      #340,@#IOTVEC+2 ;:LEVEL 7
        MOV      #ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
        MOV      #340,@#EMTVEC+2 ;:LEVEL 7
        MOV      #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
        MOV      #340,@#TRAPVEC+2 ;:LEVEL 7
        MOV      #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
        MOV      #340,@#PWRVEC+2 ;:LEVEL 7
        MOV      $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
        CLR      $TIMES      ;:INITIALIZE NUMBER OF ITERATIONS
        CLR      $ESCAPE     ;:CLEAR THE ESCAPE ON ERROR ADDRESS
        MOVB    #1,$ERMAX    ;:ALLOW ONE ERROR PER TEST
        MOV      #,$LPADR     ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
        MOV      #,$LPERR     ;:SETUP THE ERROR LOOP ADDRESS
        ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
        ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
        MOV      @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
        MOV      #64$,@#ERRVEC ;:SET UP ERROR VECTOR
        MOV      #DSWR,SWR    ;:SETUP FOR A HARDWARE SWICH REGISTER
        MOV      #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
        CMP      #-1,@SWR    ;:TRY TO REFERENCE HARDWARE SWR
        BNE     66$         ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
        ;:AND THE HARDWARE SWR IS NOT = -1
        BR      65$         ;:BRANCH IF NO TIMEOUT
        005612 013746 000004      MOV      @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
        005616 012737 005652 000004  MOV      #64$,@#ERRVEC ;:SET UP ERROR VECTOR
        005624 012737 177570 001154  MOV      #DSWR,SWR    ;:SETUP FOR A HARDWARE SWICH REGISTER
        005632 012737 177570 001156  MOV      #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
        005640 022777 177777 173306  CMP      #-1,@SWR    ;:TRY TO REFERENCE HARDWARE SWR
        005646 001012      BNE     66$         ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
        ;:AND THE HARDWARE SWR IS NOT = -1
        005650 000403      BR      65$         ;:BRANCH IF NO TIMEOUT
    
```

```

005652 012716 005660      64$:  MOV    #65$, (SP)      ;;SET UP FOR TRAP RETURN
005656 000002
005660 012737 000176 001154 65$:  MOV    #SWREG, SWR      ;;POINT TO SOFTWARE SWR
005666 012737 000174 001156      MOV    #DISPREG, DISPLAY
005674 012637 000004      66$:  MOV    (SP)+, @#ERRVEC  ;;RESTORE ERROR VECTOR

005700 005037 001230      CLR    $PASS           ;;CLEAR PASS COUNT
005704 132737 000200 001243      BITB  #APTSIZE, $ENVM  ;;TEST USER SIZE UNDER APT
005712 001403      BEQ    67$            ;;YES, USE NON-APT SWITCH
005714 012737 001244 001154      MOV    #$$SWREG, SWR  ;;NO, USE APT SWITCH REGISTER
005722
24 005722 012737 005344 000004 67$:  ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 005722 012737 005344 000004      MOV    #BADTMO, ERRVEC ;SETUP FOR UNEXPECTED TIMEOUT
26 005730 012737 000300 000006      MOV    #PR6, ERRVEC+2 ;LEVEL 6
27
28      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005736 005227 177777      INC    #-1            ;;FIRST TIME?
005742 001033      BNE    68$            ;;BRANCH IF NO
005744 022737 031402 000042      CMP    #SENDAD, @#42  ;;ACT-11?
005752 001427      BEQ    68$            ;;BRANCH IF YES
005754 104401 005762      TYPE  ,69$           ;;TYPE ASCIZ STRING
005760 000424      BR     68$            ;;GET OVER THE ASCIZ
      ;;69$: .ASCIZ <CRLF>@CZRNEAO - RM80 FUNCTIONAL TEST, PT 2@<CRLF>
006032      68$:
      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
006032 005737 000042      TST    @#42           ;;ARE WE RUNNING UNDER XXDP/ACT?
006036 001012      BNE    70$            ;;BRANCH IF YES
006040 123727 001242 000001      CMPB  $ENV, #1        ;;ARE WE RUNNING UNDER APT?
006046 001406      BEQ    70$            ;;BRANCH IF YES
006050 023727 001154 000176      CMP    SWR, #SWREG    ;;SOFTWARE SWITCH REG SELECTED?
006056 001005      BNE    71$            ;;BRANCH IF NO
006060 104407      GTSWR                ;;GET SOFT-SWR SETTINGS
006062 000403      BR     71$
006064 112737 000001 001150 70$:  MOVB   #1, $AUTOB     ;;SET AUTO-MODE INDICATOR
006072      71$:

29      ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
30      ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
31
32
33 006072 005037 001332      CLR    XXDP           ;CLEAR 'XXDP' LOAD DEVICE STORAGE
34 006076 122737 000016 000041      CMPB  #16, @#41      ;LOADED FROM AN RM80 ?
35 006104 001121      BNE    3$            ;BR IF NOT
36 006106 013737 000040 001332      MOV    @#40, XXDP    ;GET DEVICE INDICATOR AND NUMBER
37 006114 122737 000007 001332      CMPB  #7, XXDP       ;IS IT A VALID NUMBER ?
38 006122 103002      BHIS  1$            ;YES
39 006124 105037 001332      CLRB  XXDP           ;NO, DEFAULT TO DRIVE 0
40 006130 005737 000042      1$:  TST    @#42           ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
41 006134 001425      BEQ    2$            ;BR IF NEITHER
42 006136 104401 006144      TYPE  ,73$           ;;TYPE ASCIZ STRING
006142 000412      BR     72$           ;;GET OVER THE ASCIZ
      ;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
      72$:
43 006170 005046      CLR    -(SP)         ;CLEAR WORD ON STACK
44 006172 113716 001332      MOVB  XXDP, (SP)     ;GET DRIVE ADDRESS
45 006176 104403      TYPOS                ;TYPE THE ADDRESS
46 006200 001      .BYTE 1             ;ONLY 1 CHARACTER

```

```

47 006201      000          .BYTE 0          :SUPRESS LEADING ZEROS
48 006202 104401 001217    TYPE  ,SCRLF      :CR-LF
49 006206 000460          BR      3$        :GET NUMBER OF DRIVES
50
51 006210 005227 177777    2$: INC      #-1      :FIRST TIME THRU HERE ?
52 006214 001055          BNE     3$        :NO
53 006216 104401 006224    TYPE  ,75$       :TYPE ASCIZ STRING
    006222 000410          BR      74$       :GET OVER THE ASCIZ
    :75$: .ASCIZ <CRLF>/TO TEST DRIVE /
    74$:
54 006244 005046          CLR     -(SP)     :CLEAR WORD ON STACK
55 006246 113716 001332    MOVB   XXDP,(SP) :GET DRIVE ADDRESS
56 006252 104403          TYPOS          :TYPE DRIVE ADDRESS
57 006254      001          .BYTE 1          :ONLY 1 CHARACTER
58 006255      000          .BYTE 0          :SUPRESS LEADING ZEROS
59 006256 104401 006264    TYPE  ,76$       :TYPE ASCIZ STRING
    006262 000432          BR      3$        :GET OVER THE ASCIZ
    :76$: .ASCIZ /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
    3$:
63
64 :CHECK FOR AUTO MODE OR STANDALONE MODE
65 006350 005737 000042    TST    @#42      :RUNNING IN AUTO MODE ?
66 006354 001537          BEQ    STANDALONE :BR IF NO
67 006356 012737 000377 001300 MOV     #377,$DEV :SET DEVICE MAP FOR ALL DRIVES
68
69 :PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
70 006364          XSIZ:
71 006364 132737 000200 001243 BITB   #BIT7,$ENVM :SIZING ALLOWED ?
72 006372 001124          BNE    12$       :NO
73
74 006374 005001          CLR     R1        :START FROM DRIVE 0
75 006376 013700 001276    MOV    $BASE,R0  :LOAD THE BASE ADDRESS
76 006402 104401 064504    TYPE  ,SYSTAT    :TYPE 'UNIT STATUS:'
77
78 006406 136137 065004 001300 1$: BITB   ATNTBL(R1),$DEV :IS DEVICE PRESENT IN MAP ?
79 006414 001507          BEQ    11$       :BR IF NO
80 006416 104401 001217    TYPE  ,SCRLF      :CR-LF
81 006422 010146          MOV    R1,-(SP)  :SAVE R1 FOR TYPEOUT
    006424 104403          TYPOS          :GO TYPE--OCTAL ASCII
    006426      002          .BYTE 2          :TYPE 2 DIGIT(S)
    006427      000          .BYTE 0          :SUPPRESS LEADING ZEROS
82 006430 104401 064676    TYPE  ,BLNKS4     :TYPE 4 BLANKS
83
84 006434 012760 000040 000010 MOV    #CLR, RMCS2(R0) :CLEAR MASS BUS
85 006442 010160 000010 MOV    R1, RMCS2(R0) :LOAD THE DRIVE ADDRESS
86 006446 005760 000012 TST    RMD5(R0)    :ACCESS DRIVE REGISTER
87 006452 032760 010000 000010 BIT    #NED, RMCS2(R0) :IS DRIVE PRESENT ?
88 006460 001027          BNE    3$        :BR IF NO
89 006462 032760 004000 000000 BIT    #DVA, RMCS1(R0) :IS DRIVE AVAILABLE ?
90 006470 001426          BEQ    4$        :BR IF NO
91 006472 012737 064522 006632 MOV    #SRM80, 10$ :ASSUME RM80 DEVICE
92 006500 022760 020026 000026 CMP    #20026, RMDT(R0) :SINGLE PORT RM80 ?
93 006506 001407          BEQ    2$        :BR IF YES
94 006510 022760 024026 000026 CMP    #24026, RMDT(R0) :DUAL PORT RM80 ?
95 006516 001403          BEQ    2$        :BR IF YES
96 006520 104401 064527    TYPE  ,NOTRM     :DRIVE NOT AN RM80
97 006524 000412          BR      5$        :CHECK NEXT DRIVE
    
```

98	006526	032760	010000	000012	2\$:	BIT	#MOL,RMDS(R0)	:IS MEDIUM ON LINE ?
99	006534	001412				BEQ	6\$:BR IF NO
100	006536	000417				BR	7\$	
101								
102	006540	104401	064561		3\$:	TYPE	,NOTPRS	:DRIVE NOT PRESENT
103	006544	000402				BR	5\$:CHECK NEXT DRIVE
104								
105	006546	104401	064576		4\$:	TYPE	,NOTAVL	:DRIVE NOT AVAILABLE
106	006552	146137	065004	001300	5\$:	BICB	ATNTBL(R1),SDEVM	:CLEAR DEVICE FROM BIT MAP
107	006560	000425				BR	11\$:CHECK NEXT DRIVE
108								
109	006562	104401	064615		6\$:	TYPE	,UNTOFF	:DRIVE OFFLINE
110	006566	146137	065004	001300		BICB	ATNTBL(R1),SDEVM	:CLEAR DEVICE FROM BIT MAP
111	006574	000413				BR	9\$:PRINT DRIVE TYPE
112								
113	006576	005737	001332		7\$:	TST	XXDP	:LOADED FROM RM80 ?
114	006602	001406				BEQ	8\$:NO
115	006604	123701	001332			CMPB	XXDP,R1	:IS THIS THE DRIVE ?
116	006610	001003				BNE	8\$:BR IF NO
117	006612	104401	064544			TYPE	,LODEV	:DRIVE IS LOAD DEVICE
118	006616	000755				BR	5\$	
119								
120	006620	104401	064626		8\$:	TYPE	,UNTON	:DRIVE ONLINE
121	006624	104401	064700		9\$:	TYPE	,BLNKS2	:TYPE 2 BLANKS
122	006630	104401				TYPE		:PRINT DRIVE TYPE
123	006632	000000			10\$:	.WORD	0	:MESSAGE ADDRESS HERE
124								
125	006634	005201			11\$:	INC	R1	:INCREMENT THE DRIVE ADDRESS
126	006636	020127	000007			CMP	R1,#7	:ALL DRIVES ARE CHECKED ?
127	006642	003661				BLE	1\$:BRANCH IF NOT
128								
129	006644	104401	001217		12\$:	TYPE	,\$CRLF	:CR-LF
130	006650	000137	007332			JMP	CMNSTART	:JUMP TO COMMON START

```

1          .SBTTL  STANDALONE INPUT ROUTINES
2
3          STANDALONE:
4 006654   004737  061330      JSR      PC,$TKINT      ;INITIALIZE CONSOLE
5
6 006660   005227  177777      INC      #-1           ;FIRST TIME THRU HERE ?
7 006664   001023                BNE      2$           ;BR IF NO
8
9          ;SEE IF OPERATOR WANTS HELP TEXT
10
11 006666   104401  063550      TYPE     ,MSHELP      ;WANT HELP ?
12 006672   104411                RDCHR                    ;GET RESPONSE
13 006674   012637  001176      MOV      (SP)+,$TMP1  ;SAVE AND ECHO RESPONSE
14 006700   123727  001176  000131  CMPB     $TMP1,#'Y    ;WAS IT A YES RESPONSE ?
15 006706   001005                BNE      1$           ;NO
16 006710   104401  001176      TYPE     , $TMP1      ;TYPE 'Y'
17 006714   104401  101716      TYPE     ,HELP        ;YES - TYPE HELP TEXT
18 006720   000414                BR       3$           ;
19 006722   104401  064672  1$:      TYPE     ,N           ;TYPE 'N'
20 006726   104401  001217      TYPE     , $CRLF      ;CR-LF
21 006732   000407                BR       3$           ;
22
23          ;SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
24 006734                2$:
25 006734   005737  001330      TST      CHGADR        ;CHANGE RH/RM BUS ADDRESS ?
26 006740   001457                BEQ      7$           ;BR IF NO
27 006742   005037  001330      CLR      CHGADR        ;NO CHANGE NEXT TIME
28 006746   104401  001217      TYPE     , $CRLF      ;CR-LF
29
30          ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
31 006752                3$:
32 006752   104401  064161      TYPE     ,CNSL01      ;TYPE CURRENT BUS ADDRESS
33 006756   013746  001276      MOV      $BASE,-(SP)  ;:SAVE $BASE FOR TYPEOUT
34 006762   104402                TYPOC                    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
35 006764   104401  064700      TYPE     ,BLNKS2      ;TYPE 2 BLANKS
36 006770   104413                RDOCT                    ;GET NEW BUS ADDRESS
37 006772   012637  001176      MOV      (SP)+,$TMP1  ;CARRIAGE RETURN ?
38 006776   001412                BEQ      5$           ;YES-SKIP TO NEXT ENTRY
39 007000   022737  160000  001176      CMP      #160000,$TMP1 ;BASE ADDRESS IN I/O PAGE ?
40 007010   104401  064171                BLOS     4$           ;YES
41 007014   000756                TYPE     ,CNSL02      ;TYPE WARNING MESSAGE
42 007016   013737  001176  001276  4$:      BR       3$           ;TRY AGAIN
43
44 007024   104401  064233                MOV      $TMP1,$BASE  ;STORE NEW BUS ADDRESS
45 007030   005046                5$:      TYPE     ,CNSL03
46 007032   113716  001272      CLR      -(SP)
47 007036   104402                MOV      $VECT1,(SP)  ;GET CURRENT VECTOR ADDRESS
48 007040   104401  064700      TYPE     ,BLNKS2      ;TYPE 2 BLANKS
49 007044   104413                RDOCT                    ;GET NEW VECTOR ADDRESS
50 007046   012637  001176      MOV      (SP)+,$TMP1  ;CARRIAGE RETURN?
51 007052   001412                BEQ      7$           ;YES-SKIP TO NEXT ENTRY
52 007054   022737  001000  001176      CMP      #1000,$TMP1  ;VECTOR ADDRESS < 1000 ?
53 007062   101003                BHI      6$           ;YES!!
54 007064   104401  064242                TYPE     ,CNSL04      ;TYPE WARNING MESSAGE
55 007070   000755                BR       5$           ;RETRY
56 007072   113737  001176  001272  6$:      MOV      $TMP1,$VECT1 ;STORE NEW VECTOR ADDRESS
    
```

```

57
58          ;DIALOGUE TO INPUT DEVICE NUMBERS
59 007100 005227 177777      7$: INC # -1          ;FIRST TIME THRU ?
60 007104 001002              BNE 8$              ;BR IF NO
61 007106 104401 064276      TYPE ,CNSL07        ;TYPE INPUT INSTRUCTIONS
62 007112 104401 001217      TYPE ,SCRLF         ;CR-LF
63 007116 005037 001300      9$: CLR $DEVM         ;CLEAR DEVICE MAP
64 007122 104401 064462      TYPE ,MSDRVS        ;TYPE 'DRIVE(S): '
65 007126 104411              RDCHR
66 007130 012637 001176      MOV (SP)+,$TMP1      ;GET RESPONSE
67 007134 023727 001176 000101 CMP $TMP1,#'A      ;IS INPUT 'A' ?
68 007142 001007              BNE 10$           ;NO
69 007144 104401 063534      TYPE ,ALL          ;YES, TYPE 'ALL' AND GO
70 007150 012737 000377 001300 MOV #377,$DEVM     ;SET DEVICE MAP FOR ALL DRIVES
71 007156 000137 006364      JMP XSIZ          ;AUTO SIZE.
72
73 007162 023727 001176 000015 10$: CMP $TMP1,#CR      ;CARRIAGE RETURN ?
74 007170 001436              BEQ 12$           ;YES
75 007172 104401 001176      TYPE $TMP1         ;ECHO RESPONSE
76 007176 023727 001176 000060 CMP $TMP1,#'0      ;NUMBER < 0 ?
77 007204 002430              BLT 12$           ;YES
78 007206 023727 001176 000067 CMP $TMP1,#'7      ;NUMBER > 7 ?
79 007214 003427              BLE 13$           ;NO
80 007216 000423              BR 12$           ;ILLEGAL INPUT
81
82 007220 104411              11$: RDCHR
83 007222 012637 001176      MOV (SP)+,$TMP1      ;GET RESPONSE
84 007226 023727 001176 000015 CMP $TMP1,#CR      ;CARRIAGE RETURN ?
85 007234 001432              BEQ 14$           ;YES
86 007236 104401 063545      TYPE ,COMMA        ;TYPE ','
87 007242 104401 001176      TYPE $TMP1         ;ECHO RESPONSE
88 007246 023727 001176 000060 CMP $TMP1,#'0      ;NUMBER < 0 ?
89 007254 002404              BLT 12$           ;YES
90 007256 023727 001176 000067 CMP $TMP1,#'7      ;NUMBER > 7 ?
91 007264 003403              BLE 13$           ;NO
92 007266 104401 064440      12$: TYPE ,CNSL08 ;TYPE '' ?ILLEGAL INPUT''
93 007272 000711              BR 9$            ;RETRY
94
95 007274 013701 001176      13$: MOV $TMP1,R1      ;R1 = DRIVE NUMBER
96 007300 042701 177770      BIC #'C7,R1
97 007304 156137 065004 001300 BISB ATNTBL(R1),$DEVM ;SET DEVICE IN MAP
98 007312 122737 000377 001300 CMPB #377,$DEVM    ;DONE ?
99 007320 101337              BHI 11$           ;NO
100 007322 104401 001217      14$: TYPE ,SCRLF   ;CR-LF
101 007326 000137 006364      JMP XSIZ          ;GO SIZE DEVICES
    
```

```

1          :ASSEMBLE TEST QUE FROM DEVICE MAP
2 007332  CMNSTART:
3 007332  104401 064636      TYPE      ,DRIVES      :TYPE 'DRIVE(S) TO BE TESTED'
4 007336  013700 001300      MOV      $DEVM,R0    :RO = DEVICE MAP
5 007342  001004      BNE      1$          :BR IF DRIVES TO TEST
6 007344  104401 063545      TYPE      ,COMMA      :TYPE ' '
7 007350  104401 064665      TYPE      ,NONE       :TYPE 'NONE'
8 007354  012701 001470  1$:  MOV      #TSTQUE+2,R1  :R1 = ADDRESS OF FIRST ENTRY IN QUE
9 007360  010137 001466      MOV      R1,TSTQUE   :INITIALIZE ENTRY POINTER
10 007364  012702 000001      MOV      #1,R2       :R2 = DEVICE POINTER
11 007370  005003      CLR      R3          :R3 = DEVICE NUMBER
12 007372  030200  2$:  BIT      R2,R0       :IS THIS DEVICE IN MAP ?
13 007374  001413      BEQ      3$          :NO !!
14 007376  104401 063545      TYPE      ,COMMA      :TYPE ' '
15 007402  010311      MOV      R3,(R1)     :YES - ENTER DEVICE NUMBER IN QUE
16 007404  010346      MOV      R3,-(SP)    :SAVE R3 FOR TYPEOUT
    007406  104403      TYPOS     :GO TYPE--OCTAL ASCII
    007410  001      .BYTE 1      :TYPE 1 DIGIT(S)
    007411  000      .BYTE 0      :SUPPRESS LEADING ZEROS
17 007412  116361 065004 000001  MOVB     ATNTBL(R3),1(R1) :ENTER ATTENTION BIT IN QUE
18 007420  062701 000002      ADD      #2,R1       :ADVANCE ENTRY POINTER
19 007424  006302  3$:  ASL      R2          :ADVANCE DEVICE POINTER
20 007426  105702      TSTB     R2          :DONE ALL DEVICES ?
21 007430  001402      BEQ      4$          :YES
22 007432  005203      INC      R3          :ADVANCE DEVICE NUMBER
23 007434  000756      BR       2$          :ENTER NEXT DEVICE
24 007436  005011  4$:  CLR      (R1)        :TERMINATE TEST QUE
25 007440  104401 001217      TYPE      ,SCRLF     :CR-LF
26
27          :SIZE FOR CLOCK
28 007444  004737 036234      JSR      PC,SIZCLK   :SEE IF CLOCK PRESENT
29 007450  000425      BR       6$          :YES - CLOCK IS PRESENT
30 007452  104401 007460      TYPE      ,65$       :TYPE ASCIZ STRING
    007456  000413      BR       64$        :GET OVER THE ASCIZ
    65$:  .ASCIZ <CRLF>/NO 'L' OR 'P' CLOCK/
    64$:
31 007506  005737 000042      TST      @#42        :ANY MONITOR PRESENT ?
32 007512  001002      BNE      5$          :BR IF YES
33 007514  000137 005434      JMP      START       :JUMP TO START
34 007520  000137 031372  5$:  JMP      $GET42      :RETURN CONTROL TO MONITOR
35 007524
36
37 007524  000240  6$:  NOP
38 007526  105737 001300  READY: TSTB     $DEVM       :READY TO START TEST
39 007532  001007      BNE      2$          :ANY DRIVES IN MAP ?
40 007534  005737 000042      TST      @#42        :BR IF YES
41 007540  001002      BNE      1$          :ANY MONITOR PRESENT ?
42 007542  000137 005434      JMP      START       :BR IF YES
43 007546  000137 031372  1$:  JMP      $GET42      :JUMP TO START
44
45 007552  105037 001116  2$:  CLRB     $TSTNM      :RETURN CONTROL TO MONITOR
46 007556  005037 001206      CLR      $TIMES      :RESET TEST NUMBER
47 007562  005037 001326      CLR      CTLFG       :INITIALIZE NUMBER OF ITERATIONS
48 007566  005037 001512      CLR      MEDENB      :CLEAR CONTROL-C FLAG
49 007572  004737 061330      JSR      PC,$TKINT   :CLEAR MEDIA ENABLE
50 007576  012746 000240      MOV      #PR5,-(SP)  :INITIALIZE TTY
    007602  012746 007610      MOV      #64$,-(SP) :PUT NEW PS ON STACK
    :PUT NEW PC ON STACK
    
```

```

007606 000002          RTI          ;;POP NEW PC AND PS
007610
51 007610 117737 171652 001234 64$: MOVB   @TSTQUE,$UNIT  ;LOAD DRIVE NUMBER
52
53          ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND SET LAST TRACK ADDRESS
54 007616 013700 001276          MOV   $BASE,R0      ;R0 = UNIBUS ADDRESS
55 007622 012760 000040 000010  MOV   #CLR, RMCS2(R0) ;CLEAR MASSBUS
56 007630 113760 001234 000010  MOVB  $UNIT, RMCS2(R0) ;SELECT DEVICE UNDER TEST
57 007636 012737 006400 001334  MOV   #TAB!TA4!TA1,LSTRK ;SET LAST TRACK = 13.
58
59          ;TYPE DRIVE NUMBER TO BE TESTED($UNIT)
60 007644 104401 001217          TYPE   ,SCRLF      ;CR-LF
61 007650 104401 064476          TYPE   ,MSGDRV     ;TYPE 'DRIVE'
62 007654 013746 001234          MOV   $UNIT,-(SP)  ;SAVE $UNIT FOR TYPEOUT
                                ;TYPE DRIVE NUMBER
                                ;GO TYPE--OCTAL ASCII
007660 104403          TYPOS
007662      002          .BYTE  2      ;TYPE 2 DIGIT(S)
007663      000          .BYTE  0      ;SUPPRESS LEADING ZEROS
63 007664 005004          CLR   R4          ;THESE TWO LOOPS ARE ADDED TO
64 007666 005304          DEC   R4          ;WAIT FOR TTY
65 007670 001376          BNE   .-2
66 007672 005304          DEC   R4
67 007674 001376          BNE   .-2
    
```


1
2

```

*****
:*TEST 1          CONTROLLER ACCESS TEST
*****
TST1:
    
```

007676	000004			SCOPE	:SCOPE CALL
007700	000240			NOP	:START OF TEST
007702	012706	001100		MOV #STACK,SP	:INITIALIZE STACK POINTER
007706	013700	001276		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
007712	013701	001466		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
007716	012737	000001	001226	MOV #1,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
3					
4	007724	005001		CLR R1	
5	007726	013746	000004	MOV ERRVEC,-(SP)	::PUSH ERRVEC ON STACK
	007732	013746	000006	MOV ERRVEC+2,-(SP)	::PUSH ERRVEC+2 ON STACK
6	007736	012737	010040	MOV #1\$,ERRVEC	
7	007744	012737	000300	MOV #PR6,ERRVEC+2	
8					
9	007752	110160	000001	MOVB R1,RMCS1+1(R0)	:MOVE HI BYTE TO RMCS1
10	007756	010160	000002	MOV R1,RMWC(R0)	:MOVE WORD COUNT REGISTER
11	007762	016002	000002	MOV RMWC(R0),R2	
12	007766	010160	000004	MOV R1,RMBA(R0)	:MOVE BUS ADDRESS REGISTER
13	007772	016002	000004	MOV RMBA(R0),R2	
14	007776	016046	000010	MOV RMCS2(R0),-(SP)	::PUSH RMCS2(R0) ON STACK
15	010002	010160	000010	MOV R1,RMCS2(R0)	:MOVE CONTROL STATUS REGISTER
16	010006	016002	000010	MOV RMCS2(R0),R2	
17	010012	012660	000010	MOV (SP)+,RMCS2(R0)	::POP STACK INTO RMCS2(R0)
18	010016	010160	000022	MOV R1,RMDB(R0)	:MOVE DATA BUFFER
19	010022	016002	000022	MOV RMDB(R0),R2	
20	010026	012637	000006	MOV (SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
	010032	012637	000004	MOV (SP)+,ERRVEC	::POP STACK INTO ERRVEC
21	010036	000417		BR 3\$:NO BUS TIMEOUT OCCURRED
22					
23	010040	022626		1\$: CMP (SP)+,(SP)+	:ADJUST STACK
24	010042	012637	000006	MOV (SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
	010046	012637	000004	MC/ (SP)+,ERRVEC	::POP STACK INTO ERRVEC
25	010052	104110		EMT 110	
26	010054	005737	000042	TST @#42	:STAND ALONE MODE ?
27	010060	001002		BNE 2\$:NO!!
28	010062	000137	005424	JMP START1	:YES-GO GET \$BASE
29	010066	005037	001300	2\$: CLR \$DEVM	:FUDGE NO DRIVES IN MAP
30	010072	000137	031372	JMP \$GET42	:RETURN CONTROL TO MONITOR
31	010076			3\$:	

```

*****
:*TEST 2          FORMAT ZEROS
*****
TST2:
    
```

010076	000004			SCOPE	:SCOPE CALL
010100	000240			NOP	:START OF TEST
010102	012706	001100		MOV #STACK,SP	:INITIALIZE STACK POINTER
010106	013700	001276		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
010112	013701	001466		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
010116	012737	000002	001226	MOV #2,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
34					
35				:SETUP PARAMETERS FOR GENERATING DATA BUFFER	
36	010124	012737	000000	MOV #0,RMOFO	:18 BIT FORMAT

```

37 010132
38 010132 012737 001057 001446 1$: MOV #559,RMDCO ;CYLINDER = 559.
39 010140 012737 001000 001420 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
40 010146 012737 177376 001414 MOV #-258,RMWCO ;2 + 256 WORDS (2'S COMP)
41 010154 012737 101716 001416 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
42 010162 012737 000062 001412 MOV #WH,RMCS10 ;WRITE HEADER AND DATA
43
44 ;VERIFY THAT SECTOR IS NOT BAD
010170 004737 032406 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
010174 000405 BR 2$ ;GO TO 2$ IF NO ERROR
010176 104401 063446 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
010202 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
010204 000137 011010 JMP 23$ ;GO TO 23$ IF ERROR
45 010210
46 010210 012737 065116 001174 2$: MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
47 010216 012737 000001 001176 MOV #1,$TMP1
48 010224 004737 034640 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
49 010230
50
51 ;PREPARE DEVICE FOR DATA TRANSFER
52 010230 004737 031456 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
010234 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 4$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 23$ IF ERROR
010236 000404 BR 4$
010240 000240 NOP
010242 104000 EMT
010244 000137 011010 JMP 23$
53 010250 4$:
54
55 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
56 010250 012737 000004 001412 MOV #SEEK,RMCS10 ;CHANGE COMMAND TO SEEK
57 010256 012702 001555 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
58 010262 112722 000006 MOVB #RMDA,(R2)+
59 010266 112722 000034 MOVB #RMDC,(R2)+
60 010272 112722 000032 MOVB #RMOF,(R2)+
61 010276 112722 000000 MOVB #RMCS1,(R2)+
62 010302 112722 000200 MOVB #200,(R2)+
63
64 010306 004737 036004 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
010312 000404 BR 5$ ;GO TO 5$ IF NO ERROR
010314 000240 NOP ;RETURN HERE IF ERROR
010316 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
010320 000137 011010 JMP 23$ ;GO TO 23$ IF ERROR
65 010324 5$:
66 010324 012702 000002 MOV #2,R2 ;DO 2 CLOCK CYCLES
67 010330 004737 036520 6$: JSR PC,STIMER ;START CLOCK TIMER
68 010334 032777 000200 171160 7$: BIT #BIT7,@CLKADR ;TIMER DONE??
69 010342 001774 BEQ 7$ ;NO!!
70 010344 005302 DEC R2 ;DEC NUMBER OF CYCLES
71 010346 001370 BNE 6$ ;CONTINUE IF NOT DONE
72

```

```

73 010350 012737 000005 001412      MOV      #SEEK!GO,RMCS10 ;EXECUTE SEEK COMMAND
74
75 010356 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    010362 000404      BR      8$              ;GO TO 8$ IF NO ERROR
    010364 000240      NOP
    010366 104000      EMT          ;RETURN HERE IF ERROR
    010370 000137 011010      JMP      23$           ;ERROR # DEFINED BY PUT SUBROUTINE
    ;GO TO 23$ IF ERROR
76 010374      8$:
77
78      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
79 010374 004737 035450      JSR      PC,GETSTS      ;SETUP FOR STATUS
80 010400 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR SEEK TO COMPLETE
81 010404
82
83      ;GO READ SEEK STATUS
84 010404 004737 035534      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
    010410 000404      BR      10$            ;GO TO 10$ IF NO ERROR
    010412 000240      NOP          ;RETURN HERE IF ERROR
    010414 104000      EMT          ;ERROR # DEFINED BY GET SUBROUTINE
    010416 000137 011010      JMP      23$           ;GO TO 23$ IF ERROR
85 010422      10$:
86
87      ;VERIFY THE RESULTS OF THE SEEK COMMAND
88 010422 004737 044174      JSR      PC,SEKSTS      ;GO VERIFY RESULTS OF SEEK OPERATION
    010426 000405      BR      11$            ;GO TO 11$ IF NO ERROR
    010430 000240      NOP          ;RETURN HERE IF ERROR
    010432 104000      EMT          ;ERROR # DEFINED BY SEKSTS SUBROUTINE
    010434 004736      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    010436 000137 011010      JMP      23$           ;GO TO 23$ IF ERROR
89 010442      11$:
90
91      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
92 010442 012737 000063 001412      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
93 010450 012702 001560      MOV      #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
94 010454 112722 000002      MOV      #RMWC,(R2)+
95 010460 112722 000004      MOV      #RMSA,(R2)+
96 010464 112722 000000      MOV      #RMCS1,(R2)+
97 010470 112722 000200      MOV      #200,(R2)+
98
99 010474 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    010500 000404      BR      12$            ;GO TO 12$ IF NO ERROR
    010502 000240      NOP          ;RETURN HERE IF ERROR
    010504 104000      EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
    010506 000137 011010      JMP      23$           ;GO TO 23$ IF ERROR
100 010512      12$:
101
102      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
103 010512 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
104
105 010516 004737 035534      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
    010522 000404      BR      13$            ;GO TO 13$ IF NO ERROR
    010524 000240      NOP          ;RETURN HERE IF ERROR
    010526 104000      EMT          ;ERROR # DEFINED BY GET SUBROUTINE
    010530 000137 011010      JMP      23$           ;GO TO 23$ IF ERROR
106 010534      13$:
107
108      ;VERIFY RESULTS OF WRITE COMMAND

```

```

109 010534 004737 036552      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      010540 000405      BR     14$           ;GO TO 14$ IF NO ERROR
      010542 000240      NOP                    ;RETURN HERE IF ERROR
      010544 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      010546 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      010550 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR
110 010554      14$:
111 010554 004737 051612      JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      010560 000405      BR     15$           ;GO TO 15$ IF NO ERROR
      010562 000240      NOP                    ;RETURN HERE IF ERROR
      010564 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      010566 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      010570 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR
112 010574      15$:
113 010574 004737 037404      JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      010600 000405      BR     16$           ;GO TO 16$ IF NO ERROR
      010602 000240      NOP                    ;RETURN HERE IF ERROR
      010604 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      010606 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      010610 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR
114 010614      16$:
115
116      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
117 010614 012737 000073 001412      MOV    #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
118 010622 012737 102722 001416      MOV    #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
119
120 010630 004737 036004      JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      010634 000404      BR     17$           ;GO TO 17$ IF NO ERROR
      010636 000240      NOP                    ;RETURN HERE IF ERROR
      010640 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      010642 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR
121 010646      17$:
122
123      ;WAIT FOR READ TO COMPLETE AND GET STATUS
124 010646 004737 036356      JSR    PC,TIMOUT     ;WAIT FOR READ TO COMPLETE
125
126 010652 004737 035534      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      010656 000404      BR     18$           ;GO TO 18$ IF NO ERROR
      010660 000240      NOP                    ;RETURN HERE IF ERROR
      010662 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      010664 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR
127 010670      18$:
128
129      ;VERIFY THE RESULTS OF READ OPERATION
130 010670 004737 036552      JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      010674 000405      BR     19$           ;GO TO 19$ IF NO ERROR
      010676 000240      NOP                    ;RETURN HERE IF ERROR
      010700 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      010702 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      010704 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR
131 010710      19$:
132 010710 004737 051612      JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      010714 000405      BR     20$           ;GO TO 20$ IF NO ERROR
      010716 000240      NOP                    ;RETURN HERE IF ERROR
      010720 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      010722 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      010724 000137 011010      JMP    23$           ;GO TO 23$ IF ERROR

```

```

133 010730          20$:
134 010730 004737 037404      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
    010734 000405          BR      21$          ;GO TO 21$ IF NO ERROR
    010736 000240          NOP                    ;RETURN HERE IF ERROR
    010740 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
    010742 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    010744 000137 011010      JMP     23$          ;GO TO 23$ IF ERROR

135 010750          21$:
136
137 ;VERIFY DATA
138 010750 004737 035106      JSR    PC,CMPBUF      ;GO COMPARE WRITE, READ DATA BUFFERS
    010754 101716          .WORD  BUFO1          ;STARTING ADDRESS OF WRITE BUFFER
    010756 102722          .WORD  BUFTWO         ;STARTING ADDRESS OF READ BUFFER
    010760 000402          BR      22$          ;GO TO 22$ IF NO ERROR
    010762 000240          NOP                    ;RETURN HERE IF ERROR
    010764 104000          EMT                    ;ERROR # DEFINED BY CMPBUF SUBROUTINE

139 010766          22$:
140 010766 032737 010000 001444  BIT    #FMT16,RMOFO   ;TEST 16 BIT MODE YET ?
141 010774 001005          BNE    23$          ;YES
142 010776 012737 010000 001444  MOV    #FMT16,RMOFO   ;SET 16 BIT MODE AND
143 011004 000137 010132          JMP     1$          ;TEST AGAIN.
144 011010          23$:
145
146 ;*****
;*TEST 3      ZERO FILL TEST
;*****
TST3:
    011010          SCOPE          ;SCOPE CALL
    011010 000004          NOP                    ;START OF TEST
    011012 000240          MOV     #STACK,SP      ;INITIALIZE STACK POINTER
    011014 012706 001100      MOV     $BASE,R0       ;R0 = UNIBUS ADDRESS
    011020 013700 001276      MOV     TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
    011024 013701 001466      MOV     #3,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
    011030 012737 000003 001226

147 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
148
149 011036 012737 001057 001446  MOV     #559.,RMDCO    ;CYLINDER = 559.
150 011044 012737 001000 001420  MOV     #TA2,RMDAO     ;TRACK = 2, SECTOR = 0
151 011052 012737 010000 001444  MOV     #FMT16,RMOFO   ;16 BIT FORMAT
152 011060 012737 177376 001414  MOV     #-258.,RMWCO   ;2 + 256 WORDS (2'S COMP)
153 011066 012737 101716 001416  MOV     #BUFO1,RMBAO   ;DATA BUFFER ADDRESS
154 011074 012737 000062 001412  MOV     #WH,RMCS10     ;WRITE HEADER AND DATA
155
156 ;VERIFY THAT SECTOR IS NOT BAD
    011102 004737 032406      JSR    PC,BADSCT      ;CALL BAD SECTOR MODULE
    011106 000405          BR      1$          ;GO TO 1$ IF NO ERROR
    011110 104401 063446      TYPE    ,SCTMSG       ;TYPE BAD SECTOR MESSAGE
    011114 104000          EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
    011116 000137 011644      JMP     18$         ;GO TO 18$ IF ERROR

157 011122          1$:
158 011122 012737 065116 001174  MOV     #ZEROS,$TMP0   ;USE ALL ZEROS DATA PATTERN
159 011130 012737 000001 001176  MOV     #1,$TMP1
160 011136 004737 034640      JSR    PC,GENBUF      ;GO GENERATE DATA BUFFER
161 011142          2$:
162
163 ;PREPARE DEVICE FOR DATA TRANSFER
164 011142 004737 031456      JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
    011146 154130          .WORD  154130        ;TASK DESCRIPTOR AS FOLLOWS:

```

```

                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 3$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 18$ IF ERROR
011150 000404 BR 3$
011152 000240 NOP
011154 104000 EMT
011156 000137 011644 JMP 18$
165 011162 3$:
166
167 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
168 011162 012737 000005 001412 MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
169 011170 012702 001555 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
170 011174 112722 000006 MOVB #RMDA,(R2)+
171 011200 112722 000034 MOVB #RMDC,(R2)+
172 011204 112722 000032 MOVB #RMOF,(R2)+
173 011210 112722 000000 MOVB #RMCS1,(R2)+
174 011214 112722 000200 MOVB #200,(R2)+
175
176 011220 004737 036004 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011224 000404 BR 4$ ;GO TO 4$ IF NO ERROR
011226 000240 NOP ;RETURN HERE IF ERROR
011230 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
011232 000137 011644 JMP 18$ ;GO TO 18$ IF ERROR
177 011236
178
179 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
180 011236 004737 035450 JSR PC,GETSTS ;SETUP FOR STATUS
181 011242 004737 036356 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
182 011246
183
184 ;GO READ SEEK STATUS
185 011246 004737 035534 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
011252 000404 BR 6$ ;GO TO 6$ IF NO ERROR
011254 000240 NOP ;RETURN HERE IF ERROR
011256 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
011260 000137 011644 JMP 18$ ;GO TO 18$ IF ERROR
186 011264
187
188 ;VERIFY THE RESULTS OF THE SEEK COMMAND
189 011264 004737 044174 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
011270 000405 BR 7$ ;GO TO 7$ IF NO ERROR
011272 000240 NOP ;RETURN HERE IF ERROR
011274 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
011276 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
011300 000137 011644 JMP 18$ ;GO TO 18$ IF ERROR
190 011304
191
192 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
193 011304 012737 177776 001414 MOV #-2,RMWC0 ;FORMAT PARTIAL SECTOR
194 011312 012737 000063 001412 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
195 011320 012702 001560 MOV #PUTINX+3,R2 ;EXTEND REGISTER INDEX TABLE
196 011324 112722 000002 MOVB #RMWC,(R2)+
197 011330 112722 000004 MOVB #RMBA,(R2)+

```

```

198 011334 112722 000000      MOVB  #RMCS1,(R2)+
199 011340 112722 000200      MOVB  #200,(R2)+
200
201 011344 004737 036004      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011350 000404      BR    8$          ;GO TO 8$ IF NO ERROR
      011352 000240      NOP                   ;RETURN HERE IF ERROR
      011354 104000      EMT                   ;ERROR # DEFINED BY PUT SUBROUTINE
      011356 000137 011644      JMP   18$         ;GO TO 18$ IF ERROR
202 011362      8$:
203
204      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
205 011362 004737 036356      JSR   PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
206
207 011366 004737 035534      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011372 000404      BR    9$          ;GO TO 9$ IF NO ERROR
      011374 000240      NOP                   ;RETURN HERE IF ERROR
      011376 104000      EMT                   ;ERROR # DEFINED BY GET SUBROUTINE
      011400 000137 011644      JMP   18$         ;GO TO 18$ IF ERROR
208 011404      9$:
209
210      ;VERIFY RESULTS OF WRITE COMMAND
211 011404 004737 036552      JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      011410 000405      BR    10$         ;GO TO 10$ IF NO ERROR
      011412 000240      NOP                   ;RETURN HERE IF ERROR
      011414 104000      EMT                   ;ERROR # DEFINED BY PRIERR SUBROUTINE
      011416 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      011420 000137 011644      JMP   18$         ;GO TO 18$ IF ERROR
212 011424      10$:
213 011424 004737 051612      JSR   PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      011430 000405      BR    11$         ;GO TO 11$ IF NO ERROR
      011432 000240      NOP                   ;RETURN HERE IF ERROR
      011434 104000      EMT                   ;ERROR # DEFINED BY DTASTS SUBROUTINE
      011436 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      011440 000137 011644      JMP   18$         ;GO TO 18$ IF ERROR
214 011444      11$:
215 011444 004737 037404      JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      011450 000405      BR    12$         ;GO TO 12$ IF NO ERROR
      011452 000240      NOP                   ;RETURN HERE IF ERROR
      011454 104000      EMT                   ;ERROR # DEFINED BY SECERR SUBROUTINE
      011456 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      011460 000137 011644      JMP   18$         ;GO TO 18$ IF ERROR
216 011464      12$:
217
218      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
219 011464 012737 177376 001414  MOV   #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
220 011472 012737 000073 001412  MOV   #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
221 011500 012737 102722 001416  MOV   #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
222
223 011506 004737 036004      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011512 000404      BR    13$         ;GO TO 13$ IF NO ERROR
      011514 000240      NOP                   ;RETURN HERE IF ERROR
      011516 104000      EMT                   ;ERROR # DEFINED BY PUT SUBROUTINE
      011520 000137 011644      JMP   18$         ;GO TO 18$ IF ERROR
224 011524      13$:
225
226      ;WAIT FOR READ TO COMPLETE AND GET STATUS
227 011524 004737 036356      JSR   PC,TIMOUT   ;WAIT FOR READ TO COMPLETE
  
```

```

228
229 011530 004737 035534      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011534 000404          BR     14$        ;GO TO 14$ IF NO ERROR
      011536 000240          NOP                    ;RETURN HERE IF ERROR
      011540 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      011542 000137 011644    JMP    18$        ;GO TO 18$ IF ERROR
230 011546
231
232
233 011546 004737 036552      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      011552 000405          BR     15$        ;GO TO 15$ IF NO ERROR
      011554 000240          NOP                    ;RETURN HERE IF ERROR
      011556 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      011560 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      011562 000137 011644    JMP    18$        ;GO TO 18$ IF ERROR
234 011566
235 011566 004737 051612      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      011572 000405          BR     16$        ;GO TO 16$ IF NO ERROR
      011574 000240          NOP                    ;RETURN HERE IF ERROR
      011576 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      011600 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      011602 000137 011644    JMP    18$        ;GO TO 18$ IF ERROR
236 011606
237 011606 004737 037404      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      011612 000405          BR     17$        ;GO TO 17$ IF NO ERROR
      011614 000240          NOP                    ;RETURN HERE IF ERROR
      011616 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      011620 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      011622 000137 011644    JMP    18$        ;GO TO 18$ IF ERROR
238 011626
239
240
241 011626 004737 035106      JSR    PC,CMPBUF   ;GO COMPARE WRITE, READ DATA BUFFERS
      011632 101716          .WORD BUFOE       ;STARTING ADDRESS OF WRITE BUFFER
      011634 102722          .WORD BUFTWO      ;STARTING ADDRESS OF READ BUFFER
      011636 000402          BR     18$        ;GO TO 18$ IF NO ERROR
      011640 000240          NOP                    ;RETURN HERE IF ERROR
      011642 104000          EMT                    ;ERROR # DEFINED BY CMPBUF SUBROUTINE
242 011644
243
244
      ;*****
      ;*TEST 4      FORMAT CHECK ZEROS
      ;*****
      TST4:
      SCOPE          ;SCOPE CALL
      NOP           ;START OF TEST
      MOV    #STACK,SP ;INITIALIZE STACK POINTER
      MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
      MOV    TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
      MOV    #4,$STESTN ;SET TEST NUMBER IN APT MAIL BOX

245
246
247 011672 012737 001057 001446 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
      MOV    #559,RMDCO ;CYLINDER = 559.
248 011700 012737 001000 001420 ;
      MOV    #TA2,RMDAO ;TRACK = 2, SECTOR = 0
249 011706 012737 010000 001444 ;
      MOV    #FMT16,RMOFO ;16 BIT FORMAT
250 011714 012737 177376 001414 ;
      MOV    #-258,RMWCO ;2 + 256 WORDS (2'S COMP)
251 011722 012737 101716 001416 ;
      MOV    #BUFOE,RMBAO ;DATA BUFFER ADDRESS
    
```



```

252 011730 012737 000062 001412      MOV      #WH,RMCS10      ;WRITE HEADER AND DATA
253                                     ;VERIFY THAT SECTOR IS NOT BAD
254                                     JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
011736 004737 032406      BR       1$             ;GO TO 1$ IF NO ERROR
011742 000405                                     ;TYPE BAD SECTOR MESSAGE
011744 104401 063446      TYPE     ,SCTMSG       ;ERROR # DEFINED BY BADSCT SUBROUTINE
011750 104000                                     ;GO TO 18$ IF ERROR
011752 000137 012434      EMT     18$
1$:                                     JSR      PC,GENBUF      ;GO GENERATE DATA BUFFER
255 011756 012737 065116 001174      MOV      #ZEROS,$TMP0   ;USE ALL ZEROS DATA PATTERN
256 011756 012737 000001 001176      MOV      #1,$TMP1
257 011764 012737 034640      JSR      PC,GENBUF
258 011772 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
259 011776                                     JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
260                                     .WORD    154130         ;TASK DESCRIPTOR AS FOLLOWS:
261                                     ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
262 011776 004737 031456      ;CLEAR CONTROLLER & SELECT DEVICE
012002 154130      ;VERIFY CONTROLLER CLEAR OPERATION
012004 000404      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
012006 000240      ;VERIFY PACK ACKNOWLEDGE
012010 104000      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
012012 000137 012434      ;VERIFY RECALIBRATION
263 012016      ;GO TO 3$ IF NO ERROR
264                                     ;RETURN HERE IF ERROR
265                                     ;ERROR # DEFINED BY TSTPRP SUBROUTINE
266                                     ;GO TO 18$ IF ERROR
266 012016 012737 000005 001412      ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
267 012024 012702 001555      MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
268 012030 112722 000006      MOV      #PUTINX,R2     ;WRITE REGISTER INDEX TABLE
269 012034 112722 000034      MOV      #RMDA,(R2)+
270 012040 112722 000032      MOV      #RMDC,(R2)+
271 012044 112722 000000      MOV      #RMOF,(R2)+
272 012050 112722 000200      MOV      #RMCS1,(R2)+
273                                     MOV      #200,(R2)+
274 012054 004737 036004      JSR      PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
275 012060 000404      BR       4$             ;GO TO 4$ IF NO ERROR
276 012062 000240      NOP                                     ;RETURN HERE IF ERROR
277 012064 104000      EMT     18$             ;ERROR # DEFINED BY PUT SUBROUTINE
278 012066 000137 012434      JMP      18$            ;GO TO 18$ IF ERROR
279 012072                                     ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
280 012072 004737 035450      JSR      PC,GETSTS      ;SETUP FOR STATUS
281 012076 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR SEEK TO COMPLETE
282                                     ;GO READ SEEK STATUS
283 012102 004737 035534      JSR      PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
012106 000404      BR       6$             ;GO TO 6$ IF NO ERROR
012110 000240      NOP                                     ;RETURN HERE IF ERROR
012112 104000      EMT     18$             ;ERROR # DEFINED BY GET SUBROUTINE
012114 000137 012434      JMP      18$            ;GO TO 18$ IF ERROR
  
```

```

284 012120      6$:
285
286
287 012120 004737 044174      ;VERIFY THE RESULTS OF THE SEEK COMMAND
      012124 000405          JSR PC,SEKSTS      ;GO VERIFY RESULTS OF SEEK OPERATION
      012126 000240          BR 7$             ;GO TO 7$ IF NO ERROR
      012130 104000          NOP              ;RETURN HERE IF ERROR
      012132 004736          EMT                  ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      012134 000137 012434   JSR PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      012134 000137 012434   JMP 18$          ;GO TO 18$ IF ERROR

288 012140      7$:
289
290
291 012140 012737 000063 001412 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
      012146 012702 001560   MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
      012152 112722 000002   MOV #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
      012156 112722 000004   MOVB #RMWC,(R2)+
      012162 112722 000000   MOVB #RMBB,(R2)+
      012166 112722 000200   MOVB #RMCS1,(R2)+
      012166 112722 000200   MOVB #200,(R2)+

297
298 012172 004737 036004      JSR PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      012176 000404          BR 8$           ;GO TO 8$ IF NO ERROR
      012200 000240          NOP              ;RETURN HERE IF ERROR
      012202 104000          EMT                  ;ERROR # DEFINED BY PUT SUBROUTINE
      012204 000137 012434   JMP 18$          ;GO TO 18$ IF ERROR

299 012210      8$:
300
301
302 012210 004737 036356      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
      012210 004737 036356   JSR PC,TIMOUT    ;WAIT FOR COMMAND TO COMPLETE

303
304 012214 004737 035534      JSR PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      012220 000404          BR 9$           ;GO TO 9$ IF NO ERROR
      012222 000240          NOP              ;RETURN HERE IF ERROR
      012224 104000          EMT                  ;ERROR # DEFINED BY GET SUBROUTINE
      012226 000137 012434   JMP 18$          ;GO TO 18$ IF ERROR

305 012232      9$:
306
307
308 012232 004737 036552      ;VERIFY RESULTS OF WRITE COMMAND
      012236 000405          JSR PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      012240 000240          BR 10$          ;GO TO 10$ IF NO ERROR
      012242 104000          NOP              ;RETURN HERE IF ERROR
      012244 004736          EMT                  ;ERROR # DEFINED BY PRIERR SUBROUTINE
      012246 000137 012434   JSR PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      012246 000137 012434   JMP 18$          ;GO TO 18$ IF ERROR

309 012252      10$:
310 012252 004737 051612     JSR PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
      012256 000405          BR 11$          ;GO TO 11$ IF NO ERROR
      012260 000240          NOP              ;RETURN HERE IF ERROR
      012262 104000          EMT                  ;ERROR # DEFINED BY DTASTS SUBROUTINE
      012264 004736          JSR PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      012266 000137 012434   JMP 18$          ;GO TO 18$ IF ERROR

311 012272      11$:
312 012272 004737 037404     JSR PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
      012276 000405          BR 12$          ;GO TO 12$ IF NO ERROR
      012300 000240          NOP              ;RETURN HERE IF ERROR
      012302 104000          EMT                  ;ERROR # DEFINED BY SECERR SUBROUTINE
      012304 004736          JSR PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      012306 000137 012434   JMP 18$          ;GO TO 18$ IF ERROR

```

```
313 012312 12$:  
314  
315 ;WRITE CHECK HEADER AND DATA FOR SECTOR JUST WRITTEN  
316 012312 012737 000053 001412 MOV #WCH!GO, RMCS10 ;WRITE CHECK COMMAND  
317  
318 012320 004737 036004 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE  
012324 000404 BR 13$ ;GO TO 13$ IF NO ERROR  
012326 000240 NOP ;RETURN HERE IF ERROR  
012330 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE  
012332 000137 012434 JMP 18$ ;GO TO 18$ IF ERROR  
319 012336 13$:  
320 ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS  
321 JSR PC, TIMEOUT ;WAIT FOR READ TO COMPLETE  
322 012336 004737 036356  
323  
324 012342 004737 035534 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE  
012346 000404 BR 14$ ;GO TO 14$ IF NO ERROR  
012350 000240 NOP ;RETURN HERE IF ERROR  
012352 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE  
012354 000137 012434 JMP 18$ ;GO TO 18$ IF ERROR  
325 012360 14$:  
326 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION  
327 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS  
328 012360 004737 036552 BR 15$ ;GO TO 15$ IF NO ERROR  
012364 000405 NOP ;RETURN HERE IF ERROR  
012366 000240 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE  
012370 104000 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS  
012372 004736 JMP 18$ ;GO TO 18$ IF ERROR  
012374 000137 012434  
329 012400 15$:  
330 012400 004737 051612 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER  
012404 000405 BR 16$ ;GO TO 16$ IF NO ERROR  
012406 000240 NOP ;RETURN HERE IF ERROR  
012410 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE  
012412 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS  
012414 000137 012434 JMP 18$ ;GO TO 18$ IF ERROR  
331 012420 16$:  
332 012420 004737 037404 JSR PC, SECERR ;GO CHECK FOR SECONDARY ERRORS  
012424 000403 BR 17$ ;GO TO 17$ IF NO ERROR  
012426 000240 NOP ;RETURN HERE IF ERROR  
012430 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE  
012432 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS  
333 012434 17$:  
334 18$:  
335 012434  
336  
337  
:*****  
:*TEST 5 FORMAT CHECK ZEROS W/ WCE ERROR  
:*****  
TST5:  
012434 SCOPE ;SCOPE CALL  
012434 000004 NOP ;START OF TEST  
012436 000240 MOV #STACK, SP ;INITIALIZE STACK POINTER  
012440 012706 001100 MOV $BASE, R0 ;R0 = UNIBUS ADDRESS  
012444 013700 001276 MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED  
012450 013701 001466 MOV #5, $TESTN ;SET TEST NUMBER IN APT MAIL BOX  
012454 012737 000005 001226
```

```

339      ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
340 012462 012737 001057 001446      MOV      #559.,RMDCO      ;CYLINDER = 559.
341 012470 012737 001000 001420      MOV      #TA2,RMDAO      ;TRACK = 2, SECTOR = 0
342 012476 012737 010000 001444      MOV      #FMT16,RMOFO    ;16 BIT FORMAT
343 012504 012737 177376 001414      MOV      #-258.,RMWCO    ;2 + 256 WORDS (2'S COMP)
344 012512 012737 101716 001416      MOV      #BUFONE,RMBAO   ;DATA BUFFER ADDRESS
345 012520 012737 000062 001412      MOV      #WH,RMCS10     ;WRITE HEADER AND DATA
346
347      ;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT   ;CALL BAD SECTOR MODULE
      BR      1$          ;GO TO 1$ IF NO ERROR
      TYPE   .SCTMSG     ;TYPE BAD SECTOR MESSAGE
      EMT
      JMP      20$       ;ERROR # DEFINED BY BADSCT SUBROUTINE
                        ;GO TO 20$ IF ERROR
348 012526 004737 032406
349 012532 000405
350 012534 104401 063446
351 012540 104000
352 012542 000137 013374
353
354      1$:
349 012546 012737 065116 001174      MOV      #ZEROS,$TMP0   ;USE ALL ONES DATA PATTERN
350 012554 012737 000001 001176      MOV      #1,$TMP1
351 012562 004737 034640
352 012566
353
354      2$:
      JSR      PC,TSTPRP  ;PREPARE DEVICE FOR TEST
      .WORD   154130     ;TASK DESCRIPTOR AS FOLLOWS:
                        ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                        ;CLEAR CONTROLLER & SELECT DEVICE
                        ;VERIFY CONTROLLER CLEAR OPERATION
                        ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                        ;VERIFY PACK ACKNOWLEDGE
                        ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                        ;VERIFY RECALIBRATION
                        ;GO TO 3$ IF NO ERROR
                        ;RETURN HERE IF ERROR
                        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                        ;GO TO 20$ IF ERROR
355 012566 004737 031456
356 012572 154130
357
358      3$:
      BR      3$
      NOP
      EMT
      JMP      20$
359 012574 000404
360 012576 000240
361 012600 104000
362 012602 000137 013374
363 012606
364
365      ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
359 012606 012737 000005 001412      MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
360 012614 012702 001555      MOV      #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
361 012620 112722 000006      MOV      #RMDA,(R2)+
362 012624 112722 000034      MOV      #RMDC,(R2)+
363 012630 112722 000032      MOV      #RMOF,(R2)+
364 012634 112722 000000      MOV      #RMCS1,(R2)+
365 012640 112722 000200      MOV      #200,(R2)+
366
367 012644 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
368 012650 000404      BR      4$             ;GO TO 4$ IF NO ERROR
369 012652 000240      NOP
370 012654 104000      EMT
371 012656 000137 013374      JMP      20$           ;ERROR # DEFINED BY PUT SUBROUTINE
                        ;GO TO 20$ IF ERROR
372 012662
373
374      4$:
      JSR      PC,GETSTS   ;SETUP FOR STATUS
      JSR      PC,TIMOUT   ;WAIT FOR SEEK TO COMPLETE
371 012662 004737 035450
372 012666 004737 036356
373 012672
374
  
```

```

375                                     ;GO READ SEEK STATUS
376 012672 004737 035534                JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      012676 000404                        BR    6$          ;GO TO 6$ IF NO ERROR
      012700 000240                        NOP                               ;RETURN HERE IF ERROR
      012702 104000                        EMT                               ;ERROR # DEFINED BY GET SUBROUTINE
      012704 000137 013374                JMP   20$        ;GO TO 20$ IF ERROR

377 012710                               6$:
378
379                                     ;VERIFY THE RESULTS OF THE SEEK COMMAND
380 012710 004737 044174                JSR   PC,SEKSTS   ;GO VERIFY RESULTS OF SEEK OPERATION
      012714 000405                        BR    7$          ;GO TO 7$ IF NO ERROR
      012716 000240                        NOP                               ;RETURN HERE IF ERROR
      012720 104000                        EMT                               ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      012722 004736                        JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      012724 000137 013374                JMP   20$        ;GO TO 20$ IF ERROR

381 012730                               7$:
382
383                                     ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
384 012730 012737 000063 001412        MOV   #WH!GO,RMCS10 ;WRITE HEADER AND DATA
385 012736 012702 001560                MOV   #PUTINX+3,R2 ;EXTEND REGISTER INDEX TABLE
386 012742 112722 000002                MOVB  #RMWC,(R2)+
387 012746 112722 000004                MOVB  #RMBA,(R2)+
388 012752 112722 000000                MOVB  #RMCS1,(R2)+
389 012756 112722 000200                MOVB  #200,(R2)+
390
391 012762 004737 036004                JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      012766 000404                        BR    8$          ;GO TO 8$ IF NO ERROR
      012770 000240                        NOP                               ;RETURN HERE IF ERROR
      012772 104000                        EMT                               ;ERROR # DEFINED BY PUT SUBROUTINE
      012774 000137 013374                JMP   20$        ;GO TO 20$ IF ERROR

392 013000                               8$:
393
394                                     ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
395 013000 004737 036356                JSR   PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
396
397 013004 004737 035534                JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      013010 000404                        BR    9$          ;GO TO 9$ IF NO ERROR
      013012 000240                        NOP                               ;RETURN HERE IF ERROR
      013014 104000                        EMT                               ;ERROR # DEFINED BY GET SUBROUTINE
      013016 000137 013374                JMP   20$        ;GO TO 20$ IF ERROR

398 013022                               9$:
399
400                                     ;VERIFY RESULTS OF WRITE COMMAND
401 013022 004737 036552                JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      013026 000405                        BR    10$         ;GO TO 10$ IF NO ERROR
      013030 000240                        NOP                               ;RETURN HERE IF ERROR
      013032 104000                        EMT                               ;ERROR # DEFINED BY PRIERR SUBROUTINE
      013034 004736                        JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      013036 000137 013374                JMP   20$        ;GO TO 20$ IF ERROR

402 013042                               10$:
403 013042 004737 051612                JSR   PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      013046 000405                        BR    11$         ;GO TO 11$ IF NO ERROR
      013050 000240                        NOP                               ;RETURN HERE IF ERROR
      013052 104000                        EMT                               ;ERROR # DEFINED BY DTASTS SUBROUTINE
      013054 004736                        JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      013056 000137 013374                JMP   20$        ;GO TO 20$ IF ERROR

404 013062                               11$:

```

```

405 013062 004737 037404      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      013066 000405      BR     12$           ;GO TO 12$ IF NO ERROR
      013070 000240      NOP                    ;RETURN HERE IF ERROR
      013072 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      013074 004736      JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      013076 000137 013374  JMP    20$           ;GO TO 20$ IF ERROR
406 013102      12$:
407
408      ;ALTER DATA BUFFER
409 013102 005137 102720      COM    BUFTWO-2      ;COMPLEMENT LAST DATA WORD
410
411      ;SETUP AND WRITE CHECK HEADER AND DATA COMMAND
412 013106 012737 000053 001412 MOV    #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
413
414 013114 004737 036004      JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      013120 000404      BR     13$           ;GO TO 13$ IF NO ERROR
      013122 000240      NOP                    ;RETURN HERE IF ERROR
      013124 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      013126 000137 013374  JMP    20$           ;GO TO 20$ IF ERROR
415 013132      13$:
416
417      ;WAIT FOR WRITE CHECK COMMAND TO COMPLETE AND READ STATUS
418 013132 004737 036356      JSR    PC,TIMOUT     ;WAIT FOR COMMAND TO COMPLETE
419
420 013136 004737 035534      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      013142 000404      BR     14$           ;GO TO 14$ IF NO ERROR
      013144 000240      NOP                    ;RETURN HERE IF ERROR
      013146 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      013150 000137 013374  JMP    20$           ;GO TO 20$ IF ERROR
421 013154      14$:
422
423      ;CHECK FOR PRIMARY ERRORS
424 013154 004737 036552      JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      013160 000405      BR     15$           ;GO TO 15$ IF NO ERROR
      013162 000240      NOP                    ;RETURN HERE IF ERROR
      013164 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      013166 004736      JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      013170 000137 013374  JMP    20$           ;GO TO 20$ IF ERROR
425 013174      15$:
426
427      ;MAKE SURE THE WRITE CHECK ERROR WAS DETECTED
428 013174 032737 040000 001346 BIT    #WCE,RMCS21     ;IS WRITE CHECK ERROR SET??
429 013202 001023      BNE    17$           ;YES!!
430
431 013204 004737 051612      JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      013210 000405      BR     16$           ;GO TO 16$ IF NO ERROR
      013212 000240      NOP                    ;RETURN HERE IF ERROR
      013214 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      013216 004736      JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      013220 000137 013374  JMP    20$           ;GO TO 20$ IF ERROR
432
433 013224 013737 001346 001140 16$: MOV    RMCS21,$GDDAT      ;LOAD EXPECTED STATUS
434 013232 052737 040000 001140      BIS    #WCE,$GDDAT
435 013240 013737 001346 001142      MOV    RMCS21,$BDDAT ;LOAD RECEIVED STATUS
436 013246 104337      EMT    337
437 013250 000451      BR     20$
438 013252      17$:

```

```

439
440
441 013252 012737 102720 001134 :VERIFY THE ADDRESS OF THE WRITE CHECK ERROR
442 013260 013737 001342 001136     MOV #BUFTWO-2,$GDADR :LOAD EXPECTED ADDRESS
443 013266 162737 000002 001136     MOV RMBAI,$BDADR :LOAD RECEIVED ADDRESS
444                                     SUB #2,$BDADR :DECREMENT RECEIVED ADDRESS
445
446 013274 112737 000022 001526 :GET WCE DATA AND VERIFY IT IS OK :SETUP FOR READING RMDB
447 013302 112737 000200 001527     MOV #RMDB,GETINX
448                                     MOV #200,GETINX+1
449 013310 004737 035534 :GO READ REGISTER(S) WITH GET SUBROUTINE
450 013314 000404 :GO TO 18$ IF NO ERROR
451 013316 000240 :RETURN HERE IF ERROR
452 013320 104000 :ERROR # DEFINED BY GET SUBROUTINE
453 013322 000137 013374 :GO TO 20$ IF ERROR
454 013326 013737 001360 001142 18$: MOV RMDBI,$BDDAT :LOAD RECEIVED DATA WORD
455 013334 013737 102720 001140     MOV BUFTWO-2,$GDDAT :LOAD EXPECTED DATA WORD
456 013342 005137 001140     COM $GDDAT
457 013346 023737 001134 001136     CMP $GDADR,$BDADR :IS ADDRESS OK??
458 013354 001402     BEQ 19$ :YES!!
459 013356 104340     EMT 340
460 013360 000405     BR 20$
461 013362 023737 001140 001142 19$: CMP $GDDAT,$BDDAT :IS DATA WORD OK??
462 013370 001401     BEQ 20$ :YES!!
463 013372 104341     EMT 341
464 013374 20$:
465
466
467
468
469
470
471
472
473
474
475
476
477

```

 :*TEST 6 FORMAT ONES

```

TST6:
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

:SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV #559,$RMDCO :CYLINDER = 559.
MOV #TA2,$RMDAO :TRACK = 2, SECTOR = 0
MOV #FMT16,$RMOFO :16 BIT FORMAT
MOV #-258,$RMWCO :2 + 256 WORDS (2'S COMP)
MOV #BUFONE,$RMBAO :DATA BUFFER ADDRESS
MOV #WH,$RMCS10 :WRITE HEADER AND DATA

:VERIFY THAT SECTOR IS NOT BAD
JSR PC,BADSCT :CALL BAD SECTOR MODULE
BR 1$ :GO TO 1$ IF NO ERROR
TYPE ,SCTMSG :TYPE BAD SECTOR MESSAGE
EMT :ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 18$ :GO TO 18$ IF ERROR

1$: MOV #ONES,$TMP0 :USE ALL ONES DATA PATTERN
MOV #1,$TMP1
JSR PC,GENBUF :GO GENERATE DATA BUFFER

2$:

```

```

478
479
480 013526 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
    013532 154130              JSR   PC,TSTPRP      ;PREPARE DEVICE FOR TEST
                                .WORD 154130                ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 3$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 18$ IF ERROR
    013534 000404              BR    3$
    013536 000240              NOP
    013540 104000              EMT
    013542 000137 014214      JMP   18$
481 013546                    3$:
482
483
484 013546 012737 000005 001412 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
    013554 012702 001555      MOV   #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
    013560 112722 000006      MOV   #PUTINX,R2       ;WRITE REGISTER INDEX TABLE
    013564 112722 000034      MOVB #RMDA,(R2)+
    013570 112722 000032      MOVB #RMDC,(R2)+
    013574 112722 000000      MOVB #RMOF,(R2)+
    013600 112722 000200      MOVB #RMCS1,(R2)+
    013604 004737 036004      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    013610 000404              BR    4$              ;GO TO 4$ IF NO ERROR
    013612 000240              NOP
    013614 104000              EMT
    013616 000137 014214      JMP   18$            ;RETURN HERE IF ERROR
    013622                    4$:
    013622 004737 035450      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
    013626 004737 036356      JSR   PC,GETSTS      ;SETUP FOR STATUS
    013632                    JSR   PC,TIMOUT        ;WAIT FOR SEEK TO COMPLETE
493 013622                    5$:
494
495
496 013622 004737 035450      ;GO READ SEEK STATUS
    013636 000404              JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
    013640 000240              BR    6$              ;GO TO 6$ IF NO ERROR
    013642 104000              NOP
    013644 000137 014214      EMT
    013650                    JMP   18$            ;RETURN HERE IF ERROR
    013650                    6$:
    013650 004737 044174      ;VERIFY THE RESULTS OF THE SEEK COMMAND
    013654 000405              JSR   PC,SEKSTS      ;GO VERIFY RESULTS OF SEEK OPERATION
    013656 000240              BR    7$              ;GO TO 7$ IF NO ERROR
    013660 104000              NOP
    013662 004736              EMT
    013664 000137 014214      JSR   PC,@(SP)+     ;RETURN HERE IF ERROR
    013670                    JMP   18$            ;ERROR # DEFINED BY SEKSTS SUBROUTINE
    013670                    7$:
    013670 012737 000063 001412 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
    013670                    MOV   #WH!GO,RMCS10 ;WRITE HEADER AND DATA

```



```

510 013676 012702 001560      MOV      #PUTINX+3,R2      ;EXTEND REGISTER INDEX TABLE
511 013702 112722 000002      MOV      #RMWC,(R2)+
512 013706 112722 000004      MOV      #RMB A,(R2)+
513 013712 112722 000000      MOV      #RMCS1,(R2)+
514 013716 112722 000200      MOV      #200,(R2)+
515
516 013722 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      013726 000404      BR      8$              ;GO TO 8$ IF NO ERROR
      013730 000240      NOP
      013732 104000      EMT
      013734 000137 014214    JMP      18$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 18$ IF ERROR
517 013740      8$:
518
519      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
520 013740 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
521
522 013744 004737 035534      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      013750 000404      BR      9$              ;GO TO 9$ IF NO ERROR
      013752 000240      NOP
      013754 104000      EMT
      013756 000137 014214    JMP      18$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 18$ IF ERROR
523 013762      9$:
524
525      ;VERIFY RESULTS OF WRITE COMMAND
526 013762 004737 036552      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      013766 000405      BR      10$             ;GO TO 10$ IF NO ERROR
      013770 000240      NOP
      013772 104000      EMT
      013774 004736      JSR      PC,@(SP)+      ;RETURN HERE IF ERROR
      013776 000137 014214    JMP      18$            ;ERROR # DEFINED BY PRIERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 18$ IF ERROR
527 014002      10$:
528 014002 004737 051612      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      014006 000405      BR      11$             ;GO TO 11$ IF NO ERROR
      014010 000240      NOP
      014012 104000      EMT
      014014 004736      JSR      PC,@(SP)+      ;RETURN HERE IF ERROR
      014016 000137 014214    JMP      18$            ;ERROR # DEFINED BY DTASTS SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 18$ IF ERROR
529 014022      11$:
530 014022 004737 037404      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      014026 000405      BR      12$             ;GO TO 12$ IF NO ERROR
      014030 000240      NOP
      014032 104000      EMT
      014034 004736      JSR      PC,@(SP)+      ;RETURN HERE IF ERROR
      014036 000137 014214    JMP      18$            ;ERROR # DEFINED BY SECERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 18$ IF ERROR
531 014042      12$:
532
533      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
534 014042 012737 000073 001412    MOV      #RH!GO,RMCS10   ;READ HEADER & DATA COMMAND
535 014050 012737 102722 001416    MOV      #BUFTWO,RMBAO   ;CHANGE BUS ADDRESS
536
537 014056 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      014062 000404      BR      13$             ;GO TO 13$ IF NO ERROR
      014064 000240      NOP
      014066 104000      EMT
      014070 000137 014214    JMP      18$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 18$ IF ERROR
538 014074      13$:
539

```

```

540 ;WAIT FOR READ TO COMPLETE AND GET STATUS
541 014074 004737 036356 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
542
543 014100 004737 035534 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014104 000404 BR 14$ ;GO TO 14$ IF NO ERROR
    014106 000240 NOP ;RETURN HERE IF ERROR
    014110 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    014112 000137 014214 JMP 18$ ;GO TO 18$ IF ERROR
544 014116 14$:
545
546 ;VERIFY THE RESULTS OF READ OPERATION
547 014116 004737 036552 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    014122 000405 BR 15$ ;GO TO 15$ IF NO ERROR
    014124 000240 NOP ;RETURN HERE IF ERROR
    014126 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    014130 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    014132 000137 014214 JMP 18$ ;GO TO 18$ IF ERROR
548 014136 15$:
549 014136 004737 051612 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    014142 000405 BR 16$ ;GO TO 16$ IF NO ERROR
    014144 000240 NOP ;RETURN HERE IF ERROR
    014146 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
    014150 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    014152 000137 014214 JMP 18$ ;GO TO 18$ IF ERROR
550 014156 16$:
551 014156 004737 037404 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    014162 000405 BR 17$ ;GO TO 17$ IF NO ERROR
    014164 000240 NOP ;RETURN HERE IF ERROR
    014166 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    014170 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    014172 000137 014214 JMP 18$ ;GO TO 18$ IF ERROR
552 014176 17$:
553
554 ;VERIFY DATA
555 014176 004737 035106 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
    014202 101716 .WORD BUFOne ;STARTING ADDRESS OF WRITE BUFFER
    014204 102722 .WORD BUFTwo ;STARTING ADDRESS OF READ BUFFER
    014206 000402 BR 18$ ;GO TO 18$ IF NO ERROR
    014210 000240 NOP ;RETURN HERE IF ERROR
    014212 104000 EMT ;ERROR # DEFINED BY CMPBUF SUBROUTINE
556 014214 18$:
557
558 ;*****
;*TEST 7 FORMAT CHECK ONES
;*****
TST7:
    014214 SCOPE ;SCOPE CALL
    014214 000004 NOP ;START OF TEST
    014216 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
    014220 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
    014224 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
    014230 013701 001466 MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
    014234 012737 000007 001226
559
560 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
561 014242 012737 001057 001446 MOV #559,,RMDCO ;CYLINDER = 559.
562 014250 012737 001000 001420 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
563 014256 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
    
```

```

564 014264 012737 177376 001414      MOV      #-258.,RMWCO      ;2 + 256 WORDS (2'S COMP)
565 014272 012737 101716 001416      MOV      #BUFONE,RMBAO    ;DATA BUFFER ADDRESS
566 014300 012737 000062 001412      MOV      #WH,RMCS10      ;WRITE HEADER AND DATA
567
568                                     ;VERIFY THAT SECTOR IS NOT BAD
      014306 004737 032406      JSR      PC,BADSCT        ;CALL BAD SECTOR MODULE
      014312 000405                BR      1$                ;GO TO 1$ IF NO ERROR
      014314 104401 063446      TYPE     ,SCTMSG         ;TYPE BAD SECTOR MESSAGE
      014320 104000                EMT                     ;ERROR # DEFINED BY BADSCT SUBROUTINE
      014322 000137 015004      JMP      18$             ;GO TO 18$ IF ERROR
569 014326                                1$:
570 014326 012737 065054 001174      MOV      #ONES,$TMP0     ;USE ALL ONES DATA PATTERN
571 014334 012737 000001 001176      MOV      #1,$TMP1
572 014342 004737 034640      JSR      PC,GENBUF        ;GO GENERATE DATA BUFFER
573 014346                                2$:
574
575                                     ;PREPARE DEVICE FOR DATA TRANSFER
576 014346 004737 031456      JSR      PC,TSTPRP       ;PREPARE DEVICE FOR TEST
      014352 154130      .WORD    154130          ;TASK DESCRIPTOR AS FOLLOWS:
                                     ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                     ;CLEAR CONTROLLER & SELECT DEVICE
                                     ;VERIFY CONTROLLER CLEAR OPERATION
                                     ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                     ;VERIFY PACK ACKNOWLEDGE
                                     ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                     ;VERIFY RECALIBRATION
      014354 000404                BR      3$                ;GO TO 3$ IF NO ERROR
      014356 000240                NOP                     ;RETURN HERE IF ERROR
      014360 104000                EMT                     ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      014362 000137 015004      JMP      18$             ;GO TO 18$ IF ERROR
577 014366                                3$:
578
579                                     ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
580 014366 012737 000005 001412      MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
581 014374 012702 001555                MOV      #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
582 014400 112722 000006      MOV      #RMDA,(R2)+
583 014404 112722 000034      MOV      #RMDC,(R2)+
584 014410 112722 000032      MOV      #RMOF,(R2)+
585 014414 112722 000000      MOV      #RMCS1,(R2)+
586 014420 112722 000200      MOV      #200,(R2)+
587
588 014424 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      014430 000404                BR      4$                ;GO TO 4$ IF NO ERROR
      014432 000240                NOP                     ;RETURN HERE IF ERROR
      014434 104000                EMT                     ;ERROR # DEFINED BY PUT SUBROUTINE
      014436 000137 015004      JMP      18$             ;GO TO 18$ IF ERROR
589 014442                                4$:
590
591                                     ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
592 014442 004737 035450      JSR      PC,GETSTS       ;SETUP FOR STATUS
593 014446 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR SEEK TO COMPLETE
594 014452                                5$:
595
596                                     ;GO READ SEEK STATUS
597 014452 004737 035534      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      014456 000404                BR      6$                ;GO TO 6$ IF NO ERROR
      014460 000240                NOP                     ;RETURN HERE IF ERROR
  
```

```

014462 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
014464 000137 015004   JMP          18$          ;GO TO 18$ IF ERROR
598 014470          6$:
599
600 ;VERIFY THE RESULTS OF THE SEEK COMMAND
601 014470 004737 044174 JSR          PC,SEKSTS    ;GO VERIFY RESULTS OF SEEK OPERATION
    014474 000405          BR          7$          ;GO TO 7$ IF NO ERROR
    014476 000240          NOP          ;RETURN HERE IF ERROR
    014500 104000          EMT          ;ERROR # DEFINED BY SEKSTS SUBROUTINE
    014502 004736          JSR          PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
    014504 000137 015004   JMP          18$          ;GO TO 18$ IF ERROR
602 014510          7$:
603
604 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
605 014510 012737 000063 001412 MOV          #WH!GO,RMCS10 ;WRITE HEADER AND DATA
606 014516 012702 001560 MOV          #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
607 014522 112722 000002 MOVB        #RMWC,(R2)+
608 014526 112722 000004 MOVB        #RMBA,(R2)+
609 014532 112722 000000 MOVB        #RMCS1,(R2)+
610 014536 112722 000200 MOVB        #200,(R2)+
611
612 014542 004737 036004   JSR          PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    014546 000404          BR          8$          ;GO TO 8$ IF NO ERROR
    014550 000240          NOP          ;RETURN HERE IF ERROR
    014552 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
    014554 000137 015004   JMP          18$          ;GO TO 18$ IF ERROR
613 014560          8$:
614
615 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
616 014560 004737 036356   JSR          PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
617
618 014564 004737 035534   JSR          PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014570 000404          BR          9$          ;GO TO 9$ IF NO ERROR
    014572 000240          NOP          ;RETURN HERE IF ERROR
    014574 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
    014576 000137 015004   JMP          18$          ;GO TO 18$ IF ERROR
619 014602          9$:
620
621 ;VERIFY RESULTS OF WRITE COMMAND
622 014602 004737 036552   JSR          PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
    014606 000405          BR          10$         ;GO TO 10$ IF NO ERROR
    014610 000240          NOP          ;RETURN HERE IF ERROR
    014612 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
    014614 004736          JSR          PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
    014616 000137 015004   JMP          18$          ;GO TO 18$ IF ERROR
623 014622          10$:
624 014622 004737 051612   JSR          PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
    014626 000405          BR          11$         ;GO TO 11$ IF NO ERROR
    014630 000240          NOP          ;RETURN HERE IF ERROR
    014632 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
    014634 004736          JSR          PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
    014636 000137 015004   JMP          18$          ;GO TO 18$ IF ERROR
625 014642          11$:
626 014642 004737 037404   JSR          PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
    014646 000405          BR          12$         ;GO TO 12$ IF NO ERROR
    014650 000240          NOP          ;RETURN HERE IF ERROR
    014652 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
  
```

```

014654 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
014656 000137 015004  JMP    18$      ;GO TO 18$ IF ERROR
627 014662      12$:
628
629      ;WRITE CHECK HEADER AND DATA FOR SECTOR JUST WRITTEN
630 014662 012737 000053 001412  MOV    #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
631
632 014670 004737 036004      JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
014674 000404      BR    13$      ;GO TO 13$ IF NO ERROR
014676 000240      NOP           ;RETURN HERE IF ERROR
014700 104000      EMT           ;ERROR # DEFINED BY PUT SUBROUTINE
014702 000137 015004      JMP    18$      ;GO TO 18$ IF ERROR
633 014706      13$:
634
635      ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
636 014706 004737 036356      JSR    PC,TIMOUT   ;WAIT FOR READ TO COMPLETE
637
638 014712 004737 035534      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
014716 000404      BR    14$      ;GO TO 14$ IF NO ERROR
014720 000240      NOP           ;RETURN HERE IF ERROR
014722 104000      EMT           ;ERROR # DEFINED BY GET SUBROUTINE
014724 000137 015004      JMP    18$      ;GO TO 18$ IF ERROR
639 014730      14$:
640
641      ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
642 014730 004737 036552      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
014734 000405      BR    15$      ;GO TO 15$ IF NO ERROR
014736 000240      NOP           ;RETURN HERE IF ERROR
014740 104000      EMT           ;ERROR # DEFINED BY PRIERR SUBROUTINE
014742 004736      JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
014744 000137 015004      JMP    18$      ;GO TO 18$ IF ERROR
643 014750      15$:
644 014750 004737 051612      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
014754 000405      BR    16$      ;GO TO 16$ IF NO ERROR
014756 000240      NOP           ;RETURN HERE IF ERROR
014760 104000      EMT           ;ERROR # DEFINED BY DTASTS SUBROUTINE
014762 004736      JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
014764 000137 015004      JMP    18$      ;GO TO 18$ IF ERROR
645 014770      16$:
646 014770 004737 037404      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
014774 000403      BR    17$      ;GO TO 17$ IF NO ERROR
014776 000240      NOP           ;RETURN HERE IF ERROR
015000 104000      EMT           ;ERROR # DEFINED BY SECERR SUBROUTINE
015002 004736      JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
647 015004      17$:
648
649 015004      18$:
650
651

```

```

*****
;*TEST 10      FORMAT CHECK ONES W/ WCE ERRORS
*****

```

```

TST10:
015004      SCOPE      ;SCOPE CALL
015006 000240      NOP      ;START OF TEST
015010 012706 001100  MOV    #STACK,SP  ;INITIALIZE STACK POINTER
015014 013700 001276  MOV    $BASE,R0   ;R0 = UNIBUS ADDRESS
015020 013701 001466  MOV    TSTQUE,R1  ;(R1) = DEVICE BEING TESTED

```

```

015024 012737 000010 001226      MOV      #10,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
652
653      ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
654 015032 012737 001057 001446      MOV      #559.,RMDCO      ;CYLINDER = 559.
655 015040 012737 001000 001420      MOV      #TA2,RMDAO      ;TRACK = 2, SECTOR = 0
656 015046 012737 010000 001444      MOV      #FMT16,RMOFO      ;16 BIT FORMAT
657 015054 012737 177376 001414      MOV      #-258.,RMWCO      ;2 + 256 WORDS (2'S COMP)
658 015062 012737 101716 001416      MOV      #BUFONE,RMBAO      ;DATA BUFFER ADDRESS
659 015070 012737 000062 001412      MOV      #WH,RMCS10      ;WRITE HEADER AND DATA
660
661      ;VERIFY THAT SECTOR IS NOT BAD
015076 004737 032406      JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
015102 000405      BR      1$      ;GO TO 1$ IF NO ERROR
015104 104401 063446      TYPE      ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
015110 104000      EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
015112 000137 015742      JMP      20$      ;GO TO 20$ IF ERROR
662 015116      1$:
663 015116 012737 065054 001174      MOV      #ONES,$TMP0      ;USE ALL ONES DATA PATTERN
664 015124 012737 000001 001176      MOV      #1,$TMP1
665 015132 004737 034640      JSR      PC,GENBUF      ;GO GENERATE DATA BUFFER
666 015136      2$:
667
668      ;PREPARE DEVICE FOR DATA TRANSFER
669 015136 004737 031456      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
015142 154130      .WORD    154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
015144 000404      BR      3$      ;GO TO 3$ IF NO ERROR
015146 000240      NOP      ;RETURN HERE IF ERROR
015150 104000      EMT      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
015152 000137 015742      JMP      20$      ;GO TO 20$ IF ERROR
670 015156      3$:
671
672      ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
673 015156 012737 000005 001412      MOV      #SEEK!GO,RMCS10      ;CHANGE COMMAND TO SEEK
674 015164 012702 001555      MOV      #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
675 015170 112722 000006      MOV      #RMDA,(R2)+
676 015174 112722 000034      MOV      #RMDC,(R2)+
677 015200 112722 000032      MOV      #RMOF,(R2)+
678 015204 112722 000000      MOV      #RMCS1,(R2)+
679 015210 112722 000200      MOV      #200,(R2)+
680
681 015214 004737 036004      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
015220 000404      BR      4$      ;GO TO 4$ IF NO ERROR
015222 000240      NOP      ;RETURN HERE IF ERROR
015224 104000      EMT      ;ERROR # DEFINED BY PUT SUBROUTINE
015226 000137 015742      JMP      20$      ;GO TO 20$ IF ERROR
682 015232      4$:
683
684      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
685 015232 004737 035450      JSR      PC,GETSTS      ;SETUP FOR STATUS
686 015236 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR SEEK TO COMPLETE

```

```

687 015242          5$:
688
689
690 015242 004737 035534      ;GO READ SEEK STATUS
      015246 000404          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      015250 000240          BR     6$              ;GO TO 6$ IF NO ERROR
      015252 104000          NOP                    ;RETURN HERE IF ERROR
      015254 000137 015742    EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      015260          JMP     20$          ;GO TO 20$ IF ERROR

691 015260          6$:
692
693
694 015260 004737 044174      ;VERIFY THE RESULTS OF THE SEEK COMMAND
      015264 000405          JSR    PC,SEKSTS       ;GO VERIFY RESULTS OF SEEK OPERATION
      015266 000240          BR     7$              ;GO TO 7$ IF NO ERROR
      015270 104000          NOP                    ;RETURN HERE IF ERROR
      015272 004736          EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      015274 000137 015742    JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      015300          JMP     20$          ;GO TO 20$ IF ERROR

695 015300          7$:
696
697
698 015300 012737 000063 001412 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
      015306 012702 001560    MOV    #WH!GO,RMCS10  ;WRITE HEADER AND DATA
      015312 112722 000002    MOV    #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
      015316 112722 000004    MOVB  #RMWC,(R2)+
      015322 112722 000000    MOVB  #RMBA,(R2)+
      015326 112722 000200    MOVB  #RMCS1,(R2)+
      015332 004737 036004    JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      015336 000404          BR     8$              ;GO TO 8$ IF NO ERROR
      015340 000240          NOP                    ;RETURN HERE IF ERROR
      015342 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      015344 000137 015742    JMP     20$          ;GO TO 20$ IF ERROR

706 015350          8$:
707
708
709 015350 004737 036356      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
      015354 004737 035534    JSR    PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
      015360 000404          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      015362 000240          BR     9$              ;GO TO 9$ IF NO ERROR
      015364 104000          NOP                    ;RETURN HERE IF ERROR
      015366 000137 015742    EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      015372          JMP     20$          ;GO TO 20$ IF ERROR

712 015372          9$:
713
714
715 015372 004737 036552      ;VERIFY RESULTS OF WRITE COMMAND
      015376 000405          JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      015400 000240          BR     10$           ;GO TO 10$ IF NO ERROR
      015402 104000          NOP                    ;RETURN HERE IF ERROR
      015404 004736          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      015406 000137 015742    JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      015412          JMP     20$          ;GO TO 20$ IF ERROR

716 015412          10$:
717 015412 004737 051612      ;GO VERIFY RESULTS OF DATA TRANSFER
      015416 000405          JSR    PC,DTASTS      ;GO TO 11$ IF NO ERROR
      015420 000240          BR     11$           ;RETURN HERE IF ERROR
      015422 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      015424 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
  
```

```

718 015426 000137 015742          11$:  JMP      20$          ;GO TO 20$ IF ERROR
719 015432 004737 037404          JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      015436 000405          BR        12$          ;GO TO 12$ IF NO ERROR
      015440 000240          NOP              ;RETURN HERE IF ERROR
      015442 104000          EMT              ;ERROR # DEFINED BY SECERR SUBROUTINE
      015444 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      015446 000137 015742          JMP      20$          ;GO TO 20$ IF ERROR
720 015452          12$:
721
722          ;ALTER DATA BUFFER
723 015452 005137 102720          COM      BUFTWO-2      ;COMPLEMENT DATA WORD
724
725          ;SETUP AND WRITE CHECK HEADER AND DATA COMMAND
726 015456 012737 000053 001412    MOV      #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
727
728 015464 004737 036004          JSR      PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      015470 000404          BR        13$          ;GO TO 13$ IF NO ERROR
      015472 000240          NOP              ;RETURN HERE IF ERROR
      015474 104000          EMT              ;ERROR # DEFINED BY PUT SUBROUTINE
      015476 000137 015742          JMP      20$          ;GO TO 20$ IF ERROR
729 015502          13$:
730
731          ;WAIT FOR WRITE CHECK COMMAND TO COMPLETE AND READ STATUS
732 015502 004737 036356          JSR      PC,TIMOUT     ;WAIT FOR COMMAND TO COMPLETE
733
734 015506 004737 035534          JSR      PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      015512 000404          BR        14$          ;GO TO 14$ IF NO ERROR
      015514 000240          NOP              ;RETURN HERE IF ERROR
      015516 104000          EMT              ;ERROR # DEFINED BY GET SUBROUTINE
      015520 000137 015742          JMP      20$          ;GO TO 20$ IF ERROR
735 015524          14$:
736
737          ;CHECK FOR PRIMARY ERRORS
738 015524 004737 036552          JSR      PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      015530 000405          BR        15$          ;GO TO 15$ IF NO ERROR
      015532 000240          NOP              ;RETURN HERE IF ERROR
      015534 104000          EMT              ;ERROR # DEFINED BY PRIERR SUBROUTINE
      015536 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      015540 000137 015742          JMP      20$          ;GO TO 20$ IF ERROR
739 015544          15$:
740
741          ;MAKE SURE THE WRITE CHECK ERROR WAS DETECTED
742 015544 032737 040000 001346    BIT      #WCE,RMCS21    ;IS WRITE CHECK ERROR SET??
743 015552 001022          BNE      17$          ;YES!!
744
745 015554 004737 051612          JSR      PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      015560 000405          BR        16$          ;GO TO 16$ IF NO ERROR
      015562 000240          NOP              ;RETURN HERE IF ERROR
      015564 104000          EMT              ;ERROR # DEFINED BY DTASTS SUBROUTINE
      015566 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      015570 000137 015742          JMP      20$          ;GO TO 20$ IF ERROR
746 015574 013737 001346 001140    16$:  MOV      RMCS2I,$GDDAT  ;LOAD EXPECTED STATUS
747 015602 052737 040000 001140    BIS      #WCE,$GDDAT
748 015610 013737 001346 001142    MOV      RMCS2I,$BDDAT ;LOAD RECEIVED STATUS
749 015616 104337
750 015620          17$:
  
```



```

751
752 ;VERIFY THE ADDRESS OF THE WRITE CHECK ERROR
753 015620 012737 102720 001134 MOV #BUFTWO-2,$GDADR ;LOAD EXPECTED ADDRESS
754 015626 013737 001342 001136 MOV RMBAI,$BDADR ;LOAD RECEIVED ADDRESS
755 015634 162737 000002 001136 SUB #2,$BDADR ;DECREMENT RECEIVED ADDRESS
756
757 ;GET WCE DATA AND VERIFY IT IS OK ;SETUP FOR READING RMDB
758 015642 112737 000022 001526 MOV #RMDB,GETINX
759 015650 112737 000200 001527 MOV #200,GETINX+1
760
761 015656 004737 035534 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
015662 000404 BR 18$ ;GO TO 18$ IF NO ERROR
015664 000240 NOP ;RETURN HERE IF ERROR
015666 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
015670 000137 015742 JMP 20$ ;GO TO 20$ IF ERROR
762 015674 013737 001360 001142 18$: MOV RMDBI,$BDDAT ;LOAD RECEIVED DATA WORD
763 015702 013737 102720 001140 MOV BUFTWO-2,$GDDAT ;LOAD EXPECTED DATA WORD
764 015710 005137 001140 COM $GDDAT
765 015714 023737 001134 001136 CMP $GDADR,$BDADR ;IS ADDRESS OK??
766 015722 001402 BEQ 19$ ;YES!!
767 015724 104340 EMT 340
768 015726 000405 BR 20$
769 015730
770 015730 023737 001140 001142 19$: CMP $GDDAT,$BDDAT ;IS DATA WORD OK??
771 015736 001401 BEQ 20$ ;YES!!
772 015740 104341 EMT 341
773 015742 20$:
774
775

```

```

*****
:*TEST 11 FORMAT MULTIPLE SECTORS
*****
TST11:

```

```

015742
015742 000004 SCOPE ;SCOPE CALL
015744 000240 NOP ;START OF TEST
015746 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
015752 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
015756 013701 001466 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
015762 012737 000011 001226 MOV #11,$STSTN ;:SET TEST NUMBER IN APT MAIL BOX
776
777 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
778 015770 012737 010000 001444 MOV #FMT16,RMOFO ;:16 BIT FORMAT
779 015776 1$:
780 015776 012737 001057 001446 MOV #559,RMDCO ;:CYLINDER = 559.
781 016004 012737 001000 001420 MOV #TA2,RMDAO ;:TRACK = 2, SECTOR = 0
782 016012 012737 176774 001414 MOV #-258.*2,RMWCO ;:WORD COUNT FOR 2 SECTORS (2'S COMP)
783 016020 012737 101716 001416 MOV #BUFONE,RMBAO ;:DATA BUFFER ADDRESS
784 016026 012737 000062 001412 MOV #WH,RMC$10 ;:WRITE HEADER AND DATA
785
786 ;VERIFY THAT SECTOR IS NOT BAD
016034 004737 032406 JSR PC,BADSCT ;:CALL BAD SECTOR MODULE
016040 000405 BR 2$ ;:GO TO 2$ IF NO ERROR
016042 104401 063446 TYPE ,SCTMSG ;:TYPE BAD SECTOR MESSAGE
016046 104000 EMT ;:ERROR # DEFINED BY BADSCT SUBROUTINE
016050 000137 016560 JMP 19$ ;:GO TO 19$ IF ERROR
787 016054 2$:
788 016054 012737 065116 001174 MOV #ZEROS,$TMP0 ;:USE ALL ZEROS DATA PATTERN
789 016062 012737 000001 001176 MOV #1,$TMP1

```

```

790 016070 004737 034640          JSR    PC,GENBUF      ;GO GENERATE DATA BUFFER
791 016074          3$:
792
793
794 016074 004737 031456          JSR    PC,TSTPRP     ;PREPARE DEVICE FOR DATA TRANSFER
    016100 154130          .WORD   154130     ;PREPARE DEVICE FOR TEST
                                                ;TASK DESCRIPTOR AS FOLLOWS:
                                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                                ;CLEAR CONTROLLER & SELECT DEVICE
                                                ;VERIFY CONTROLLER CLEAR OPERATION
                                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                                ;VERIFY PACK ACKNOWLEDGE
                                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                                ;VERIFY RECALIBRATION
                                                ;GO TO 4$ IF NO ERROR
                                                ;RETURN HERE IF ERROR
                                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                                ;GO TO 19$ IF ERROR

    016102 000404          BR     4$
    016104 000240          NOP
    016106 104000          EMT
    016110 000137 016560          JMP    19$
795 016114          4$:
796
797
798 016114 012737 000005 001412  ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
799 016122 012702 001555          MOV    #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
800 016126 112722 000006          MOV    #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
801 016132 112722 000034          MOVB  #RMDA,(R2)+
802 016136 112722 000032          MOVB  #RMDC,(R2)+
803 016142 112722 000000          MOVB  #RMOF,(R2)+
804 016146 112722 000200          MOVB  #RMCS1,(R2)+
805
806 016152 004737 036004          JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    016156 000404          BR     5$            ;GO TO 5$ IF NO ERROR
    016160 000240          NOP                ;RETURN HERE IF ERROR
    016162 104000          EMT                ;ERROR # DEFINED BY PUT SUBROUTINE
    016164 000137 016560          JMP    19$          ;GO TO 19$ IF ERROR
807 016170          5$:
808
809
810 016170 004737 035450          ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
811 016174 004737 036356          JSR    PC,GETSTS     ;SETUP FOR STATUS
812 016200          JSR    PC,TIMOUT    ;WAIT FOR SEEK TO COMPLETE
813
814
815 016200 004737 035534          ;GO READ SEEK STATUS
    016204 000404          JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
    016206 000240          BR     7$            ;GO TO 7$ IF NO ERROR
    016210 104000          NOP                ;RETURN HERE IF ERROR
    016212 000137 016560          EMT                ;ERROR # DEFINED BY GET SUBROUTINE
    016216          JMP    19$          ;GO TO 19$ IF ERROR
816 016216          7$:
817
818
819 016216 004737 044174          ;VERIFY THE RESULTS OF THE SEEK COMMAND
    016222 000405          JSR    PC,SEKSTS    ;GO VERIFY RESULTS OF SEEK OPERATION
    016224 000240          BR     8$            ;GO TO 8$ IF NO ERROR
    016226 104000          NOP                ;RETURN HERE IF ERROR
    016230 004736          EMT                ;ERROR # DEFINED BY SEKSTS SUBROUTINE
    016232 000137 016560          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
    016236          JMP    19$          ;GO TO 19$ IF ERROR
820
821          8$:
  
```

```

822      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
823 016236 012737 000063 001412  MOV    #WH!GO,RMCS10 ;WRITE HEADER AND DATA
824 016244 012702 001560          MOV    #PUTINX+3,R2 ;EXTEND REGISTER INDEX TABLE
825 016250 112722 000002          MOVB  #RMWC,(R2)+
826 016254 112722 000004          MOVB  #RMBA,(R2)+
827 016260 112722 000000          MOVB  #RMCS1,(R2)+
828 016264 112722 000200          MOVB  #200,(R2)+
829
830 016270 004737 036004          JSR   PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR    9$ ;GO TO 9$ IF NO ERROR
      NOP   ;RETURN HERE IF ERROR
      EMT   ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP  19$ ;GO TO 19$ IF ERROR
9$:
831 016306
832
833      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
834 016306 004737 036356          JSR   PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
835
836 016312 004737 035534          JSR   PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR    10$ ;GO TO 10$ IF NO ERROR
      NOP   ;RETURN HERE IF ERROR
      EMT   ;ERROR # DEFINED BY GET SUBROUTINE
      JMP  19$ ;GO TO 19$ IF ERROR
10$:
837 016330
838
839      ;VERIFY RESULTS OF WRITE COMMAND
840 016330 004737 036552          JSR   PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      BR    11$ ;GO TO 11$ IF NO ERROR
      NOP   ;RETURN HERE IF ERROR
      EMT   ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR   PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP  19$ ;GO TO 19$ IF ERROR
11$:
841 016350
842 016350 004737 051612          JSR   PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
      BR    12$ ;GO TO 12$ IF NO ERROR
      NOP   ;RETURN HERE IF ERROR
      EMT   ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR   PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP  19$ ;GO TO 19$ IF ERROR
12$:
843 016370
844 016370 004737 037404          JSR   PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
      BR    13$ ;GO TO 13$ IF NO ERROR
      NOP   ;RETURN HERE IF ERROR
      EMT   ;ERROR # DEFINED BY SECERR SUBROUTINE
      JSR   PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP  19$ ;GO TO 19$ IF ERROR
13$:
845 016410
846
847      ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
848 016410 012737 000053 001412  MOV    #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
849
850 016416 004737 036004          JSR   PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR    14$ ;GO TO 14$ IF NO ERROR
      NOP   ;RETURN HERE IF ERROR
      EMT   ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP  19$ ;GO TO 19$ IF ERROR
14$:
851 016434
  
```

```

852
853
854 016434 004737 036356      ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
855
856 016440 004737 035534      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      016444 000404          BR      15$             ;GO TO 15$ IF NO ERROR
      016446 000240          NOP
      016450 104000          EMT
      016452 000137 016560    JMP      19$             ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 19$ IF ERROR
857 016456
858
859
860 016456 004737 036552      ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
      016462 000405          JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      016464 000240          BR      16$             ;GO TO 16$ IF NO ERROR
      016466 104000          NOP
      016470 004736          EMT
      016472 000137 016560    JSR      PC,@(SP)+      ;RETURN HERE IF ERROR
      JMP      19$             ;ERROR # DEFINED BY PRIERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 19$ IF ERROR
861 016476
862 016476 004737 051612      16$:
      016502 000405          JSR      PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      016504 000240          BR      17$             ;GO TO 17$ IF NO ERROR
      016506 104000          NOP
      016510 004736          EMT
      016512 000137 016560    JSR      PC,@(SP)+      ;RETURN HERE IF ERROR
      JMP      19$             ;ERROR # DEFINED BY DTASTS SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 19$ IF ERROR
863 016516
864 016516 004737 037404      17$:
      016522 000405          JSR      PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      016524 000240          BR      18$             ;GO TO 18$ IF NO ERROR
      016526 104000          NOP
      016530 004736          EMT
      016532 000137 016560    JSR      PC,@(SP)+      ;RETURN HERE IF ERROR
      JMP      19$             ;ERROR # DEFINED BY SECERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 19$ IF ERROR
865 016536
866 016536 032737 001000 001444 18$:
867 016544 001005          BIT      #SSEI,RMOFO   ;TEST 16 BIT MODE W/ SSEI YET ?
868 016546 052737 001000 001444 BNE      19$             ;YES !!
869 016554 000137 015776    BIS      #SSEI,RMOFO   ;SET SSEI IN OFFSET AND
870 016560          JMP      1$             ;TEST AGAIN.
871
872
      ;*****
      ;*TEST 12      READ HEADER & DATA W/ HEAD SWITCHING
      ;*****
      TST12:
      016560          SCOPE
      016560 000004          NOP
      016562 000240          MOV      #STACK,SP    ;SCOPE CALL
      016564 012706 001100    MOV      $BASE,R0     ;START OF TEST
      016570 013700 001276    MOV      TSTQUE,R1    ;INITIALIZE STACK POINTER
      016574 013701 001466    MOV      #12,$TESTN   ;R0 = UNIBUS ADDRESS
      016600 012737 000012 001226 MOV      #12,$TESTN   ;(R1) = DEVICE BEING TESTED
      ;:SET TEST NUMBER IN APT MAIL BOX

873
874
875 016606 012737 001036 001420 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
876 016614 012737 010000 001444 MOV      #TA2!30.,RMDAO ;TRACK = 2, SECTOR = 30.
877 016622          MOV      #FMT16,RMOFO ;16 BIT FORMAT
878 016622 012737 001057 001446 1$:
879 016630 012737 176774 001414 MOV      #559.,RMDCO   ;CYLINDER = 559.
880 016636 012737 101716 001416 MOV      #-258.*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2'S COMP)
      MOV      #BUFONE,RMBAO ;DATA BUFFER ADDRESS
  
```

```

881 016644      2$:
882
883
884 016644 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
      016650 154130      JSR PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      .WORD 154130      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF "SKI" OR "PIP" IS SET
      ;VERIFY RECALIBRATION
      ;GO TO 3$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 13$ IF ERROR
      016652 000404      BR 3$
      016654 000240      NOP
      016656 104000      EMT
      016660 000137 017220    JMP 13$
885 016664      3$:
886
887
888 016664 012737 000005 001412 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
      016672 012702 001555      MOV #SEEK!GO,RMCS10      ;CHANGE COMMAND TO SEEK
      016676 112722 000006      MOV #PUTINX,R2          ;WRITE REGISTER INDEX TABLE
      016702 112722 000034      MOV #RMDC,(R2)+
      016706 112722 000032      MOV #RMDA,(R2)+
      016712 112722 000000      MOV #RMOF,(R2)+
      016716 112722 000200      MOV #RMCS1,(R2)+
      016722 004737 036004      JSR PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      016726 000404      BR 4$                  ;GO TO 4$ IF NO ERROR
      016730 000240      NOP                      ;RETURN HERE IF ERROR
      016732 104000      EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
      016734 000137 017220    JMP 13$                  ;GO TO 13$ IF ERROR
897 016740      4$:
898
899
900 016740 004737 035450      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
      016744 004737 036356      JSR PC,GETSTS          ;SETUP FOR STATUS
      016750      JSR PC,TIMOUT          ;WAIT FOR SEEK TO COMPLETE
903
904
905 016750 004737 035534      ;GO READ SEEK STATUS
      016754 000404      JSR PC,GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
      016756 000240      BR 6$                  ;GO TO 6$ IF NO ERROR
      016760 104000      NOP                      ;RETURN HERE IF ERROR
      016762 000137 017220    EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
      016766      JMP 13$                  ;GO TO 13$ IF ERROR
906 016766      6$:
907
908
909 016766 004737 044174      ;VERIFY THE RESULTS OF THE SEEK COMMAND
      016772 000405      JSR PC,SEKSTS          ;GO VERIFY RESULTS OF SEEK OPERATION
      016774 000240      BR 7$                  ;GO TO 7$ IF NO ERROR
      016776 104000      NOP                      ;RETURN HERE IF ERROR
      017000 004736      EMT                      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      017002 000137 017220    JSR PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
      017006      JMP 13$                  ;GO TO 13$ IF ERROR
910 017006      7$:
911
912
      ;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
  
```

```

913 017006 012737 000073 001412      MOV      #RH!GO,RMCS10      ;READ HEADER AND DATA
914 017014 012702 001560              MOV      #PUTINX+3,R2      ;EXTEND REGISTER INDEX TABLE
915 017020 112722 000002              MOVB    #RMWC,(R2)+
916 017024 112722 000004              MOVB    #RMBA,(R2)+
917 017030 112722 000000              MOVB    #RMCS1,(R2)+
918 017034 112722 000200              MOVB    #200,(R2)+
919
920 017040 004737 036004              JSR     PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      017044 000404              BR     8$                  ;GO TO 8$ IF NO ERROR
      017046 000240              NOP
      017050 104000              EMT
      017052 000137 017220          JMP     13$                ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 13$ IF ERROR
8$:
921 017056
922
923      ;WAIT FOR READ COMMAND TO COMPLETE AND READ STATUS
924 017056 004737 036356              JSR     PC,TIMOUT          ;WAIT FOR COMMAND TO COMPLETE
925
926 017062 004737 035534              JSR     PC,GET             ;GO READ REGISTER(S) WITH GET SUBROUTINE
      017066 000404              BR     9$                  ;GO TO 9$ IF NO ERROR
      017070 000240              NOP
      017072 104000              EMT
      017074 000137 017220          JMP     13$                ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 13$ IF ERROR
927
928      ;CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
929      ;DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
930 017100
931 017100 032737 000040 001400          BIT     #SSE,RMER2I        ;IS 'SSE' BIT SET ?
932 017106 001030              BNE    12$                ;BR IF YES
933
934      ;VERIFY RESULTS OF READ COMMAND
935 017110 004737 036552              JSR     PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
      017114 000405              BR     10$                 ;GO TO 10$ IF NO ERROR
      017116 000240              NOP
      017120 104000              EMT
      017122 004736              JSR     PC,@(SP)+          ;GO CHECK FOR PRIMARY ERRORS
      017124 000137 017220          JMP     13$                ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 13$ IF ERROR
10$:
936 017130
937 017130 004737 051612              JSR     PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
      017134 000405              BR     11$                 ;GO TO 11$ IF NO ERROR
      017136 000240              NOP
      017140 104000              EMT
      017142 004736              JSR     PC,@(SP)+          ;GO VERIFY RESULTS OF DATA TRANSFER
      017144 000137 017220          JMP     13$                ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 13$ IF ERROR
11$:
938 017150
939 017150 004737 037404              JSR     PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
      017154 000405              BR     12$                 ;GO TO 12$ IF NO ERROR
      017156 000240              NOP
      017160 104000              EMT
      017162 004736              JSR     PC,@(SP)+          ;GO CHECK FOR SECONDARY ERRORS
      017164 000137 017220          JMP     13$                ;GO TO 12$ IF NO ERROR
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 13$ IF ERROR
12$:
940 017170
941 017170 032737 001000 001444          BIT     #SSEI,RMOFO        ;TEST 16 BIT MODE W/ SSEI ?
942 017176 001010              BNE    13$                 ;YES !!
943 017200 012737 001037 001420          MOV     #TA2!31.,RMDAO     ;TRACK = 2, SECTOR = 31.
944 017206 052737 001000 001444          BIS     #SSEI,RMOFO        ;SET SSEI BIT IN OFFSET AND
945 017214 000137 016622              JMP     1$                  ;TEST AGAIN
946 017220
13$:

```

947
948

 :*TEST 13 READ HEADER & DATA W/ MID TRANSFER SEEK

 TST13:

017220
 017220 000004
 017222 000240
 017224 012706 001100
 017230 013700 001276
 017234 013701 001466
 017240 012737 000013 001226

```

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
  
```

949
950

951 017246 013737 001334 001420
 952 017254 112737 000036 001420
 953 017262 012737 010000 001444
 954 017270
 955 017270 012737 001057 001446
 956 017276 012737 176774 001414
 957 017304 012737 101716 001416
 958 017312
 959

```

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
MOV LSTRK,RMDAO ;START LAST TRACK AND
MOVB #30,RMDAO  ;LAST SECTOR
MOV #FMT16,RMOFO ;16 BIT FORMAT
1$:
MOV #559,RMDCO  ;CYLINDER = 559.
MOV #-258,*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2'S COMP)
MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
2$:
  
```

960
961

017312 004737 031456
 017316 154130

```

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
.WORD 154130  ;TASK DESCRIPTOR AS FOLLOWS:
              ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
              ;CLEAR CONTROLLER & SELECT DEVICE
              ;VERIFY CONTROLLER CLEAR OPERATION
              ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
              ;VERIFY PACK ACKNOWLEDGE
              ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
              ;VERIFY RECALIBRATION
              ;GO TO 3$ IF NO ERROR
              ;RETURN HERE IF ERROR
              ;ERROR # DEFINED BY TSTPRP SUBROUTINE
              ;GO TO 13$ IF ERROR
BR          3$
NOP
EMT
JMP        13$
3$:
  
```

962
963

964
 965 017332 012737 000005 001412
 966 017340 012702 001555
 967 017344 112722 000006
 968 017350 112722 000034
 969 017354 112722 000032
 970 017360 112722 000000
 971 017364 112722 000200
 972

```

;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)+
  
```

973
974

017370 004737 036004
 017374 000404
 017376 000240
 017400 104000
 017402 000137 017674
 974 017406

```

JSR PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR          4$  ;GO TO 4$ IF NO ERROR
NOP           ;RETURN HERE IF ERROR
EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
JMP        13$ ;GO TO 13$ IF ERROR
4$:
  
```

975
976

977 017406 004737 035450
 978 017412 004737 036356

```

;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
  
```

```

979 017416          5$:
980
981                ;GO READ SEEK STATUS
982 017416 004737 035534      JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
    017422 000404          BR      6$          ;GO TO 6$ IF NO ERROR
    017424 000240          NOP          ;RETURN HERE IF ERROR
    017426 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
    017430 000137 017674    JMP      13$         ;GO TO 13$ IF ERROR
983 017434          6$:
984
985                ;VERIFY THE RESULTS OF THE SEEK COMMAND
986 017434 004737 044174      JSR      PC,SEKSTS   ;GO VERIFY RESULTS OF SEEK OPERATION
    017440 000405          BR      7$          ;GO TO 7$ IF NO ERROR
    017442 000240          NOP          ;RETURN HERE IF ERROR
    017444 104000          EMT          ;ERROR # DEFINED BY SEKSTS SUBROUTINE
    017446 004736          JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    017450 000137 017674    JMP      13$         ;GO TO 13$ IF ERROR
987 017454          7$:
988
989                ;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
990 017454 012737 000073 001412  MOV      #RH!GO,RMCS10 ;READ HEADER AND DATA
991 017462 012702 001560          MOV      #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
992 017466 112722 000002          MOVB     #RMWC,(R2)+
993 017472 112722 000004          MOVB     #RMBA,(R2)+
994 017476 112722 000000          MOVB     #RMCS1,(R2)+
995 017502 112722 000200          MOVB     #200,(R2)+
996
997 017506 004737 036004      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    017512 000404          BR      8$          ;GO TO 8$ IF NO ERROR
    017514 000240          NOP          ;RETURN HERE IF ERROR
    017516 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
    017520 000137 017674    JMP      13$         ;GO TO 13$ IF ERROR
998 017524          8$:
999
1000               ;WAIT FOR READ COMMAND TO COMPLETE AND READ STATUS
1001 017524 004737 036356      JSR      PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
1002
1003               ;GO READ REGISTER(S) WITH GET SUBROUTINE
    017530 004737 035534      JSR      PC,GET      ;GO TO 9$ IF NO ERROR
    017534 000404          BR      9$          ;RETURN HERE IF ERROR
    017536 000240          NOP          ;ERROR # DEFINED BY GET SUBROUTINE
    017540 104000          EMT          ;GO TO 13$ IF ERROR
    017542 000137 017674    JMP      13$
1004
1005               ;CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
1006               ;DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
1007 017546          9$:
1008 017546 032737 000040 001400  BIT      #SSE,RMER2I  ;IS 'SSE' BIT SET ?
1009 017554 001030          BNE     12$         ;BR IF YES
1010
1011               ;VERIFY RESULTS OF READ COMMAND
1012 017556 004737 036552      JSR      PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
    017562 000405          BR      10$        ;GO TO 10$ IF NO ERROR
    017564 000240          NOP          ;RETURN HERE IF ERROR
    017566 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
    017570 004736          JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    017572 000137 017674    JMP      13$         ;GO TO 13$ IF ERROR
1013 017576          10$:

```



```

1014 017576 004737 051612      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      017602 000405          BR      11$          ;GO TO 11$ IF NO ERROR
      017604 000240          NOP          ;RETURN HERE IF ERROR
      017606 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
      017610 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      017612 000137 017674      JMP      13$          ;GO TO 13$ IF ERROR
1015 017616                    11$:
1016 017616 004737 037404      JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      017622 000405          BR      12$          ;GO TO 12$ IF NO ERROR
      017624 000240          NOP          ;RETURN HERE IF ERROR
      017626 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
      017630 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      017632 000137 017674      JMP      13$          ;GO TO 13$ IF ERROR
1017 017636                    12$:
1018 017636 032737 001000 001444  BIT    #SSEI,RMOFO   ;TEST 16 BIT MODE W/ SSEI SET ?
1019 017644 001013          BNE    13$          ;YES !!
1020 017646 013737 001334 001420  MOV    LSTRK,RMDAO   ;START LAST TRACK AND
1021 017654 112737 000037 001420  MOVB   #31,RMDAO    ;LAST SECTOR
1022 017662 052737 001000 001444  BIS    #SSEI,RMOFO   ;SET SSEI IN OFFSET AND
1023 017670 000137 017270      JMP      1$          ;TEST AGAIN.
1024 017674                    13$:
1025
1026
;*****
;*TEST 14      FORMAT W/ IMPLIED SEEK
;*****
TST14:
      017674 000004          SCOPE          ;SCOPE CALL
      017676 000240          NOP          ;START OF TEST
      017700 012706 001100      MOV    #STACK,SP    ;INITIALIZE STACK POINTER
      017704 013700 001276      MOV    $BASE,R0     ;R0 = UNIBUS ADDRESS
      017710 013701 001466      MOV    TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
      017714 012737 000014 001226  MOV    #14,$TESTN   ;SET TEST NUMBER IN APT MAIL BOX

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1027
1028
1029 017722 012737 001057 001446  MOV    #559,RMDCO   ;CYLINDER = 559.
1030 017730 012737 001000 001420  MOV    #TA2,RMDAO   ;TRACK = 2, SECTOR = 0
1031 017736 012737 010000 001444  MOV    #FMT16,RMOFO ;16 BIT FORMAT
1032 017744 012737 176774 001414  MOV    #-258.*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2'S COMP)
1033 017752 012737 101716 001416  MOV    #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1034 017760 012737 000062 001412  MOV    #WH,RMCS10   ;WRITE HEADER AND DATA
1035
1036
;VERIFY THAT SECTOR IS NOT BAD
      017766 004737 032406      JSR    PC,BADSCT    ;CALL BAD SECTOR MODULE
      017772 000405          BR      1$          ;GO TO 1$ IF NO ERROR
      017774 104401 063446      TYPE   ,SCTMSG     ;TYPE BAD SECTOR MESSAGE
      020000 104000          EMT          ;ERROR # DEFINED BY BADSCT SUBROUTINE
      020002 000137 020512      JMP      18$        ;GO TO 18$ IF ERROR
1037 020006                    1$:
1038 020006 012737 065116 001174  MOV    #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
1039 020014 012737 000001 001176  MOV    #1,$TMP1
1040 020022 004737 034640      JSR    PC,GENBUF    ;GO GENERATE DATA BUFFER
1041 020026                    2$:
1042
1043
1044 020026 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
      020032 154130          JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      .WORD 154130          .WORD 154130        ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
  
```

```

                                :CLEAR CONTROLLER & SELECT DEVICE
                                :VERIFY CONTROLLER CLEAR OPERATION
                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                :VERIFY PACK ACKNOWLEDGE
                                :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                :VERIFY RECALIBRATION
                                :GO TO 3$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 18$ IF ERROR

020034 000404                BR      3$
020036 000240                NOP
020040 104000                EMT
020042 000137 020512        JMP      18$
1045 020046                3$:
1046
1047                ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE OFF CYLINDER
1048 020046 013737 001446 020514  MOV    RMDCO,19$           ;SAVE CYLINDER ADDRESS
1049 020054 012737 000000 001446  MOV    #0,RMDCO           ;SEEK TO FIRST CYLINDER
1050 020062 012737 000005 001412  MOV    #SEEK!GO,RMCS10    ;CHANGE COMMAND TO SEEK
1051 020070 012702 001555                MOV    #PUTINX,R2         ;WRITE REGISTER INDEX TABLE
1052 020074 112722 000006                MOVB   #RMDA,(R2)+
1053 020100 112722 000034                MOVB   #RMDC,(R2)+
1054 020104 112722 000032                MOVB   #RMOF,(R2)+
1055 020110 112722 000000                MOVB   #RMCS1,(R2)+
1056 020114 112722 000200                MOVB   #200,(R2)+
1057
1058 020120 064737 036004        JSR    PC,PUT             ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      4$           ;GO TO 4$ IF NO ERROR
                                NOP           ;RETURN HERE IF ERROR
                                EMT           ;ERROR # DEFINED BY PUT SUBROUTINE
                                JMP      18$        ;GO TO 18$ IF ERROR
1059 020136                4$:
1060
1061                ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
1062 020136 004737 035450        JSR    PC,GETSTS         ;SETUP FOR STATUS
1063 020142 004737 036356        JSR    PC,TIMOUT        ;WAIT FOR SEEK TO COMPLETE
1064 020146                5$:
1065
1066                ;GO READ SEEK STATUS
1067 020146 004737 035534        JSR    PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR      6$           ;GO TO 6$ IF NO ERROR
                                NOP           ;RETURN HERE IF ERROR
                                EMT           ;ERROR # DEFINED BY GET SUBROUTINE
                                JMP      18$        ;GO TO 18$ IF ERROR
1068 020164                6$:
1069
1070                ;VERIFY THE RESULTS OF THE SEEK COMMAND
1071 020164 004737 044174        JSR    PC,SEKSTS        ;GO VERIFY RESULTS OF SEEK OPERATION
                                BR      7$           ;GO TO 7$ IF NO ERROR
                                NOP           ;RETURN HERE IF ERROR
                                EMT           ;ERROR # DEFINED BY SEKSTS SUBROUTINE
                                JSR    PC,a(SP)+    ;GO BACK FOR MORE ERROR CHECKS
                                JMP      18$        ;GO TO 18$ IF ERROR
1072 020204                7$:
1073
1074                ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1075 020204 013737 020514 001446  MOV    19$,RMDCO         ;RESTORE DISK ADDRESS
1076 020212 012737 000063 001412  MOV    #WH!GO,RMCS10    ;WRITE HEADER AND DATA
1077 020220 012702 001560                MOV    #PUTINX+3,R2     ;EXTEND REGISTER INDEX TABLE
1078 020224 112722 000002                MOVB   #RMWC,(R2)+

```

```

1079 020230 112722 000004      MOVB  #RMBA,(R2)+
1080 020234 112722 000000      MOVB  #RMCS1,(R2)+
1081 020240 112722 000200      MOVB  #200,(R2)+
1082
1083 020244 004737 036004      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020250 000404          BR    8$             ;GO TO 8$ IF NO ERROR
      020252 000240          NOP                    ;RETURN HERE IF ERROR
      020254 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      020256 000137 020512      JMP   18$           ;GO TO 18$ IF ERROR
1034 020262      8$:
1085
1086      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
1087 020262 004737 036356      JSR   PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1088
1089 020266 004737 035534      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020272 000404          BR    9$             ;GO TO 9$ IF NO ERROR
      020274 000240          NOP                    ;RETURN HERE IF ERROR
      020276 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      020300 000137 020512      JMP   18$           ;GO TO 18$ IF ERROR
1090 020304      9$:
1091
1092      ;VERIFY RESULTS OF WRITE COMMAND
1093 020304 004737 036552      JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      020310 000405          BR    10$            ;GO TO 10$ IF NO ERROR
      020312 000240          NOP                    ;RETURN HERE IF ERROR
      020314 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      020316 004736          JSR   PC,a(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      020320 000137 020512      JMP   18$           ;GO TO 18$ IF ERROR
1094 020324      10$:
1095 020324 004737 051612      JSR   PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      020330 000405          BR    11$            ;GO TO 11$ IF NO ERROR
      020332 000240          NOP                    ;RETURN HERE IF ERROR
      020334 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      020336 004736          JSR   PC,a(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      020340 000137 020512      JMP   18$           ;GO TO 18$ IF ERROR
1096 020344      11$:
1097 020344 004737 037404      JSR   PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      020350 000405          BR    12$            ;GO TO 12$ IF NO ERROR
      020352 000240          NOP                    ;RETURN HERE IF ERROR
      020354 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      020356 004736          JSR   PC,a(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      020360 000137 020512      JMP   18$           ;GO TO 18$ IF ERROR
1098 020364      12$:
1099
1100      ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
1101 020364 012737 000053 001412  MOV   #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
1102
1103 020372 004737 036004      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020376 000404          BR    13$            ;GO TO 13$ IF NO ERROR
      020400 000240          NOP                    ;RETURN HERE IF ERROR
      020402 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      020404 000137 020512      JMP   18$           ;GO TO 18$ IF ERROR
1104 020410      13$:
1105
1106      ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
1107 020410 004737 036356      JSR   PC,TIMOUT      ;WAIT FOR WRITE CHECK TO FINISH
1108

```

```

1109 020414 004737 035534      JSR   PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020420 000404          BR    14$             ;GO TO 14$ IF NO ERROR
      020422 000240          NOP                    ;RETURN HERE IF ERROR
      020424 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      020426 000137 020512    JMP   18$             ;GO TO 18$ IF ERROR
  
```

1110 020432 14\$:

1111
 1112 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION

```

1113 020432 004737 036552      JSR   PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      020436 000405          BR    15$             ;GO TO 15$ IF NO ERROR
      020440 000240          NOP                    ;RETURN HERE IF ERROR
      020442 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      020444 004736          JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      020446 000137 020512    JMP   18$             ;GO TO 18$ IF ERROR
  
```

1114 020452 15\$:

```

1115 020452 004737 051612      JSR   PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      020456 000405          BR    16$             ;GO TO 16$ IF NO ERROR
      020460 000240          NOP                    ;RETURN HERE IF ERROR
      020462 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      020464 004736          JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      020466 000137 020512    JMP   18$             ;GO TO 18$ IF ERROR
  
```

1116 020472 16\$:

```

1117 020472 004737 037404      JSR   PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
      020476 000405          BR    17$             ;GO TO 17$ IF NO ERROR
      020500 000240          NOP                    ;RETURN HERE IF ERROR
      020502 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      020504 004736          JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      020506 000137 020512    JMP   18$             ;GO TO 18$ IF ERROR
  
```

1118 020512 17\$:

1119
 1120 020512 000401 18\$: BR 20\$

1121
 1122 020514 000000 19\$: .WORD 0 ;TEMPORARY STORAGE

1123
 1124 020516 20\$:

1125
 1126

```

:*****
:*TEST 15      FORMAT EACH SECTOR ADDRESS
:*****
TST15:
  
```

```

020516 000004      SCOPE      ;SCOPE CALL
020516 000240      NOP          ;START OF TEST
020522 012706 001100  MOV   #STACK,SP ;INITIALIZE STACK POINTER
020526 013700 001276  MOV   $BASE,R0   ;R0 = UNIBUS ADDRESS
020532 013701 001466  MOV   TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
020536 012737 000015 001226  MOV   #15,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
  
```

1127
 1128 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER

```

1129 020544 012737 001057 001446  MOV   #559.,RMDCO ;CYLINDER = 559.
1130 020552 012737 001000 001420  MOV   #TA2,RMDAO  ;TRACK = 2, SECTOR = 0
1131 020560 012737 011000 001444  MOV   #FMT16!SSE1,RMOFO ;16 BIT FORMAT
  
```

1132 020566 1\$:

```

1133 020566 012737 177376 001414  MOV   #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
1134 020574 012737 101716 001416  MOV   #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1135 020602 012737 000062 001412  MOV   #WH,RMCS10  ;WRITE HEADER AND DATA
  
```

1136
 1137

;VERIFY THAT SECTOR IS NOT BAD

```

020610 004737 032406      JSR   PC,BADSCT      ;CALL BAD SECTOR MODULE
020614 000405              BR    2$             ;GO TO 2$ IF NO ERROR
020616 104401 063446      TYPE  ,SCTMSG        ;TYPE BAD SECTOR MESSAGE
020622 104000              EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
020624 000137 021260      JMP   16$           ;GO TO 16$ IF ERROR
1138 020630              2$:
1139 020630 012737 001420 001174  MOV   #RMDAO,$TMP0    ;USE SECTOR FOR DATA PATTERN
1140 020636 012737 000001 001176  MOV   #1,$TMP1
1141 020644 004737 034640      JSR   PC,GENBUF      ;GO GENERATE DATA BUFFER
1142 020650              3$:
1143
1144      ;PREPARE DEVICE FOR DATA TRANSFER
1145 020650 004737 031456      JSR   PC,TSTPRP     ;PREPARE DEVICE FOR TEST
020654 154130      .WORD 154130        ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 4$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 16$ IF ERROR
020656 000404      BR    4$
020660 000240      NOP
020662 104000      EMT
020664 000137 021260      JMP   16$
1146 020670              4$:
1147
1148      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1149 020670 012737 000063 001412  MOV   #WH!GO,RMCS10  ;WRITE HEADER AND DATA
1150 020676 012702 001555      MOV   #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
1151 020702 112722 000006      MOVB  #RMDA,(R2)+
1152 020706 112722 000034      MOVB  #RMDC,(R2)+
1153 020712 112722 000032      MOVB  #RMOF,(R2)+
1154 020716 112722 000002      MOVB  #RMWC,(R2)+
1155 020722 112722 000004      MOVB  #RMBA,(R2)+
1156 020726 112722 000000      MOVB  #RMCS1,(R2)+
1157 020732 112722 000200      MOVB  #200,(R2)+
1158
1159 020736 004737 036004      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
020742 000404      BR    5$           ;GO TO 5$ IF NO ERROR
020744 000240      NOP                ;RETURN HERE IF ERROR
020746 104000      EMT                ;ERROR # DEFINED BY PUT SUBROUTINE
020750 000137 021260      JMP   16$           ;GO TO 16$ IF ERROR
1160 020754              5$:
1161
1162      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1163 020754 004737 035450      JSR   PC,GETSTS
1164 020760 004737 036356      JSR   PC,TIMOUT    ;WAIT FOR COMMAND TO COMPLETE
1165
1166 020764 004737 035534      JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
020770 000404      BR    6$           ;GO TO 6$ IF NO ERROR
020772 000240      NOP                ;RETURN HERE IF ERROR
020774 104000      EMT                ;ERROR # DEFINED BY GET SUBROUTINE
020776 000137 021260      JMP   16$           ;GO TO 16$ IF ERROR
1167 021002              6$:
1168
1169      ;VERIFY RESULTS OF WRITE COMMAND

```

1170	021002	004737	036552		JSR	PC,PRIERR		:GO CHECK FOR PRIMARY ERRORS
	021006	000405			BR	7\$:GO TO 7\$ IF NO ERROR
	021010	000240			NOP			:RETURN HERE IF ERROR
	021012	104000			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	021014	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021016	000137	021260		JMP	16\$:GO TO 16\$ IF ERROR
1171	021022			7\$:				
1172	021022	004737	051612		JSR	PC,DTASTS		:GO VERIFY RESULTS OF DATA TRANSFER
	021026	000405			BR	8\$:GO TO 8\$ IF NO ERROR
	021030	000240			NOP			:RETURN HERE IF ERROR
	021032	104000			EMT			:ERROR # DEFINED BY DTASTS SUBROUTINE
	021034	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021036	000137	021260		JMP	16\$:GO TO 16\$ IF ERROR
1173	021042			8\$:				
1174	021042	004737	037404		JSR	PC,SECERR		:GO CHECK FOR SECONDARY ERRORS
	021046	000405			BR	9\$:GO TO 9\$ IF NO ERROR
	021050	000240			NOP			:RETURN HERE IF ERROR
	021052	104000			EMT			:ERROR # DEFINED BY SECERR SUBROUTINE
	021054	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021056	000137	021260		JMP	16\$:GO TO 16\$ IF ERROR
1175	021062			9\$:				
1176								
1177								
1178	021062	012737	000073	001412	:READ HEADER AND DATA FOR SECTOR JUST WRITTEN			
1179	021070	012737	102722	001416	MOV	#RH!GO,RMCS10		:READ HEADER & DATA COMMAND
1180					MOV	#BUFTWO,RMBAO		:CHANGE BUS ADDRESS
1181	021076	004737	036004		JSR	PC,PUT		:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	021102	000404			BR	10\$:GO TO 10\$ IF NO ERROR
	021104	000240			NOP			:RETURN HERE IF ERROR
	021106	104000			EMT			:ERROR # DEFINED BY PUT SUBROUTINE
	021110	000137	021260		JMP	16\$:GO TO 16\$ IF ERROR
1182	021114			10\$:				
1183								
1184								
1185	021114	004737	036356		:WAIT FOR READ TO COMPLETE AND GET STATUS			
1186					JSR	PC,TIMOUT		:WAIT FOR READ TO COMPLETE
1187	021120	004737	035534		JSR	PC,GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
	021124	000404			BR	11\$:GO TO 11\$ IF NO ERROR
	021126	000240			NOP			:RETURN HERE IF ERROR
	021130	104000			EMT			:ERROR # DEFINED BY GET SUBROUTINE
	021132	000137	021260		JMP	16\$:GO TO 16\$ IF ERROR
1188	021136			11\$:				
1189								
1190								
1191	021136	004737	036552		:VERIFY THE RESULTS OF READ OPERATION			
	021142	000405			JSR	PC,PRIERR		:GO CHECK FOR PRIMARY ERRORS
	021144	000240			BR	12\$:GO TO 12\$ IF NO ERROR
	021146	104000			NOP			:RETURN HERE IF ERROR
	021150	004736			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	021152	000137	021260		JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021156				JMP	16\$:GO TO 16\$ IF ERROR
1192	021156			12\$:				
1193	021156	004737	051612		JSR	PC,DTASTS		:GO VERIFY RESULTS OF DATA TRANSFER
	021162	000405			BR	13\$:GO TO 13\$ IF NO ERROR
	021164	000240			NOP			:RETURN HERE IF ERROR
	021166	104000			EMT			:ERROR # DEFINED BY DTASTS SUBROUTINE
	021170	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021172	000137	021260		JMP	16\$:GO TO 16\$ IF ERROR

```

1194 021176
1195 021176 004737 037404
      021202 000405
      021204 000240
      021206 104000
      021210 004736
      021212 000137 021260
1196 021216
1197
1198
1199 021216 004737 035106
      021222 101716
      021224 102722
      021226 000404
      021230 000240
      021232 104000
      021234 000137 021260
1200
1201
1202 021240
1203 021240 005237 001420
1204 021244 122737 000037 001420
1205 021252 103402
1206 021254 000137 020566
1207 021260
1208
1209

      021260
      021260 000004
      021262 000240
      021264 012706 001100
      021270 013700 001276
      021274 013701 001466
      021300 012737 000016 001226
1210
1211
1212 021306 012737 001060 001446
1213 021314 012737 000000 001420
1214 021322 012737 010000 001444
1215 021330
1216 021330 012737 177376 001414
1217 021336 012737 101716 001416
1218 021344 012737 000062 001412
1219
1220
      021352 004737 032406
      021356 000405
      021360 104401 063446
      021364 104000
      021366 000137 022022
1221 021372
1222 021372 012737 001420 001174
1223 021400 012737 000001 001176
1224 021406 004737 034640
1225 021412

13$:
      JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
      BR 14$ ;GO TO 14$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP 16$ ;GO TO 16$ IF ERROR

14$:
;VERIFY DATA
      JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
      .WORD BUFO1E ;STARTING ADDRESS OF WRITE BUFFER
      .WORD BUFTWO ;STARTING ADDRESS OF READ BUFFER
      BR 15$ ;GO TO 15$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      EMT ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      JMP 16$ ;GO TO 16$ IF ERROR

;INCREMENT ADDRESS AND FORMAT NEXT SECTOR
15$:
      INC RMDAO ;ADVANCE SECTOR COUNT
      CMPB #31.,RMDAO ;DONE ALL SECTORS??
      BLO 16$ ;YES!!
      JMP 1$ ;GO DO NEXT SECTOR

16$:
;*****
;*TEST 16 FORMAT EACH TRACK ADDRESS
;*****
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

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
      MOV #560.,RMDCO ;CYLINDER = 560.
      MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
      MOV #FMT16,RMOFO ;16 BIT FORMAT

1$:
      MOV #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
      MOV #BUFO1E,RMBAO ;DATA BUFFER ADDRESS
      MOV #WH,RMC$10 ;WRITE HEADER AND DATA

;VERIFY THAT SECTOR IS NOT BAD
      JSR PC,BADSCT ;CALL BAD SECTOR MODULE
      BR 2$ ;GO TO 2$ IF NO ERROR
      TYPE .SCTMSG ;TYPE BAD SECTOR MESSAGE
      EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
      JMP 16$ ;GO TO 16$ IF ERROR

2$:
      MOV #RMDAO,$TMP0 ;USE TRACK FOR DATA PATTERN
      MOV #1,$TMP1
      JSR PC,GENBUF ;GO GENERATE DATA BUFFER

3$:

```

```

1226
1227
1228 021412 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
      021416 154130      JSR PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      .WORD 154130      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                          ;CLEAR CONTROLLER & SELECT DEVICE
                                          ;VERIFY CONTROLLER CLEAR OPERATION
                                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                          ;VERIFY PACK ACKNOWLEDGE
                                          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                          ;VERIFY RECALIBRATION
                                          ;GO TO 4$ IF NO ERROR
                                          ;RETURN HERE IF ERROR
                                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                          ;GO TO 16$ IF ERROR

      021420 000404      BR 4$
      021422 000240      NOP
      021424 104000      EMT
      021426 000137 022022    JMP 16$
1229 021432      4$:
1230
1231      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1232 021432 012737 000063 001412    MOV #WH!GO,RMCS10      ;WRITE HEADER AND DATA
1233 021440 012702 001555      MOV #PUTINX,R2        ;WRITE REGISTER INDEX TABLE
1234 021444 112722 000006      MOVB #RMDA,(R2)+
1235 021450 112722 000034      MOVB #RMDC,(R2)+
1236 021454 112722 000032      MOVB #RMOF,(R2)+
1237 021460 112722 000002      MOVB #RMWC,(R2)+
1238 021464 112722 000004      MOVB #RMBA,(R2)+
1239 021470 112722 000000      MOVB #RMCS1,(R2)+
1240 021474 112722 000200      MOVB #200,(R2)+
1241
1242 021500 004737 036004      JSR PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021504 000404      BR 5$                ;GO TO 5$ IF NO ERROR
      021506 000240      NOP
      021510 104000      EMT
      021512 000137 022022    JMP 16$                ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 16$ IF ERROR
1243 021516      5$:
1244
1245      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1246 021516 004737 035450      JSR PC,GETSTS
1247 021522 004737 036356      JSR PC,TIMOUT        ;WAIT FOR COMMAND TO COMPLETE
1248
1249 021526 004737 035534      JSR PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021532 000404      BR 6$                ;GO TO 6$ IF NO ERROR
      021534 000240      NOP
      021536 104000      EMT
      021540 000137 022022    JMP 16$                ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 16$ IF ERROR
1250 021544      6$:
1251
1252      ;VERIFY RESULTS OF WRITE COMMAND
1253 021544 004737 036552      JSR PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      021550 000405      BR 7$                ;GO TO 7$ IF NO ERROR
      021552 000240      NOP
      021554 104000      EMT
      021556 004736      JSR PC,@(SP)+
      021560 000137 022022    JMP 16$                ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PRIERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 16$ IF ERROR
1254 021564      7$:
1255 021564 004737 051612      JSR PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      021570 000405      BR 8$                ;GO TO 8$ IF NO ERROR
      021572 000240      NOP
      ;RETURN HERE IF ERROR
  
```


	021574	104000			EMT			:ERROR # DEFINED BY DTASTS SUBROUTINE
	021576	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021600	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1256	021604			8\$:				
1257	021604	004737	037404		JSR	PC,SECERR		:GO CHECK FOR SECONDARY ERRORS
	021610	000405			BR	9\$:GO TO 9\$ IF NO ERROR
	021612	000240			NOP			:RETURN HERE IF ERROR
	021614	104000			EMT			:ERROR # DEFINED BY SECERR SUBROUTINE
	021616	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021620	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1258	021624			9\$:				
1259								
1260								
1261	021624	012737	000073	001412	MOV	#RH!GO,RMCS10		:READ HEADER & DATA COMMAND
1262	021632	012737	102722	001416	MOV	#BUFTWO,RMBAO		:CHANGE BUS ADDRESS
1263								
1264	021640	004737	036004		JSR	PC,PUT		:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	021644	000404			BR	10\$:GO TO 10\$ IF NO ERROR
	021646	000240			NOP			:RETURN HERE IF ERROR
	021650	104000			EMT			:ERROR # DEFINED BY PUT SUBROUTINE
	021652	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1265	021656			10\$:				
1266								
1267								
1268	021656	004737	036356		JSR	PC,TIMOUT		:WAIT FOR READ TO COMPLETE
1269								
1270	021662	004737	035534		JSR	PC,GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
	021666	000404			BR	11\$:GO TO 11\$ IF NO ERROR
	021670	000240			NOP			:RETURN HERE IF ERROR
	021672	104000			EMT			:ERROR # DEFINED BY GET SUBROUTINE
	021674	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1271	021700			11\$:				
1272								
1273								
1274	021700	004737	036552		JSR	PC,PRIERR		:GO CHECK FOR PRIMARY ERRORS
	021704	000405			BR	12\$:GO TO 12\$ IF NO ERROR
	021706	000240			NOP			:RETURN HERE IF ERROR
	021710	104000			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	021712	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021714	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1275	021720			12\$:				
1276	021720	004737	051612		JSR	PC,DTASTS		:GO VERIFY RESULTS OF DATA TRANSFER
	021724	000405			BR	13\$:GO TO 13\$ IF NO ERROR
	021726	000240			NOP			:RETURN HERE IF ERROR
	021730	104000			EMT			:ERROR # DEFINED BY DTASTS SUBROUTINE
	021732	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021734	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1277	021740			13\$:				
1278	021740	004737	037404		JSR	PC,SECERR		:GO CHECK FOR SECONDARY ERRORS
	021744	000405			BR	14\$:GO TO 14\$ IF NO ERROR
	021746	000240			NOP			:RETURN HERE IF ERROR
	021750	104000			EMT			:ERROR # DEFINED BY SECERR SUBROUTINE
	021752	004736			JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
	021754	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR
1279	021760			14\$:				
1280								
1281								

:VERIFY DATA

1282	021760	004737	035106		JSR	PC,CMPBUF	:GO COMPARE WRITE, READ DATA BJJFFERS
	021764	101716			.WORD	BUFONE	:STARTING ADDRESS OF WRITE BUFFER
	021766	102722			.WORD	BUFTWO	:STARTING ADDRESS OF READ BUFFER
	021770	000404			BR	15\$:GO TO 15\$ IF NO ERROR
	021772	000240			NOP		:RETURN HERE IF ERROR
	021774	104000			EMT		:ERROR # DEFINED BY CMPBUF SUBROUTINE
	021776	000137	022022		JMP	16\$:GO TO 16\$ IF ERROR

1283
 1284 :INCREMENT ADDRESS AND FORMAT NEXT TRACK

1285	022002				15\$:		
1286	022002	105237	001421		INCB	RMDAO+1	:ADVANCE TRACK COUNT
1287	022006	123737	001421	001335	CMPB	RMDAO+1,LSTRK+1	:LAST TRACK ?
1288	022014	101002			BHI	16\$:YES!!
1289	022016	000137	021330		JMP	1\$:GO DO NEXT SECTOR

1290 022022 16\$:

1291
 1292
 :*****
 :*TEST 17 READ HEADER & DATA IN LAST SECTOR
 :*****
 TST17:

022022					SCOPE		:SCOPE CALL
022022	000004				NOP		:START OF TEST
022024	000240				MOV	#STACK,SP	:INITIALIZE STACK POINTER
022026	012706	001100			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
022032	013700	001276			MOV	TSTQUE,R1	:(R1) = DEVICE BEING TESTED
022036	013701	001466			MOV	#17,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
022042	012737	000017	001226				

1293
 1294 :SETUP PARAMETERS FOR READING LAST SECTOR

1295	022050	013737	001334	001420	MOV	LSTRK,RMDAO	:SET LAST TRACK AND
1296	022056	112737	000036	001420	MOV	#30.,RMDAO	:LAST SECTOR
1297	022064	012737	010000	001444	MOV	#FMT16,RMOFO	:16 BIT FORMAT
1298	022072				1\$:		
1299	022072	012737	001060	001446	MOV	#560.,RMDCO	:LAST CYLINDER
1300	022100	012737	177376	001414	MOV	#-258.,RMWCO	:2 + 256 WORDS (2'S COMP)
1301	022106	012737	102722	001416	MOV	#BUFTWO,RMBAO	:DATA BUFFER ADDRESS
1302	022114	012737	000073	001412	MOV	#RH!GO,RMCS10	:READ HEADER AND DATA

1303
 1304 :PREPARE DEVICE FOR DATA TRANSFER

1305	022122	004737	031456		JSR	PC,TSTPRP	:PREPARE DEVICE FOR TEST
	022126	154130			.WORD	154130	:TASK DESCRIPTOR AS FOLLOWS:
							:SELECT DEVICE & VERIFY DEVICE AVAILABLE
							:CLEAR CONTROLLER & SELECT DEVICE
							:VERIFY CONTROLLER CLEAR OPERATION
							:PACK ACKNOWLEDGE IF VOLUME NOT VALID
							:VERIFY PACK ACKNOWLEDGE
							:RECALIBRATE IF 'SKI' OR 'PIP' IS SET
							:VERIFY RECALIBRATION
							:GO TO 2\$ IF NO ERROR
							:RETURN HERE IF ERROR
							:ERROR # DEFINED BY TSTPRP SUBROUTINE
							:GO TO 9\$ IF ERROR

1306	022142				2\$:		
1307	022142	012702	001555		MOV	#PUTINX,R2	:WRITE REGISTER INDEX TABLE
1308	022146	112722	000006		MOVB	#RMDA,(R2)+	
1309	022152	112722	000034		MOVB	#RMDC,(R2)+	
1310	022156	112722	000032		MOVB	#RMOF,(R2)+	
1311	022162	112722	000004		MOVB	#RMBA,(R2)+	

```

1312 022166 112722 000002          MOVB  #RMWC,(R2)+
1313 022172 112722 000000          MOVB  #RMCS1,(R2)+
1314 022176 112722 000200          MOVB  #200,(R2)+          ; TERMINATOR
1315 022202          3$:
1316
1317          ;READ HEADER AND DATA OF LAST SECTOR
1318 022202 004737 036004          JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          022206 000404          BR    4$             ;GO TO 4$ IF NO ERROR
          022210 000240          NOP                    ;RETURN HERE IF ERROR
          022212 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
          022214 000137 022374          JMP   9$             ;GO TO 9$ IF ERROR
1319 022220          4$:
1320
1321          ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1322 022220 004737 035450          JSR   PC,GETSTS
1323 022224 004737 036356          JSR   PC,TIMOUT          ;WAIT FOR READ TO COMPLETE
1324
1325 022230 004737 035534          JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
          022234 000404          BR    5$             ;GO TO 5$ IF NO ERROR
          022236 000240          NOP                    ;RETURN HERE IF ERROR
          022240 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
          022242 000137 022374          JMP   9$             ;GO TO 9$ IF ERROR
1326
1327          ;CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
1328          ;DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
1329 022246          5$:
1330 022246 032737 000040 001400          BIT   #SSE,RMER2I          ;IS 'SSE' BIT SET ?
1331 022254 001030          BNE   8$             ;BR IF YES
1332
1333          ;VERIFY THE RESULTS OF READ OPERATION
1334 022256 004737 036552          JSR   PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
          022262 000405          BR    6$             ;GO TO 6$ IF NO ERROR
          022264 000240          NOP                    ;RETURN HERE IF ERROR
          022266 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
          022270 004736          JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
          022272 000137 022374          JMP   9$             ;GO TO 9$ IF ERROR
1335 022276          6$:
1336 022276 004737 051612          JSR   PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
          022302 000405          BR    7$             ;GO TO 7$ IF NO ERROR
          022304 000240          NOP                    ;RETURN HERE IF ERROR
          022306 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
          022310 004736          JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
          022312 000137 022374          JMP   9$             ;GO TO 9$ IF ERROR
1337 022316          7$:
1338 022316 004737 037404          JSR   PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
          022322 000405          BR    8$             ;GO TO 8$ IF NO ERROR
          022324 000240          NOP                    ;RETURN HERE IF ERROR
          022326 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
          022330 004736          JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
          022332 000137 022374          JMP   9$             ;GO TO 9$ IF ERROR
1339 022336          8$:
1340 022336 032737 001000 001444          BIT   #SSEI,RMOFO          ;TEST 16 BIT MODE W/ SSEI SET ?
1341 022344 001013          BNE   9$             ;YES !!
1342 022346 013737 001334 001420          MOV   LSTRK,RMDAO          ;SET LAST TRACK AND
1343 022354 112737 000037 001420          MOVB  #31,RMDAO          ;LAST SECTOR
1344 022362 052737 001000 001444          BIS   #SSEI,RMOFO          ;SET SSEI BIT IN OFFSET AND
1345 022370 000137 022072          JMP   1$             ;TEST AGAIN.
  
```

1346 022374
 1347
 1348

9\$:

```

:*****
:*TEST 20      READ HEADER & DATA W/ AOE ERROR
:*****
TST20:

```

022374 000004
 022376 000240
 022400 012706 001100
 022404 013700 001276
 022410 013701 001466
 022414 012737 000020 001226

```

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

```

1349
 1350
 1351 022422 013737 001334 001420
 1352 022430 112737 000036 001420
 1353 022436 012737 010000 001444
 1354 022444
 1355 022444 012737 001060 001446
 1356 022452 012737 176774 001414
 1357 022460 012737 102722 001416
 1358 022466 012737 000073 001412
 1359

```

;SETUP PARAMETERS FOR READING 2 SECTORS STARTING WITH LAST SECTOR
MOV LSTRK,RMDAO ;SET LAST TRACK AND
MOVB #30,RMDAO  ;LAST SECTOR
MOV #FMT16,RMOFO ;16 BIT FORMAT

```

1\$:

```

MOV #560,RMDCO  ;LAST CYLINDER
MOV #-258,*2,RMWCO ;READ 2 SECTORS
MOV #BUFTWO,RMBAO ;DATA BUFFER ADDRESS
MOV #RH!GO,RMCS10 ;READ HEADER AND DATA

```

1360
 1361 022474 004737 031456
 022500 154130

;PREPARE DEVICE FOR DATA TRANSFER

```

JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
.WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 2$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 8$ IF ERROR

```

022502 000404
 022504 000240
 022506 104000
 022510 000137 022746

```

BR 2$
NOP
EMT
JMP 8$

```

1362 022514
 1363 022514 012702 001555
 1364 022520 112722 000006
 1365 022524 112722 000034
 1366 022530 112722 000032
 1367 022534 112722 000002
 1368 022540 112722 000004
 1369 022544 112722 000000
 1370 022550 112722 000200

2\$:

```

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)+

```

1371
 1372 022554 004737 036004
 022560 000404
 022562 000240
 022564 104000
 022566 000137 022746

```

JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 3$ ;GO TO 3$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 8$ ;GO TO 8$ IF ERROR

```

1373 022572
 1374
 1375
 1376 022572 004737 035450
 1377 022576 004737 036356

3\$:

;SETUP INPUT REGISTER BUFFER FOR READING STATUS

```

JSR PC,GETSTS
JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE

```

```

1378
1379 022602 004737 035534      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      022606 000404          BR     4$          ;GO TO 4$ IF NO ERROR
      022610 000240          NOP                    ;RETURN HERE IF ERROR
      022612 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      022614 000137 022746      JMP    8$          ;GO TO 8$ IF ERROR
  
```

```

1380
1381      ;CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
1382      ;DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
  
```

```

1383 022620      4$:
1384 022620 032737 000040 001400  BIT    #SSE,RMER2I      ;IS 'SSE' BIT SET ?
1385 022626 001030          BNE    7$          ;BR IF YES
  
```

```

1386
1387      ;VERIFY THE RESULTS OF READ OPERATION
1388 022630      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      022634 000405          BR     5$          ;GO TO 5$ IF NO ERROR
      022636 000240          NOP                    ;RETURN HERE IF ERROR
      022640 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      022642 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      022644 000137 022746      JMP    8$          ;GO TO 8$ IF ERROR
  
```

```

1389 022650      5$:
1390 022650 004737 051612      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      022654 000405          BR     6$          ;GO TO 6$ IF NO ERROR
      022656 000240          NOP                    ;RETURN HERE IF ERROR
      022660 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      022662 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      022664 000137 022746      JMP    8$          ;GO TO 8$ IF ERROR
  
```

```

1391 022670      6$:
1392 022670 004737 037404      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      022674 000405          BR     7$          ;GO TO 7$ IF NO ERROR
      022676 000240          NOP                    ;RETURN HERE IF ERROR
      022700 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      022702 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      022704 000137 022746      JMP    8$          ;GO TO 8$ IF ERROR
  
```

```

1393 022710      7$:
1394 022710 032737 001000 001444  BIT    #SSEI,RMOFO      ;TEST 16 BIT MODE W/ SSEI SET ?
1395 022716 001013          BNE    8$          ;YES !!
1396 022720 013737 001334 001420  MOV    LSTRK,RMDAO      ;SET LAST TRACK AND
1397 022726 112737 000037 001420  MOVB  #31,,RMDAO        ;LAST SECTOR
1398 022734 052737 001000 001444  BIS    #SSEI,RMOFO      ;SET SSEI BIT IN OFFSET AND
1399 022742 000137 022444          JMP    1$          ;TEST AGAIN.
  
```

```

1400 022746      8$:
1401
1402
  
```

```

*****
;*TEST 21      READ INVALID SECTOR ADDRESS
*****
TST21:
  
```

```

      022746      SCOPE
      022746 000004          NOP                    ;SCOPE CALL
      022750 000240          MOV     #STACK,SP      ;START OF TEST
      022752 012706 001100      MOV     $BASE,R0        ;INITIALIZE STACK POINTER
      022756 013700 001276      MOV     TSTQUE,R1      ;R0 = UNIBUS ADDRESS
      022762 013701 001466      MOV     #21,$TESTN     ;(R1) = DEVICE BEING TESTED
      022766 012737 000021 001226  MOV     #559,,RMDCO     ;SET TEST NUMBER IN APT MAIL BOX
1403
1404 022774 012737 001057 001446      MOV     #TA2!30,,RMDAO ;CYLINDER = 559.
1405 023002 012737 001036 001420      MOV     #0,RMOFO       ;TRACK = 2, INVALID SECTOR = 30.
1406 023010 012737 000000 001444      MOV     #0,RMOFO       ;18 BIT FORMAT
  
```

```

1407 023016 012737 177376 001414      MOV    #-258.,RMWCO    ;2 + 256 WORDS (2'S COMP)
1408 023024 012737 101716 001416      MOV    #BUFONE,RMBAO  ;DATA BUFFER ADDRESS
1409 023032 012737 000073 001412      MOV    #RH!GO,RMCS10 ;READ HEADER AND DATA
1410 023040
1411
1412
1413 023040 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
      JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      .WORD 154130        ;TASK DESCRIPTOR AS FOLLOWS:
                          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                          ;CLEAR CONTROLLER & SELECT DEVICE
                          ;VERIFY CONTROLLER CLEAR OPERATION
                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                          ;VERIFY PACK ACKNOWLEDGE
                          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                          ;VERIFY RECALIBRATION
                          ;GO TO 2$ IF NO ERROR
                          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                          ;GO TO 9$ IF ERROR
      BR    2$
      NOP
      EMT
      JMP    9$
1414 023046 000404
      023050 000240
      023052 104000
      023054 000137 023310
1415
1416
1417 023060 012702 001555      ;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
      MOV    #PUTINX,R2    ;LOAD REGISTER INDEX TABLE
1418 023064 112722 000006      MOV    #RMDA,(R2)+
1419 023070 112722 000034      MOV    #RMDC,(R2)+
1420 023074 112722 000032      MOV    #RMOF,(R2)+
1421 023100 112722 000002      MOV    #RMWC,(R2)+
1422 023104 112722 000004      MOV    #RMBA,(R2)+
1423 023110 112722 000000      MOV    #RMCS1,(R2)+
1424 023114 112722 000200      MOV    #200,(R2)+    ;SET TERMINATOR BYTE
1425
1426 023120 004737 036004      JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR    3$          ;GO TO 3$ IF NO ERROR
      NOP              ;RETURN HERE IF ERROR
      EMT              ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP    9$          ;GO TO 9$ IF ERROR
1427 023136
1428
1429
1430 023136 004737 035450      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
      JSR    PC,GETSTS
1431 023142 004737 036356      JSR    PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1432
1433 023146 004737 035534      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR    4$          ;GO TO 4$ IF NO ERROR
      NOP              ;RETURN HERE IF ERROR
      EMT              ;ERROR # DEFINED BY GET SUBROUTINE
      JMP    9$          ;GO TO 9$ IF ERROR
1434 023164
1435
1436
1437 023164 004737 036552      ;VERIFY RESULTS OF READ COMMAND
      JSR    PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      BR    5$          ;GO TO 5$ IF NO ERROR
      NOP              ;RETURN HERE IF ERROR
      EMT              ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      JMP    9$          ;GO TO 9$ IF ERROR
1438 023204
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

```
1439 023204 004737 051612      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      023210 000405      BR     6$             ;GO TO 6$ IF NO ERROR
      023212 000240      NOP                    ;RETURN HERE IF ERROR
      023214 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      023216 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      023220 000137 023310      JMP    9$             ;GO TO 9$ IF ERROR
1440 023224                      6$:
1441 023224 004737 037404      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      023230 000405      BR     7$             ;GO TO 7$ IF NO ERROR
      023232 000240      NOP                    ;RETURN HERE IF ERROR
      023234 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      023236 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      023240 000137 023310      JMP    9$             ;GO TO 9$ IF ERROR
1442 023244                      7$:
1443
1444 ;INCREMENT ADDRESS AND FORMAT NEXT SECTOR
1445 023244 005237 001420      INC    RMDAO          ;ADVANCE SECTOR COUNT
1446 023250 023727 001420 000100  CMP    RMDAO,#64.     ;DONE YET ?
1447 023256 101412      BLOS   8$             ;NO - TEST NEXT SECTOR
1448
1449 023260 032737 010000 001444  BIT    #FMT16,RMOFO   ;TEST 16 BIT MODE YET ?
1450 023266 001010      BNE    9$             ;YES !!
1451 023270 012737 001037 001420  MOV    #TA2!31.,RMDAO ;TRACK = 2, INVALID SECTOR = 31.
1452 023276 012737 010000 001444  MOV    #FMT16,RMOFO   ;SET 16 BIT MODE IN OFFSET AND
1453 023304 000137 023040      JMP    1$             ;TEST AGAIN.
1454
1455 023310                      9$:
1456
1457
```

```
::*****
:*TEST 22 READ INVALID TRACK ADDRESS
:*****
TST22:
```

```
023310
023310 000004      SCOPE          ;SCOPE CALL
023312 000240      NOP            ;START OF TEST
023314 012706 001100  MOV    #STACK,SP      ;INITIALIZE STACK POINTER
023320 013700 001276  MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
023324 013701 001466  MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
023330 012737 000022 001226  MOV    #22,$TESTN    ;:SET TEST NUMBER IN APT MAIL BOX
1458
1459 023336 012737 001057 001446  MOV    #559.,RMDCO    ;CYLINDER = 559.
1460 023344 013737 001334 001420  MOV    LSTRK,RMDAO    ;LOAD LAST TRACK, SECTOR = 0
1461 023352 105237 001421      INCB   RMDAO+1       ;INCREMENT TO FIRST INVALID TRACK
1462 023356 012737 010000 001444  MOV    #FMT16,RMOFO   ;16 BIT FORMAT
1463 023364 012737 177376 001414  MOV    #-258.,RMWCO   ;2 + 256 WORDS (2'S COMP)
1464 023372 012737 101716 001416  MOV    #BUFONE,RMBAO  ;DATA BUFFER ADDRESS
1465 023400 012737 000073 001412  MOV    #RH!GO,RMCS10 ;READ HEADER AND DATA
1466 023406
1467
1468 ;PREPARE DEVICE FOR DATA TRANSFER
1469 023406 004737 031456      JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      023412 154130      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' CR 'PIP' IS SET
```

```

023414 000404 BR 2$ ;VERIFY RECALIBRATION
023416 000240 NOP ;GO TO 2$ IF NO ERROR
023420 104000 EMT ;RETURN HERE IF ERROR
023422 000137 023632 JMP 8$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 8$ IF ERROR
1470 023426 2$:
1471
1472 ;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
1473 023426 012702 001555 MOV #PUTINX,R2 ;READ REGISTER INDEX TABLE
1474 023432 112722 000006 MOVB #RMDA,(R2)+
1475 023436 112722 000034 MOVB #RMDC,(R2)+
1476 023442 112722 000032 MOVB #RMOF,(R2)+
1477 023446 112722 000002 MOVB #RMWC,(R2)+
1478 023452 112722 000004 MOVB #RMBA,(R2)+
1479 023456 112722 000000 MOVB #RMCS1,(R2)+
1480 023462 112722 000200 MOVB #200,(R2)+ ;SET TERMINATOR BYTE
1481
1482 023466 004737 036004 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
023472 000404 BR 3$ ;GO TO 3$ IF NO ERROR
023474 000240 NOP ;RETURN HERE IF ERROR
023476 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
023500 000137 023632 JMP 8$ ;GO TO 8$ IF ERROR
1483 023504 3$:
1484
1485 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1486 023504 004737 035450 JSR PC,GETSTS
1487 023510 004737 036356 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
1488
1489 023514 004737 035534 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
023520 000404 BR 4$ ;GO TO 4$ IF NO ERROR
023522 000240 NOP ;RETURN HERE IF ERROR
023524 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
023526 000137 023632 JMP 8$ ;GO TO 8$ IF ERROR
1490 023532 4$:
1491
1492 ;VERIFY RESULTS OF READ COMMAND
1493 023532 004737 036552 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
023536 000405 BR 5$ ;GO TO 5$ IF NO ERROR
023540 000240 NOP ;RETURN HERE IF ERROR
023542 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
023544 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
023546 000137 023632 JMP 8$ ;GO TO 8$ IF ERROR
1494 023552 5$:
1495 023552 004737 051612 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
023556 000405 BR 6$ ;GO TO 6$ IF NO ERROR
023560 000240 NOP ;RETURN HERE IF ERROR
023562 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
023564 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
023566 000137 023632 JMP 8$ ;GO TO 8$ IF ERROR
1496 023572 6$:
1497 023572 004737 037404 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
023576 000405 BR 7$ ;GO TO 7$ IF NO ERROR
023600 000240 NOP ;RETURN HERE IF ERROR
023602 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
023604 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
023606 000137 023632 JMP 8$ ;GO TO 8$ IF ERROR
1498 023612 7$:

```


CZRNEAO RM80 FCTNL PT2 MACRO V04.00 28-JAN-82 16:38:23 PAGE 13-48
T22 READ INVALID TRACK ADDRESS

1499
1500
1501 023612 105237 001421
1502 023616 123727 001421 000200
1503 023624 101002
1504 023626 000137 023406
1505 023632
1506
1507

: INCREMENT ADDRESS AND FORMAT NEXT TRACK
INCB RMDAO+1 : ADVANCE TRACK COUNT
CMPB RMDAO+1,#128. : DONE ALL TRACKS??
BHI 8\$: YES!!
JMP 1\$: GO DO NEXT TRACK

8\$:
:*****
: *TEST 23 READ INVALID CYLINDER ADDRESS
:*****
TST23:

023632
023632 000004
023634 000240
023636 012706 001100
023642 013700 00127C
023646 013701 001466
023652 012737 000023 001226
1508
1509 023660 012737 001061 001446
1510 023666 012737 000000 001420
1511 023674 012737 010000 001444
1512 023702 012737 177376 001414
1513 023710 012737 101716 001416
1514 023716 012737 000073 001412
1515 023724
1516
1517

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 #23,\$TESTN : SET TEST NUMBER IN APT MAIL BOX
MOV #561.,RMDCO : START AT FIRST INVALID CYLINDER
MOV #0,RMDAO : TRACK = 0, SECTOR = 0
MOV #FMT16,RMOFO : 16 BIT FORMAT
MOV #-258.,RMWCO : 2 + 256 WORDS (2'S COMP)
MOV #BUFONE,RMBAO : DATA BUFFER ADDRESS
MOV #RH!GO,RMCS10 : READ HEADER AND DATA

1518 023724 004737 031456
023730 154130

1\$:
: PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP : PREPARE DEVICE FOR TEST
.WORD 154130 : TASK DESCRIPTOR AS FOLLOWS:
: SELECT DEVICE & VERIFY DEVICE AVAILABLE
: CLEAR CONTROLLER & SELECT DEVICE
: VERIFY CONTROLLER CLEAR OPERATION
: PACK ACKNOWLEDGE IF VOLUME NOT VALID
: VERIFY PACK ACKNOWLEDGE
: RECALIBRATE IF "SKI" OR "PIP" IS SET
: VERIFY RECALIBRATION
: GO TO 2\$ IF NO ERROR
: RETURN HERE IF ERROR
: ERROR # DEFINED BY TSTPRP SUBROUTINE
: GO TO 8\$ IF ERROR
BR 2\$
NOP
EMT
JMP 8\$

023732 000404
023734 000240
023736 104000
023740 000137 024150
1519 023744

2\$:
: SETUP AND EXECUTE READ HEADER AND DATA COMMAND
MOV #PUTINX,R2 : LOAD REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+ : SET TERMINATOR BYTE

1520
1521
1522 023744 012702 001555
1523 023750 112722 000006
1524 023754 112722 000034
1525 023760 112722 000032
1526 023764 112722 000002
1527 023770 112722 000004
1528 023774 112722 000000
1529 024000 112722 000200
1530
1531 024004 004737 036004
024010 000404
024012 000240
024014 104000

JSR PC,PUT : GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 3\$: GO TO 3\$ IF NO ERROR
NOP : RETURN HERE IF ERROR
EMT : ERROR # DEFINED BY PUT SUBROUTINE

```

1532 024016 000137 024150          JMP      8$          ;GO TO 8$ IF ERROR
1533 024022
1534
1535 024022 004737 035450          ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1536 024026 004737 036356          JSR      PC,GETSTS
1537          JSR      PC,TIMOUT          ;WAIT FOR COMMAND TO COMPLETE
1538 024032 004737 035534          JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
024036 000404          BR       4$          ;GO TO 4$ IF NO ERROR
024040 000240          NOP
024042 104000          EMT
024044 000137 024150          JMP      8$          ;ERROR # DEFINED BY GET SUBROUTINE
1539 024050          ;GO TO 8$ IF ERROR
1540
1541
1542 024050 004737 036552          ;VERIFY RESULTS OF READ COMMAND
024054 000405          JSR      PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
024056 000240          BR       5$          ;GO TO 5$ IF NO ERROR
024060 104000          NOP          ;RETURN HERE IF ERROR
024062 004736          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
024064 000137 024150          JSR      PC,a(SP)+          ;GO BACK FOR MORE ERROR CHECKS
1543 024070          JMP      8$          ;GO TO 8$ IF ERROR
1544 024070 004737 051612          5$: JSR      PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
024074 000405          BR       6$          ;GO TO 6$ IF NO ERROR
024076 000240          NOP          ;RETURN HERE IF ERROR
024100 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
024102 004736          JSR      PC,a(SP)+          ;GO BACK FOR MORE ERROR CHECKS
024104 000137 024150          JMP      8$          ;GO TO 8$ IF ERROR
1545 024110          6$: JSR      PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
1546 024110 004737 037404          BR       7$          ;GO TO 7$ IF NO ERROR
024114 000405          NOP          ;RETURN HERE IF ERROR
024116 000240          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
024120 104000          JSR      PC,a(SP)+          ;GO BACK FOR MORE ERROR CHECKS
024122 004736          JMP      8$          ;GO TO 8$ IF ERROR
024124 000137 024150
1547 024130          7$:
1548
1549          ;INCREMENT ADDRESS AND FORMAT NEXT CYLINDER
1550 024130 005237 001446          INC      RMDCO          ;ADVANCE CYLINDER COUNT
1551 024134 023727 001446 002000          CMP      RMDCO,#1024.          ;DONE ALL CYLINDERS??
1552 024142 103002          BHIS    8$          ;YES!!
1553 024144 000137 023724          JMP      1$          ;GO DO NEXT SECTOR
1554 024150          8$:
1555
1556          ;*****
          ;*TEST 24          FORMAT AT OFFSET
          ;*****
          TST24:
024150          SCOPE          ;SCOPE CALL
024150 000004          NOP          ;START OF TEST
024152 000240          MOV      #STACK,SP          ;INITIALIZE STACK POINTER
024154 012706 001100          MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
024160 013700 001276          MOV      TSTQUE,R1          ;(R1) = DEVICE BEING TESTED
024164 013701 001466          MOV
024170 012737 000024 001226          MOV      #24,$TESTN          ;SET TEST NUMBER IN APT MAIL BOX
1557
1558          ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1559 024176 012737 001057 001446          MOV      #559.,RMDCO          ;CYLINDER = 559.
  
```

```

1560 024204 012737 001000 001420      MOV      #TA2,RMDAO      ;TRACK = 2, SECTOR = 0
1561 024212 012737 010000 001444      MOV      #FMT16,RMOFO   ;16 BIT FORMAT
1562 024220 012737 177376 001414      MOV      #-258.,RMWCO   ;2 + 256 WORDS (2'S COMP)
1563 024226 012737 101716 001416      MOV      #BUFONE,RMBAO  ;DATA BUFFER ADDRESS
1564 024234 012737 000063 001412      MOV      #WH!GO,RMCS10  ;WRITE HEADER AND DATA
1565
1566      ;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT   ;CALL BAD SECTOR MODULE
      BR      1$          ;GO TO 1$ IF NO ERROR
      TYPE    .SCTMSG     ;TYPE BAD SECTOR MESSAGE
      EMT
      JMP     17$        ;ERROR # DEFINED BY BADSCT SUBROUTINE
                        ;GO TO 17$ IF ERROR
1$:
1567 024262      MOV      #RMDAO,$TMP0   ;USE SECTOR FOR DATA PATTERN
1568 024262 012737 001420 001174      MOV      #1,$TMP1
1569 024270 012737 000001 001176      JSR      PC,GENBUF     ;GO GENERATE DATA BUFFER
1570 024276 004737 034640
1571 024302
1572
1573      ;PREPARE DEVICE FOR DATA TRANSFER
1574 024302 004737 031456      JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                        ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                        ;CLEAR CONTROLLER & SELECT DEVICE
                        ;VERIFY CONTROLLER CLEAR OPERATION
                        ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                        ;VERIFY PACK ACKNOWLEDGE
                        ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                        ;VERIFY RECALIBRATION
                        ;GO TO 3$ IF NO ERROR
                        ;RETURN HERE IF ERROR
                        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                        ;GO TO 17$ IF ERROR
      BR      3$
      NOP
      EMT
      JMP     17$
3$:
      024310 000404      BR      3$
      024312 000240      NOP
      024314 104000      EMT
      024316 000137 025062      JMP     17$
1575 024322      3$:
1576
1577      ;OFFSET DEVICE FOR WRITE
1578 024322 012737 000015 001412      MOV      #OFFSET!GO,RMCS10 ;CHANGE TO OFFSET COMMAND
1579 024330 012702 001555      MOV      #PUTINX,R2       ;WRITE PUT INDEX TABLE
1580 024334 112722 000006      MOV      #RMDA,(R2)+
1581 024340 112722 000034      MOV      #RMDC,(R2)+
1582 024344 112722 000032      MOV      #RMOF,(R2)+
1583 024350 112722 000000      MOV      #RMCS1,(R2)+
1584 024354 112712 000200      MOV      #200,(R2)       ;TERMINATE TABLE
1585
1586 024360 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      024364 000404      BR      4$             ;GO TO 4$ IF NO ERROR
      024366 000240      NOP                   ;RETURN HERE IF ERROR
      024370 104000      EMT                   ;ERROR # DEFINED BY PUT SUBROUTINE
      024372 000137 025062      JMP     17$           ;GO TO 17$ IF ERROR
1587 024376      4$:
1588
1589      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND AT OFFSET
1590 024376 012737 000063 001412      MOV      #WH!GO,RMCS10   ;WRITE HEADER AND DATA COMMAND
1591 024404 012702 001555      MOV      #PUTINX,R2     ;WRITE REGISTER INDEX TABLE
1592 024410 112722 000002      MOV      #RMWC,(R2)+
1593 024414 112722 000004      MOV      #RMBA,(R2)+
1594 024420 112722 000000      MOV      #RMCS1,(R2)+
1595 024424 112722 000200      MOV      #200,(R2)+
    
```

```

1596
1597 024430 004737 036004      JSR    PC,PUT      :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      024434 000404      BR     5$          :GO TO 5$ IF NO ERROR
      024436 000240      NOP                    :RETURN HERE IF ERROR
      024440 104000      EMT                    :ERROR # DEFINED BY PUT SUBROUTINE
      024442 000137 025062      JMP    17$         :GO TO 17$ IF ERROR
1598 024446      5$:
1599
1600      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1601 024446 004737 035450      JSR    PC,GETSTS
1602 024452 004737 036356      JSR    PC,TIMOUT      :WAIT FOR COMMAND TO COMPLETE
1603
1604 024456 004737 035534      JSR    PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
      024462 000404      BR     6$          :GO TO 6$ IF NO ERROR
      024464 000240      NOP                    :RETURN HERE IF ERROR
      024466 104000      EMT                    :ERROR # DEFINED BY GET SUBROUTINE
      024470 000137 025062      JMP    17$         :GO TO 17$ IF ERROR
1605 024474      6$:
1606
1607      ;VERIFY RESULTS OF WRITE COMMAND
1608 024474 004737 036552      JSR    PC,PRIERR    :GO CHECK FOR PRIMARY ERRORS
      024500 000405      BR     7$          :GO TO 7$ IF NO ERROR
      024502 000240      NOP                    :RETURN HERE IF ERROR
      024504 104000      EMT                    :ERROR # DEFINED BY PRIERR SUBROUTINE
      024506 004736      JSR    PC,a(SP)+    :GO BACK FOR MORE ERROR CHECKS
      024510 000137 025062      JMP    17$         :GO TO 17$ IF ERROR
1609 024514      7$:
1610 024514 004737 051612      JSR    PC,DTASTS    :GO VERIFY RESULTS OF DATA TRANSFER
      024520 000405      BR     8$          :GO TO 8$ IF NO ERROR
      024522 000240      NOP                    :RETURN HERE IF ERROR
      024524 104000      EMT                    :ERROR # DEFINED BY DTASTS SUBROUTINE
      024526 004736      JSR    PC,a(SP)+    :GO BACK FOR MORE ERROR CHECKS
      024530 000137 025062      JMP    17$         :GO TO 17$ IF ERROR
1611 024534      8$:
1612 024534 004737 037404      JSR    PC,SECERR    :GO CHECK FOR SECONDARY ERRORS
      024540 000405      BR     9$          :GO TO 9$ IF NO ERROR
      024542 000240      NOP                    :RETURN HERE IF ERROR
      024544 104000      EMT                    :ERROR # DEFINED BY SECERR SUBROUTINE
      024546 004736      JSR    PC,a(SP)+    :GO BACK FOR MORE ERROR CHECKS
      024550 000137 025062      JMP    17$         :GO TO 17$ IF ERROR
1613 024554      9$:
1614      ;OFFSET DEVICE FOR READ
1615 024554 012737 000015 001412  MOV    #OFFSET!GO, RMCS10 :CHANGE TO OFFSET COMMAND
1616 024562 012702 001555      MOV    #PUTINX,R2     :WRITE PUT INDEX TABLE
1617 024566 112722 000006      MOVB   #RMDA,(R2)+
1618 024572 112722 000034      MOVB   #RMDC,(R2)+
1619 024576 112722 000032      MOVB   #RMOF,(R2)+
1620 024602 112722 000000      MOVB   #RMCS1,(R2)+
1621 024606 112722 000200      MOVB   #200,(R2)+
1622
1623 024612 004737 036004      JSR    PC,PUT      :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      024616 000404      BR     10$         :GO TO 10$ IF NO ERROR
      024620 000240      NOP                    :RETURN HERE IF ERROR
      024622 104000      EMT                    :ERROR # DEFINED BY PUT SUBROUTINE
      024624 000137 025062      JMP    17$         :GO TO 17$ IF ERROR
1624 024630      10$:
1625

```

```

1626                                     ;READ HEADER AND DATA AT OFFSET
1627 024630 012737 000073 001412      MOV      #RH!GO,RMCS10      ;READ HEADER & DATA COMMAND
1628 024636 012737 102722 001416      MOV      #BUFTWO,RMBA0     ;CHANGE BUS ADDRESS
1629 024644 012702 001555                MOV      #PUTINX,R2        ;WRITE PUT INDEX TABLE
1630 024650 112722 000002                MOVB     #RMWC,(R2)+
1631 024654 112722 000004                MOVB     #RMBA,(R2)+
1632 024660 112722 000000                MOVB     #RMCS1,(R2)+
1633 024664 112722 000200                MOVB     #200,(R2)+
1634
1635 024670 004737 036004                JSR      PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      11$                    ;GO TO 11$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY PUT SUBROUTINE
                                JMP      17$                    ;GO TO 17$ IF ERROR
1636 024706 000137 025062                11$:
1637
1638                                     ;WAIT FOR READ TO COMPLETE AND GET STATUS
1639 024706 004737 036356                JSR      PC,TIMOUT        ;WAIT FOR READ TO COMPLETE
1640
1641 024712 004737 035534                JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR      12$                    ;GO TO 12$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY GET SUBROUTINE
                                JMP      17$                    ;GO TO 17$ IF ERROR
1642 024730 000137 025062                12$:
1643
1644                                     ;VERIFY THE RESULTS OF READ OPERATION
1645 024730 004737 036552                JSR      PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
                                BR      13$                    ;GO TO 13$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR      PC,@(SP)+            ;GO BACK FOR MORE ERROR CHECKS
                                JMP      17$                    ;GO TO 17$ IF ERROR
1646 024750 000137 025062                13$:
1647 024750 004737 051512                JSR      PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR      14$                    ;GO TO 14$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR      PC,@(SP)+            ;GO BACK FOR MORE ERROR CHECKS
                                JMP      17$                    ;GO TO 17$ IF ERROR
1648 024770 000137 025062                14$:
1649 024770 004737 037404                JSR      PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
                                BR      15$                    ;GO TO 15$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY SECERR SUBROUTINE
                                JSR      PC,@(SP)+            ;GO BACK FOR MORE ERROR CHECKS
                                JMP      17$                    ;GO TO 17$ IF ERROR
1650 025010 000137 025062                15$:
1651
1652                                     ;VERIFY DATA
1653 025010 004737 035106                JSR      PC,CMPBUF        ;GO COMPARE WRITE, READ DATA BUFFERS
                                .WORD    BUFOE                ;STARTING ADDRESS OF WRITE BUFFER
                                .WORD    BUFTWO               ;STARTING ADDRESS OF READ BUFFER
                                BR      16$                    ;GO TO 16$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY CMPBUF SUBROUTINE
                                JMP      17$                    ;GO TO 17$ IF ERROR
025014 101716
025016 102722
025020 000404
025022 000240
025024 104000
025026 000137 025062
    
```

1654 025032
 1655 025032 032737 000200 001444
 1656 025040 001010
 1657 025042 052737 000200 001444
 1658 025050 012737 101716 001416
 1659 025056 000137 024302
 1660 025062
 1661
 1662

16\$: BIT #OFD,RMOFO ;DONE BOTH OFFSETS??
 BNE 17\$;YES!!
 BIS #OFD,RMOFO ;SET FORWARD OFFSET
 MOV #BUFONE,RMBAO ;CHANGE DATA BUFFER
 JMP 2\$

17\$:
 ::*****
 ;*TEST 25 IVC FORMAT TEST
 ::*****
 TST25:

025062
 025062 000004
 025064 000240
 025066 012706 001100
 025072 013700 001276
 025076 013701 001466
 025102 012737 000025 001226

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 #25,\$TSTN ;:SET TEST NUMBER IN APT MAIL BOX

1663
 1664
 1665 025110 012737 001057 001446
 1666 025116 012737 001000 001420
 1667 025124 012737 010000 001444
 1668 025132 012737 177376 001414
 1669 025140 012737 101716 001416
 1670 025146 012737 000062 001412
 1671
 1672

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
 MOV #559,RMDCO ;CYLINDER = 559.
 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
 MOV #FMT16,RMOFO ;16 BIT FORMAT
 MOV #-258,RMWCO ;2 + 256 WORDS (2'S COMP)
 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
 MOV #WH,RMC\$10 ;WRITE HEADER AND DATA

025154 004737 032406
 025160 000405
 025162 104401 063446
 025166 104000
 025170 000137 025704
 1673 025174
 1674 025174 012737 001420 001174
 1675 025202 012737 000001 001176
 1676 025210 004737 034640
 1677 025214
 1678
 1679

;VERIFY THAT SECTOR IS NOT BAD
 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
 BR 1\$;GO TO 1\$ IF NO ERROR
 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
 JMP 16\$;GO TO 16\$ IF ERROR

025214 004737 031456
 025220 154130

1\$: MOV #RMDAO,\$TMP0 ;USE SECTOR FOR DATA PATTERN
 MOV #1,\$TMP1
 JSR PC,GENBUF ;GO GENERATE DATA BUFFER

025222 000404
 025224 000240
 025226 104000
 025230 000137 025704
 1681 025234
 1682
 1683
 1684 025234 012737 000001 001436

;PREPARE DEVICE FOR DATA TRANSFER
 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
 ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
 ;CLEAR CONTROLLER & SELECT DEVICE
 ;VERIFY CONTROLLER CLEAR OPERATION
 ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
 ;VERIFY PACK ACKNOWLEDGE
 ;RECALIBRATE IF "SKI" OR "PIP" IS SET
 ;VERIFY RECALIBRATION
 BR 3\$;GO TO 3\$ IF NO ERROR
 NOP ;RETURN HERE IF ERROR
 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
 JMP 16\$;GO TO 16\$ IF ERROR

3\$: ;RESET VOLUME VALID BY SETTING AND RESETTING DIAGNOSTIC MODE
 MOV #DMD,RMMR10 ;SET DIAGNOSTIC MODE

```

1685 025242 112737 000024 001555      MOVB  #RMMR1,PUTINX      ;WRITE REGISTER INDEX TABLE
1686 025250 112737 000200 001556      MOVB  #200,PUTINX+1
1687
1688 025256 004737 036004      JSR   PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      025262 000404      BR    4$                ;GO TO 4$ IF NO ERROR
      025264 000240      NOP                    ;RETURN HERE IF ERROR
      025266 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      025270 000137 025704      JMP   16$              ;GO TO 16$ IF ERROR
1689 025274      4$:
1690
1691      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1692 025274 012737 000063 001412      MOV   #WH!GO,RMCS10    ;WRITE HEADER AND DATA
1693 025302 012737 000000 001436      MOV   #0,RMMR10        ;RESET DIAGNOSTIC MODE
1694 025310 012702 001555      MOV   #PUTINX,R2       ;WRITE REGISTER INDEX TABLE
1695 025314 112722 000024      MOVB  #RMMR1,(R2)+
1696 025320 112722 000006      MOVB  #RMDA,(R2)+
1697 025324 112722 000034      MOVB  #RMDC,(R2)+
1698 025330 112722 000032      MOVB  #RMOF,(R2)+
1699 025334 112722 000002      MOVB  #RMWC,(R2)+
1700 025340 112722 000004      MOVB  #RMBA,(R2)+
1701 025344 112722 000000      MOVB  #RMCS1,(R2)+
1702 025350 112722 000200      MOVB  #200,(R2)+
1703
1704 025354 004737 036004      JSR   PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      025360 000404      BR    5$                ;GO TO 5$ IF NO ERROR
      025362 000240      NOP                    ;RETURN HERE IF ERROR
      025364 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      025366 000137 025704      JMP   16$              ;GO TO 16$ IF ERROR
1705 025372      5$:
1706
1707      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1708 025372 004737 035450      JSR   PC,GETSTS
1709 025376 004737 036356      JSR   PC,TIMOUT        ;WAIT FOR COMMAND TO COMPLETE
1710
1711 025402 004737 035534      JSR   PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      025406 000404      BR    6$                ;GO TO 6$ IF NO ERROR
      025410 000240      NOP                    ;RETURN HERE IF ERROR
      025412 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      025414 000137 025704      JMP   16$              ;GO TO 16$ IF ERROR
1712 025420      6$:
1713
1714      ;VERIFY RESULTS OF WRITE COMMAND
1715 025420 004737 036552      JSR   PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      025424 000405      BR    7$                ;GO TO 7$ IF NO ERROR
      025426 000240      NOP                    ;RETURN HERE IF ERROR
      025430 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      025432 004736      JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      025434 000137 025704      JMP   16$              ;GO TO 16$ IF ERROR
1716 025440      7$:
1717 025440 032737 010000 001400      BIT   #IVC,RMER2I      ;IS IVC STATUS SET??
1718 025446 001012      BNE   8$                ;YES!!
1719 025450 013737 001400 001142      MOV   RMER2I,$BDDAT    ;RECEIVED STATUS FOR TYPEOUT
1720 025456 013737 001400 001140      MOV   RMER2I,$GDDAT    ;EXPECTED STATUS
1721 025464 052737 010000 001140      BIS   #IVC,$GDDAT
1722 025472 104342      EMT
1723 025474      8$:
1724 025474 004737 037404      JSR   PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
  
```

```

T25 IVC FORMAT TEST

025500 000405 BR 9$ :GO TO 9$ IF NO ERROR
025502 000240 NOP :RETURN HERE IF ERROR
025504 104000 EMT :ERROR # DEFINED BY SECERR SUBROUTINE
025506 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
025510 000137 JMP 16$ :GO TO 16$ IF ERROR
1725 025514 025704 9$:
1726
1727 :CLEAR IVC ERROR
1728 025514 004737 031456 JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
025520 040100 .WORD 040100 :TASK DESCRIPTOR AS FOLLOWS:
:CLEAR CONTROLLER & SELECT DEVICE
:VERIFY CONTROLLER CLEAR OPERATION
025522 000404 BR 10$ :GO TO 10$ IF NO ERROR
025524 000240 NOP :RETURN HERE IF ERROR
025526 104000 EMT :ERROR # DEFINED BY TSTPRP SUBROUTINE
025530 000137 JMP 16$ :GO TO 16$ IF ERROR
1729 025534 10$:
1730
1731 :READ HEADER AND DATA FOR SECTOR JUST WRITTEN
1732 025534 012737 000073 001412 MOV #RH!GO,RMCS10 :READ HEADER & DATA COMMAND
1733 025542 012737 102722 001416 MOV #BUFTWO,RMBAO :CHANGE BUS ADDRESS
1734
1735 025550 004737 036004 JSR PC,PUT :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
025554 000404 BR 11$ :GO TO 11$ IF NO ERROR
025556 000240 NOP :RETURN HERE IF ERROR
025560 104000 EMT :ERROR # DEFINED BY PUT SUBROUTINE
025562 000137 JMP 16$ :GO TO 16$ IF ERROR
1736 025566 11$:
1737
1738 :SETUP INPUT REGISTER BUFFER FOR READING STATUS
1739 025566 004737 035450 JSR PC,GETSTS
1740 025572 004737 036356 JSR PC,TIMOUT :WAIT FOR READ TO COMPLETE
1741
1742 025576 004737 035534 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
025602 000404 BR 12$ :GO TO 12$ IF NO ERROR
025604 000240 NOP :RETURN HERE IF ERROR
025606 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
025610 000137 JMP 16$ :GO TO 16$ IF ERROR
1743 025614 12$:
1744
1745 :VERIFY THE RESULTS OF READ OPERATION
1746 025614 004737 036552 JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
025620 000405 BR 13$ :GO TO 13$ IF NO ERROR
025622 000240 NOP :RETURN HERE IF ERROR
025624 104000 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
025626 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
025630 000137 JMP 16$ :GO TO 16$ IF ERROR
1747 025634 13$:
1748 025634 032737 010000 001400 BIT #IVC,RMER2I :IS IVC STATUS SET??
1749 025642 001012 BNE 14$ :YES!!
1750 025644 013737 001400 001142 MOV RMER2I,$BDDAT :RECEIVED STATUS FOR TYPEOUT
1751 025652 013737 001400 001140 MOV RMER2I,$GDDAT :EXPECTED STATUS
1752 025660 052737 010000 001140 BIS #IVC,$GDDAT
1753 025666 104342 EMT 342
1754 025670 14$:
1755 025670 004737 037404 JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
025674 000403 BR 15$ :GO TO 15$ IF NO ERROR

```



```

025676 000240      NOP      ;RETURN HERE IF ERROR
025700 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
025702 004736      JSR      PC,a(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1756 025704      15$:
1757
1758 025704      16$:
1759
1760
::*****
:*TEST 26      FORMAT ERROR TEST
::*****
TST26:
025704          SCOPE      ;SCOPE CALL
025704 000004      NOP      ;START OF TEST
025706 000240      MOV      #STACK,SP ;INITIALIZE STACK POINTER
025710 012706 001100 MOV      $BASE,R0    ;R0 = UNIBUS ADDRESS
025714 013700 001276 MOV      TSTQUE,R1   ;(R1) = DEVICE BEING TESTED
025720 013701 001466 MOV      #26,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
025724 012737 000026 001226

1761
1762      ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1763 025732 012737 001057 001446 MOV      #559.,RMDCO ;CYLINDER = 559.
1764 025740 012737 001000 001420 MOV      #TA2,RMDAO  ;TRACK = 2, SECTOR = 0
1765 025746 012737 010000 001444 MOV      #FMT16,RMOFO ;16 BIT FORMAT
1766 025754 012737 177376 001414 MOV      #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
1767 025762
1768 025762 012737 101716 001416 1$:      MOV      #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1769 025770 012737 000062 001412      MOV      #WH,RMCS10 ;WRITE HEADER AND DATA
1770
1771      ;VERIFY THAT SECTOR IS NOT BAD
025776 004737 032406      JSR      PC,BADSCT ;CALL BAD SECTOR MODULE
026002 000405      BR      2$      ;GO TO 2$ IF NO ERROR
026004 104401 063446      TYPE      ,SCTMSG ;TYPE BAD SECTOR MESSAGE
026010 104000      EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
026012 000137 026606      JMP      22$     ;GO TO 22$ IF ERROR
1772 026016
1773 026016 012737 065116 001174 2$:      MOV      #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
1774 026024 012737 000001 001176      MOV      #1,$TMP1
1775 026032 004737 034640      JSR      PC,GENBUF ;GO GENERATE DATA BUFFER
1776 026036
1777
1778      ;PREPARE DEVICE FOR DATA TRANSFER
1779 026036 004737 031456      JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
026042 154130      .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 4$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 22$ IF ERROR
026044 000404      BR      4$
026046 000240      NOP
026050 104000      EMT
026052 000137 026606      JMP      22$
1780 026056      4$:
1781
1782      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1783 026056 012737 000062 001412      MOV      #WH,RMCS10 ;WRITE HEADER AND DATA

```

```

1784 026064 012702 001555      MOV      #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
1785 026070 112722 000006      MOVB     #RMDA,(R2)+
1786 026074 112722 000034      MOVB     #RMDC,(R2)+
1787 026100 112722 000032      MOVB     #RMOF,(R2)+
1788 026104 112722 000002      MOVB     #RMWC,(R2)+
1789 026110 112722 000004      MOVB     #RMB A,(R2)+
1790 026114 112722 000000      MOVB     #RMCS1,(R2)+
1791 026120 112722 000200      MOVB     #200,(R2)+
1792
1793 026124 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      5$          ;GO TO 5$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
                                JMP      22$      ;GO TO 22$ IF ERROR
1794 026142 5$:
1795 026142 012702 000002      MOV      #2,R2          ;DO 2 CLOCK CYCLES
1796 026146 004737 036520      JSR      PC,STIMER      ;START CLOCK TIMER
1797 026152 032777 000200 153342 7$:      BIT      #BIT7,@CLKADR ;TIMER DONE??
1798 026160 001774          BEQ      7$            ;NO!!
1799 026162 005302          DEC      R2            ;DEC NUMBER OF CYCLES
1800 026164 001370          BNE     6$            ;CONTINUE IF NOT DONE
1801
1802 026166 012737 000063 001412      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
1803
1804 026174 004737 036004      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      8$          ;GO TO 8$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
                                JMP      22$      ;GO TO 22$ IF ERROR
1805 026212 8$:
1806
1807      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1808 026212 004737 035450      JSR      PC,GETSTS
1809 026216 004737 036356      JSR      PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1810
1811 026222 004737 035534      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR      9$          ;GO TO 9$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY GET SUBROUTINE
                                JMP      22$      ;GO TO 22$ IF ERROR
1812 026240 9$:
1813
1814      ;VERIFY RESULTS OF WRITE COMMAND
1815 026240 004737 036552      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
                                BR      10$         ;GO TO 10$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
                                JMP      22$      ;GO TO 22$ IF ERROR
1816 026260 10$:
1817 026260 004737 051612      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR      11$         ;GO TO 11$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
                                JMP      22$      ;GO TO 22$ IF ERROR
1818 026300 11$:
    
```

```

1819 026300 004737 037404      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      026304 000405      BR      12$          ;GO TO 12$ IF NO ERROR
      026306 000240      NOP                    ;RETURN HERE IF ERROR
      026310 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      026312 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      026314 000137 026606  JMP      22$          ;GO TO 22$ IF ERROR

1820 026320      12$:
1821
1822      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN IN OPPOSITE FORMAT
1823      ;AND CHECK THAT "FER" SETS
1824
1825 026320 005137 001444      COM      RMOFO          ;CHANGE TO OPPOSITE FORMAT
1826 026324 042737 167777 001444  BIC      #^C<FMT16>,RMOFO
1827 026332 012737 102722 001416  MOV      #BUFTWO,RMBAO  ;CHANGE BUS ADDRESS
1828 026340 012737 000072 001412  MOV      #RH,RMCS10    ;READ HEADER & DATA COMMAND
1829
1830 026346 004737 036004      JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026352 000404      BR      13$          ;GO TO 13$ IF NO ERROR
      026354 000240      NOP                    ;RETURN HERE IF ERROR
      026356 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      026360 000137 026606  JMP      22$          ;GO TO 22$ IF ERROR

1831 026364      13$:
1832 026364 012702 000002      MOV      #2,R2         ;DO 2 CLOCK CYCLES
1833 026370 004737 036520 153120 14$: JSR    PC,STIMER      ;START CLOCK TIMER
1834 026374 032777 000200      BIT      #BIT7,@CLKADR ;TIMER DONE??
1835 026402 001774      BEQ      15$          ;NO!!
1836 026404 005302      DEC      R2           ;DEC NUMBER OF CYCLES
1837 026406 001370      BNE      14$          ;CONTINUE IF NOT DONE
1838
1839 026410 012737 000073 001412  MOV      #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
1840
1841 026416 004737 036004      JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026422 000404      BR      16$          ;GO TO 16$ IF NO ERROR
      026424 000240      NOP                    ;RETURN HERE IF ERROR
      026426 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      026430 000137 026606  JMP      22$          ;GO TO 22$ IF ERROR
1842 026434      16$:
1843
1844      ;WAIT FOR READ TO COMPLETE AND GET STATUS
1845 026434 004737 036356      JSR    PC,TIMOUT     ;WAIT FOR READ TO COMPLETE
1846
1847 026440 004737 035534      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026444 000404      BR      17$          ;GO TO 17$ IF NO ERROR
      026446 000240      NOP                    ;RETURN HERE IF ERROR
      026450 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      026452 000137 026606  JMP      22$          ;GO TO 22$ IF ERROR
1848 026456      17$:
1849
1850      ;VERIFY THE RESULTS OF READ OPERATION
1851 026456 004737 036552      JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      026462 000405      BR      18$          ;GO TO 18$ IF NO ERROR
      026464 000240      NOP                    ;RETURN HERE IF ERROR
      026466 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      026470 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      026472 000137 026606  JMP      22$          ;GO TO 22$ IF ERROR
1852 026476      18$:
1853
    
```

```

1854 ;VERIFY THAT FORMAT ERROR IS SET
1855 026476 032737 000020 001352 BIT #FER,RMER1I ;IS FORMAT ERROR SET??
1856 026504 001022 BNE 20$ ;YES!!
1857
1858 026506 004737 051612 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
026512 000405 BR 19$ ;GO TO 19$ IF NO ERROR
026514 000240 NOP ;RETURN HERE IF ERROR
026516 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
026520 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
026522 000137 026606 JMP 22$ ;GO TO 22$ IF ERROR

1859
1860 026526 013737 001352 001142 19$: MOV RMER1I,$BDDAT ;BAD DATA FOR TYPEOUT
1861 026534 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED DATA
1862 026542 052737 000020 001140 BIS #FER,$GDDAT
1863 026550 104343 EMT 343
1864 026552
1865 026552 004737 037404 20$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
026556 000405 BR 21$ ;GO TO 21$ IF NO ERROR
026560 000240 NOP ;RETURN HERE IF ERROR
026562 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
026564 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
026566 000137 026606 JMP 22$ ;GO TO 22$ IF ERROR

1866 026572
1867 026572 032737 010000 001444 21$: BIT #FMT16,RMOFO ;TEST WRITE IN 18 BIT FORMAT AND
1868 ;READ IN 16 BIT FORMAT ?
1869 026600 001002 BNE 22$ ;BR IF YES
1870 026602 000137 025762 JMP 1$ ;TEST AGAIN
1871 026606
1872
1873
;*****
;*TEST 27 FORMAT HCE, FIRST HEADER WORD
;*****
TST27:
026606
026606 000004 SCOPE ;SCOPE CALL
026610 000240 NOP ;START OF TEST
026612 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
026616 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
026622 013701 001466 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
026626 012737 000027 001226 MOV #27,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

1874
1875 026634 012737 000001 027672 MOV #1,25$ ;INIT BIT POSITION
1876
1877 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1878 026642 1$:
1879 026642 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
1880 026650 012737 001057 001446 MOV #559.,RMDCO ;CYLINDER = 559.
1881 026656 012737 001000 001420 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
1882 026664 012737 177376 001414 MOV #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
1883 026672 012737 101716 001416 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1884 026700 012737 000062 001412 MOV #WH,RMCS10 ;WRITE HEADER AND DATA
1885
1886 ;VERIFY THAT SECTOR IS NOT BAD
026706 004737 032406 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
026712 000405 BR 2$ ;GO TO 2$ IF NO ERROR
026714 104401 063446 TYPE .SCTMSG ;TYPE BAD SECTOR MESSAGE
026720 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
026722 000137 027674 JMP 27$ ;GO TO 27$ IF ERROR
    
```

```

1887 026726          2$:
1888 026726 012737 065116 001174      MOV    #ZEROS,$TMP0      ;USE ALL ZEROS DATA PATTERN
1889 026734 012737 000001 001176      MOV    #1,$TMP1
1890 026742 004737 034640              JSR    PC,GENBUF        ;GO GENERATE DATA BUFFER
1891 026746
1892
1893
1894 026746 004737 031456      ;PREPARE DEVICE FOR DATA TRANSFER
      026752 154130          JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
                          .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                          ;CLEAR CONTROLLER & SELECT DEVICE
                          ;VERIFY CONTROLLER CLEAR OPERATION
                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                          ;VERIFY PACK ACKNOWLEDGE
                          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                          ;VERIFY RECALIBRATION
                          ;GO TO 4$ IF NO ERROR
                          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                          ;GO TO 27$ IF ERROR
      026754 000404          BR     4$
      026756 000240          NOP
      026760 104000          EMT
      026762 000137 027674      JMP    27$
1895 026766          4$:
1896
1897      ;COMPLEMENT DATA BIT IN FIRST HEADER WORD
1898 026766 033737 027672 101716      BIT    25$,BUFONE      ;IS BIT IN HEADER ON??
1899 026774 001404          BEQ    5$              ;NO!!
1900 026776 043737 027672 101716      BIC    25$,BUFONE      ;RESET BIT IN HEADER
1901 027004 000403          BR     6$
1902 027006 053737 027672 101716      5$:  BIS    25$,BUFONE      ;SET BIT IN HEADER
1903
1904      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1905 027014          6$:
1906 027014 012737 000063 001412      MOV    #WH!GO,RMCS10    ;WRITE HEADER AND DATA
1907 027022 012702 001555          MOV    #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
1908 027026 112722 000006          MOVB  #RMDA,(R2)+
1909 027032 112722 000034          MOVB  #RMDC,(R2)+
1910 027036 112722 000032          MOVB  #RMOF,(R2)+
1911 027042 112722 000002          MOVB  #RMWC,(R2)+
1912 027046 112722 000004          MOVB  #RMBA,(R2)+
1913 027052 112722 000000          MOVB  #RMCS1,(R2)+
1914 027056 112722 000200          MOVB  #200,(R2)+
1915
1916 027062 004737 036004          JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027066 000404          BR     7$              ;GO TO 7$ IF NO ERROR
      027070 000240          NOP                    ;RETURN HERE IF ERROR
      027072 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      027074 000137 027670      JMP    24$            ;GO TO 24$ IF ERROR
1917 027100          7$:
1918
1919      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1920 027100 004737 035450          JSR    PC,GETSTS
1921 027104 004737 036356          JSR    PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1922
1923 027110 004737 035534          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027114 000404          BR     8$              ;GO TO 8$ IF NO ERROR
      027116 000240          NOP                    ;RETURN HERE IF ERROR
      027120 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      027122 000137 027670      JMP    24$            ;GO TO 24$ IF ERROR

```

```

1924 027126      8$:
1925
1926
1927 027126 004737 036552      ;VERIFY RESULTS OF WRITE COMMAND
      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      BR     9$             ;GO TO 9$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      JMP    24$           ;GO TO 24$ IF ERROR

1928 027146      9$:
1929 027146 004737 051612      JSR    PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
      BR     10$          ;GO TO 10$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      JMP    24$           ;GO TO 24$ IF ERROR

1930 027166      10$:
1931 027166 004737 037404      JSR    PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
      BR     11$          ;GO TO 11$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      JMP    24$           ;GO TO 24$ IF ERROR

1932 027206      11$:
1933
1934      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
1935 027206 012737 000073 001412  MOV    #RH!GO,RMCS10      ;READ HEADER & DATA COMMAND
1936 027214 012737 102722 001416  MOV    #BUFTWO,RMBAO      ;CHANGE BUS ADDRESS
1937
1938 027222 004737 036004      JSR    PC,PUT             ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR     12$          ;GO TO 12$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP    24$           ;GO TO 24$ IF ERROR

1939 027240      12$:
1940
1941      ;WAIT FOR READ TO COMPLETE AND GET STATUS
1942 027240 004737 036356      JSR    PC,TIMOUT          ;WAIT FOR READ TO COMPLETE
1943
1944 027244 004737 035534      JSR    PC,GET             ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR     13$          ;GO TO 13$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      JMP    24$           ;GO TO 24$ IF ERROR

1945 027262      13$:
1946
1947      ;VERIFY THE RESULTS OF READ OPERATION
1948 027262 004737 036552      JSR    PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
      BR     14$          ;GO TO 14$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      JMP    24$           ;GO TO 24$ IF ERROR

1949 027302      14$:
1950 027302 032737 020000 027672  BIT    #SSF,25$          ;SHOULD SSE BE SET ?
1951 027310 001037          BNE    16$              ;YES !!
1952 027312 032737 140000 027672  BIT    #MSE!USE,25$      ;SHOULD BSE BE SET ?
  
```

```

1953 027320 001062          BNE      18$          :YES !!
1954 027322 032737 010000 027672  BIT      #FMT16,25$  :IS THIS FER ??
1955 027330 001505          BEQ      20$          :NO !!
1956
1957
1958 027332 032737 000020 001352  :VERIFY THAT FER IS SET
1959 027340 001136          BIT      #FER,RMER1I  :IS FER SET ??
1960
1961 027342 004737 051612          BNE      22$          :YES !!
      027346 000405          JSR      PC,DTASTS    :GO VERIFY RESULTS OF DATA TRANSFER
      027350 000240          BR       15$         :GO TO 15$ IF NO ERROR
      027352 104000          NOP      :RETURN HERE IF ERROR
      027354 004736          EMT     :ERROR # DEFINED BY DTASTS SUBROUTINE
      027356 000137 027670  JSR      PC,@(SP)+    :GO BACK FOR MORE ERROR CHECKS
      15$: JMP      24$         :GO TO 24$ IF ERROR
1962 027362
1963 027362 013737 001352 001142 15$: MOV      RMER1I,$BDDAT :RECEIVED STATUS
1964 027370 013737 001352 001140  MOV      RMER1I,$GDDAT :EXPECTED STATUS
1965 027376 052737 000020 001140  BIS      #FER,$GDDAT
1966 027404 104343          EMT     343
1967 027406 000530          BR       24$
1968
1969
1970 027410          :VERIFY THAT SSE IS SET
1971 027410 032737 000040 001400 16$: BIT      #SSE,RMER2I  :IS 'SSE' BIT SET ?
1972 027416 001107          BNE      22$          :YES !!
1973
1974 027420 004737 051612          JSR      PC,DTASTS    :GO VERIFY RESULTS OF DATA TRANSFER
      027424 000405          BR       17$         :GO TO 17$ IF NO ERROR
      027426 000240          NOP      :RETURN HERE IF ERROR
      027430 104000          EMT     :ERROR # DEFINED BY DTASTS SUBROUTINE
      027432 004736          JSR      PC,@(SP)+    :GO BACK FOR MORE ERROR CHECKS
      027434 000137 027670  JMP      24$         :GO TO 24$ IF ERROR
1975 027440
1976 027440 013737 001400 001142 17$: MOV      RMER2I,$BDDAT :GET RECIEVED STATUS
1977 027446 013737 001400 001140  MOV      RMER2I,$GDDAT :GET EXPECTED STATUS AND
1978 027454 052737 000040 001140  BIS      #SSE,$GDDAT  :SET SSE.
1979 027462 104224          EMT     224
1980 027464 000501          BR       24$          :SKIP REST OF TEST
1981
1982
1983 027466          :VERIFY THAT BSE IS SET
1984 027466 032737 100000 001400 18$: BIT      #BSE,RMER2I  :IS BSE SET ??
1985 027474 001060          BNE      22$          :YES !!
1986
1987 027476 004737 051612          JSR      PC,DTASTS    :GO VERIFY RESULTS OF DATA TRANSFER
      027502 000405          BR       19$         :GO TO 19$ IF NO ERROR
      027507 000240          NOP      :RETURN HERE IF ERROR
      027506 104000          EMT     :ERROR # DEFINED BY DTASTS SUBROUTINE
      027510 004736          JSR      PC,@(SP)+    :GO BACK FOR MORE ERROR CHECKS
      027512 000137 027670  JMP      24$         :GO TO 24$ IF ERROR
1988 027516
1989 027516 013737 001400 001142 19$: MOV      RMER2I,$BDDAT :RECEIVED STATUS
1990 027524 013737 001400 001140  MOV      RMER2I,$GDDAT :EXPECTED STATUS
1991 027532 052737 100000 001140  BIS      #BSE,$GDDAT
1992 027540 104345          EMT     345
1993 027542 000452          BR       24$          :SKIP REST OF TEST
1994
  
```

```

1995                                     ;VERIFY THAT HCE IS SET
1996 027544                               20$:
1997 027544 032737 000200 001352         BIT      #HCE,RMER1I          ;IS "HCE" SET??
1998 027552 001031                               BNE      22$                ;YES!!
1999
2000 027554 004737 051612                 JSR      PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
      027560 000405                               BR       21$                ;GO TO 21$ IF NO ERROR
      027562 000240                               NOP
      027564 104000                               EMT
      027566 004736                               JSR      PC,@(SP)+          ;ERROR # DEFINED BY DTASTS SUBROUTINE
      027570 000137 027670                               JMP      24$                ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 24$ IF ERROR
2001 027574                               21$:
2002 027574 013737 001352 001142         MOV      RMER1I,$BDDAT      ;RECEIVED STATUS FOR TYPEOUT
2003 027602 013737 001352 001140         MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
2004 027610 052737 000200 001140         BIS      #HCE,$GDDAT
2005 027616 012737 000001 001174         MOV      #1,$TMP0          ;GET HEADER WORD NUMBER
2006 027624 013737 027672 001176         MOV      25$,$TMP1         ;GET FAILING BIT POSITION
2007 027632 104344                               EMT
2008 027634 000415                               BR       24$
2009 027636                               22$:
2010
2011                                     ;CHECK FOR OTHER ERRORS
2012 027636 004737 037404                 JSR      PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
      027642 000405                               BR       23$                ;GO TO 23$ IF NO ERROR
      027644 000240                               NOP
      027646 104000                               EMT
      027650 004736                               JSR      PC,@(SP)+          ;ERROR # DEFINED BY SECERR SUBROUTINE
      027652 000137 027670                               JMP      24$                ;GO TO 24$ IF ERROR
2013 027656                               23$:
2014
2015                                     ;ADVANCE THE BIT POSITION AND FORMAT AGAIN IF NOT DONE
2016 027656 006537 027672                 ASL      25$                ;SHIFT TO NEXT BIT POSITION
2017 027662 001404                               BEQ      26$                ;EXIT IF DONE
2018 027664 000137 026642                 JMP      1$                 ;GO DO NEXT SECTOR
2019
2020 027670 000401                               24$: BR       27$          ;JUMP TO NEXT TEST
2021 027672 000000                               25$: .WORD    0            ;STORAGE FOR BIT POSITION
2022
2023 027674                               26$:
2024
2025 027674                               27$:
2026
2027
*****
;*TEST 30      FORMAT HCE, SECOND HEADER WORD
*****
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
2028
2029 027722 012737 000001 030516         MOV      #1,19$            ;INIT BIT POSITION
2030
2031                                     ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
2032 027730                               1$:

```



```

2033 027730 012737 001057 001446      MOV      #559.,RMDCO      :CYLINDER = 559.
2034 027736 012737 001000 001420      MOV      #TA2,RMDAO      :TRACK = 2, SECTOR = 0
2035 027744 012737 010000 001444      MOV      #FMT16,RMOFO    :16 BIT FORMAT
2036 027752 012737 177376 001414      MOV      #-258.,RMWCO    :2 + 256 WORDS (2'S COMP)
2037 027760 012737 101716 001416      MOV      #BUFONE,RMBAO   :DATA BUFFER ADDRESS
2038 027766 012737 000062 001412      MCV      #WH,RMCS10     :WRITE HEADER AND DATA
2039
2040      :VERIFY THAT SECTOR IS NOT BAD
          027774 004737 032406      JSR      PC,BADSCT      :CALL BAD SECTOR MODULE
          030000 000405      BR       2$             :GO TO 2$ IF NO ERROR
          030002 104401 063446      TYPE    ,SCTMSG       :TYPE BAD SECTOR MESSAGE
          030006 104000      EMT
          030010 000137 030520      JMP      21$          :ERROR # DEFINED BY BADSCT SUBROUTINE
          :GO TO 21$ IF ERROR
2041 030014      2$:
2042 030014 012737 065116 001174      MOV      #ZEROS,$TMP0   :USE ALL ZEROS DATA PATTERN
2043 030022 012737 000001 001176      MOV      #1,$TMP1
2044 030030 004737 034640      JSR      PC,GENBUF     :GO GENERATE DATA BUFFER
2045 030034
2046
2047      :PREPARE DEVICE FOR DATA TRANSFER
2048 030034 004737 031456      JSR      PC,TSTPRP     :PREPARE DEVICE FOR TEST
          030040 154130      .WORD   154130        :TASK DESCRIPTOR AS FOLLOWS:
          :SELECT DEVICE & VERIFY DEVICE AVAILABLE
          :CLEAR CONTROLLER & SELECT DEVICE
          :VERIFY CONTROLLER CLEAR OPERATION
          :PACK ACKNOWLEDGE IF VOLUME NOT VALID
          :VERIFY PACK ACKNOWLEDGE
          :RECALIBRATE IF "SKI" OR "PIP" IS SET
          :VERIFY RECALIBRATION
          :GO TO 4$ IF NO ERROR
          :RETURN HERE IF ERROR
          :ERROR # DEFINED BY TSTPRP SUBROUTINE
          :GO TO 21$ IF ERROR
          030042 000404      BR       4$
          030044 000240      NOP
          030046 104000      EMT
          030050 000137 030520      JMP      21$
2049
2050      :COMPLEMENT DATA BIT IN SECOND HEADER WORD
2051 030054      4$:
2052 030054 033737 030516 101720      BIT     19$,BUFONE+2   :IS BIT IN HEADER ON??
2053 030062 001404      BEQ     5$             :NO!!
2054 030064 043737 030516 101720      BIC     19$,BUFONE+2   :RESET BIT IN HEADER
2055 030072 000403      BR     6$
2056 030074 053737 030516 101720      5$: BIS     19$,BUFONE+2 :SET BIT IN HEADER
2057
2058      :SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
2059 030102      6$:
2060 030102 012737 000063 001412      MOV      #WH!GO,RMCS10  :WRITE HEADER AND DATA
2061 030110 012702 001555      MOV      #PUTINX,R2     :WRITE REGISTER INDEX TABLE
2062 030114 112722 000006      MOVB    #RMDA,(R2)+
2063 030120 112722 000034      MOVB    #RMDC,(R2)+
2064 030124 112722 000032      MOVB    #RMOF,(R2)+
2065 030130 112722 000002      MOVB    #RMWC,(R2)+
2066 030134 112722 000004      MOVB    #RMBA,(R2)+
2067 030140 112722 000000      MOVB    #RMCS1,(R2)+
2068 030144 112722 000200      MOVB    #200,(R2)+
2069
2070 030150 004737 036004      JSR      PC,PUT        :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          030154 000404      BR       7$           :GO TO 7$ IF NO ERROR
          030156 000240      NOP                 :RETURN HERE IF ERROR
    
```

```

030160 104000          EMT          :ERROR # DEFINED BY PUT SUBROUTINE
030162 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
2071 030166          7$:
2072
2073          :SETUP INPUT REGISTER BUFFER FOR READING STATUS
2074 030166 004737 035450 JSR PC,GETSTS
2075 030172 004737 036356 JSR PC,TIMOUT          :WAIT FOR COMMAND TO COMPLETE
2076
2077 030176 004737 035534 JSR PC,GET          :GO READ REGISTER(S) WITH GET SUBROUTINE
030202 000404          BR 8$          :GO TO 8$ IF NO ERROR
030204 000240          NOP          :RETURN HERE IF ERROR
030206 104000          EMT          :ERROR # DEFINED BY GET SUBROUTINE
030210 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
2078 030214          8$:
2079
2080          :VERIFY RESULTS OF WRITE COMMAND
2081 030214 004737 036552 JSR PC,PRIERR          :GO CHECK FOR PRIMARY ERRORS
030220 000405          BR 9$          :GO TO 9$ IF NO ERROR
030222 000240          NOP          :RETURN HERE IF ERROR
030224 104000          EMT          :ERROR # DEFINED BY PRIERR SUBROUTINE
030226 004736          JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
030230 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
2082 030234          9$:
2083 030234 004737 051612 JSR PC,DTASTS          :GO VERIFY RESULTS OF DATA TRANSFER
030240 000405          BR 10$         :GO TO 10$ IF NO ERROR
030242 000240          NOP          :RETURN HERE IF ERROR
030244 104000          EMT          :ERROR # DEFINED BY DTASTS SUBROUTINE
030246 004736          JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
030250 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
2084 030254          10$:
2085 030254 004737 037404 JSR PC,SECERR          :GO CHECK FOR SECONDARY ERRORS
030260 000405          BR 11$         :GO TO 11$ IF NO ERROR
030262 000240          NOP          :RETURN HERE IF ERROR
030264 104000          EMT          :ERROR # DEFINED BY SECERR SUBROUTINE
030266 004736          JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
030270 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
2086 030274          11$:
2087
2088          :READ HEADER AND DATA FOR SECTOR JUST WRITTEN
2089 030274 012737 000073 001412 MOV #RH!GO,RMCS10      :READ HEADER & DATA COMMAND
2090 030302 012737 102722 001416 MOV #BUFTWO,RMBAO      :CHANGE BUS ADDRESS
2091
2092 030310 004737 036004 JSR PC,PUT          :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030314 000404          BR 12$         :GO TO 12$ IF NO ERROR
030316 000240          NOP          :RETURN HERE IF ERROR
030320 104000          EMT          :ERROR # DEFINED BY PUT SUBROUTINE
030322 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
2093 030326          12$:
2094
2095          :WAIT FOR READ TO COMPLETE AND GET STATUS
2096 030326 004737 036356 JSR PC,TIMOUT          :WAIT FOR READ TO COMPLETE
2097
2098 030332 004737 035534 JSR PC,GET          :GO READ REGISTER(S) WITH GET SUBROUTINE
030336 000404          BR 13$         :GO TO 13$ IF NO ERROR
030340 000240          NOP          :RETURN HERE IF ERROR
030342 104000          EMT          :ERROR # DEFINED BY GET SUBROUTINE
030344 000137 030514  JMP 18$      :GO TO 18$ IF ERROR
    
```

```
2099 030350          13$:
2100
2101                ;VERIFY THE RESULTS OF READ OPERATION
2102 030350 004737 036552      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
                                BR      14$      ;GO TO 14$ IF NO ERROR
                                NOP                    ;RETURN HERE IF ERROR
                                EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR    PC,a(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP    18$      ;GO TO 18$ IF ERROR
2103 030370          14$:
2104
2105                ;VERIFY THAT HEADER COMPARE ERROR IS SET
2106 030370 032737 000200 001352  BIT    #HCE,RMER1I    ;IS 'HCE' SET??
2107 030376 001031                BNE    16$      ;YES!!
2108
2109 030400 004737 051612      JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR      15$      ;GO TO 15$ IF NO ERROR
                                NOP                    ;RETURN HERE IF ERROR
                                EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR    PC,a(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP    18$      ;GO TO 18$ IF ERROR
2110 030420          15$:
2111 030420 013737 001352 001142  MOV    RMER1I,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
2112 030426 013737 001352 001140  MOV    RMER1I,$GDDAT ;EXPECTED STATUS
2113 030434 052737 000200 001140  BIS    #HCE,$GDDAT
2114 030442 012737 000002 001174  MOV    #2,$TMP0     ;GET HEADER WORD NUMBER
2115 030450 013737 030516 001176  MOV    19$,$TMP1   ;GET FAILING BIT POSITION
2116 030456 104344
2117 030460 000415
2118 030462          16$:
2119
2120                ;CHECK FOR OTHER ERRORS
2121 030462 004737 037404      JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
                                BR      17$      ;GO TO 17$ IF NO ERROR
                                NOP                    ;RETURN HERE IF ERROR
                                EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
                                JSR    PC,a(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP    18$      ;GO TO 18$ IF ERROR
2122 030502          17$:
2123
2124                ;ADVANCE THE BIT POSITION AND FORMAT NEXT SECTOR IF NOT DONE
2125 030502 006337 030516      ASL    19$          ;SHIFT TO NEXT BIT POSITION
2126 030506 001404                BEQ    20$          ;EXIT IF DONE
2127 030510 000137 027730      JMP    1$          ;GO DO NEXT SECTOR
2128
2129 030514 000401          18$: BR      21$          ;JUMP OVER STORAGE
2130 030516 000000          19$: .WORD 0          ;STORAGE FOR BIT POSITION
2131
2132 030520          20$:
2133
2134 030520          21$:
2135
2136                ;*****
                ;*TEST 31          FORMAT FE CYLINDERS
                ;*****
TST31:                SCOPE          ;SCOPE CALL
030520 000004
```

```

030522 000240          NOP          :START OF TEST
030524 012706 001100  MOV          #STACK,SP      :INITIALIZE STACK POINTER
030530 013700 001276  MOV          $BASE,R0       :R0 = UNIBUS ADDRESS
030534 013701 001466  MOV          TSTQUE,R1      :(R1) = DEVICE BEING TESTED
030540 012737 000001  MOV          #1,$TIMES      :DO 1 ITERATION
030546 012737 000031  MOV          #31,$TESTN     :SET TEST NUMBER IN APT MAIL BOX

2137
2138
2139 030554 012737 001057 001446 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
2140 030562 012737 001000 001420 MOV          #559.,RMDCO    :CYLINDER = 559.
2141 030570          MOV          #TA2,RMDAO    :TRACK = 2, SECTOR = 0
2142 030570 012737 011000 001444 1$: MOV          #FMT16!SSEI,RMOFO :16 BIT FORMAT WITH SSEI SET
2143 030576 012737 177776 001414 MOV          #-2,RMWCO      :2 WORDS (2'S COMP)
2144 030604 012737 101716 001416 MOV          #BUFONE,RMBAO  :DATA BUFFER ADDRESS
2145 030612 012737 000062 001412 MOV          #WH,RMCS10    :WRITE HEADER AND DATA
2146
2147 ;VERIFY THAT SECTOR IS NOT BAD
030620 004737 032406  JSR          PC,BADSCT     :CALL BAD SECTOR MODULE
030624 000405          BR           2$          :GO TO 2$ IF NO ERROR
030626 104401 063446  TYPE         ,SCTMSG      :TYPE BAD SECTOR MESSAGE
030632 104000          EMT          :ERROR # DEFINED BY BADSCT SUBROUTINE
030634 000137 031146  JMP          12$          :GO TO 12$ IF ERROR

2148 030640          2$: CMP          RMDCO,#560.    :IS DESIRED CYLINDER, OUTSIDE FE CYLINDERS ?
2149 030640 023727 001446 001060 BLOS         3$          :NO !!
2150 030646 101402          JMP          12$          :YES, EXIT TEST
2151 030650 000137 031146  3$: JSR          PC,GENBUF    :GO GENERATE DATA BUFFER
2152 030654          4$:
2153 030654 004737 034640  JSR          PC,GENBUF    :GO GENERATE DATA BUFFER
2154 030660          5$:
2155 030660 005737 001514  TST          SSFENB        :TEST SSF ENABLE FOR 1ST HEADER WORD
2156 030664 001403          BEQ          5$          :NO, DO NOT SET 'SSF'
2157 030666 052737 020000 101716 BIS          #SSF,BUFONE   :YES, SET SSF BIT
2158 030674          5$:
2159
2160 ;PREPARE DEVICE FOR DATA TRANSFER
2161 030674 004737 031456  JSR          PC,TSTPRP     :PREPARE DEVICE FOR TEST
030700 154130          .WORD       154130 :TASK DESCRIPTOR AS FOLLOWS:
                                :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                :CLEAR CONTROLLER & SELECT DEVICE
                                :VERIFY CONTROLLER CLEAR OPERATION
                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                :VERIFY PACK ACKNOWLEDGE
                                :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                :VERIFY RECALIBRATION
                                :GO TO 6$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 12$ IF ERROR

030702 000404          BR           6$
030704 000240          NOP
030706 104000          EMT
030710 000137 031146  JMP          12$

2162
2163 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
2164 030714          6$:
2165 030714 012737 000063 001412 MOV          #WH!GO,RMCS10 :WRITE HEADER AND DATA
2166 030722 012702 001555          MOV          #PUTINX,R2   :WRITE REGISTER INDEX TABLE
2167 030726 112722 000006          MOVB         #RMDA,(R2)+
2168 030732 112722 000034          MOVB         #RMDC,(R2)+
2169 030736 112722 000032          MOVB         #RMOF,(R2)+
2170 030742 112722 000002          MOVB         #RMWC,(R2)+
    
```

```

2171 030746 112722 000004      MOVB  #RMBA,(R2)+
2172 030752 112722 000000      MOVB  #RMCS1,(R2)+
2173 030756 112722 000200      MOVB  #200,(R2)+
2174
2175 030762 004737 036004      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      030766 000404      BR    7$             ;GO TO 7$ IF NO ERROR
      030770 000240      NOP
      030772 104000      EMT
      030774 000137 031146      JMP   12$           ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 12$ IF ERROR
2176 031000      7$:
2177
2178 031000 004737 035450      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR   PC,GETSTS   ;GO TO GETSTS SUBROUTINE
2179
2180 031004 004737 036356      ;WAIT FOR COMMAND TO COMPLETE
      JSR   PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
2181
2182 031010 004737 035534      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031014 000404      BR    8$             ;GO TO 8$ IF NO ERROR
      031016 000240      NOP
      031020 104000      EMT
      031022 000137 031146      JMP   12$           ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 12$ IF ERROR
2183 031026      8$:
2184
2185
2186 031026 004737 036552      ;VERIFY RESULTS OF WRITE COMMAND
      JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      031032 000405      BR    9$             ;GO TO 9$ IF NO ERROR
      031034 000240      NOP
      031036 104000      EMT
      031040 004736      JSR   PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      031042 000137 031146      JMP   12$           ;GO TO 12$ IF ERROR
2187 031046      9$:
2188 031046 004737 037404      JSR   PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      031052 000405      BR    10$            ;GO TO 10$ IF NO ERROR
      031054 000240      NOP
      031056 104000      EMT
      031060 004736      JSR   PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      031062 000137 031146      JMP   12$           ;GO TO 12$ IF ERROR
2189 031066      10$:
2190
2191
2192 031066 005237 001420      ;INCREMENT ADDRESS
      INC   RMDAO
      2193 031072 123727 001420 000037      CMPB  RMDAO,#31.    ;ADVANCE SECTOR COUNT
      2194 031100 101420      BLOS  11$            ;DONE ALL SECTORS ?
      ;NO !!
2195
2196 031102 105037 001420      CLRB  RMDAO          ;START AT SECTOR 0 AND
      2197 031106 105237 001421      INCB  RMDAO+1        ;ADVANCE TO NEXT TRACK.
      2198 031112 123737 001421 001335      CMPB  RMDAO+1,LSTRK+1 ;DONE WITH LAST TRACK ?
      2199 031120 101410      BLOS  11$            ;NO !!
2200
2201 031122 105037 001421      CLRB  RMDAO+1        ;START WITH TRACK 0 AND
      2202 031126 005237 001446      INC   RMDCO          ;ADVANCE TO NEXT CYLINDER.
      2203 031132 023727 001446 001060      CMP   RMDCO,#560.   ;DONE WITH LAST CYLINDER ?
      2204 031140 101002      BHI   12$            ;YES !!
      2205 031142 000137 030570      11$: JMP   1$             ;GO DO TEST AGAIN
2206
2207 031146      12$:
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

.SBTTL END OF SUB-PASS ROUTINE

:THIS IS THE END OF SUB-PASS ROUTINE. THIS ROUTINE IS USED TO
:TERMINATE THE OPERATION OF THE CURRENT DEVICE UNDER TEST AND
:SELECT THE NEXT DEVICE FOR TEST. IF THERE ARE NO MORE DEVICES
:TO TEST, EXIT IS MADE TO 'SEOP' ROUTINE. OTHERWISE, RETURN
:IS MADE TO 'SHUT' ROUTINE.

```
031146 000004 $EOSP: SCOPE
031150 000240 NOP
031152 013700 001466 MOV TSTQUE,RO ;GET POINTER TO TSTQUE
031156 062700 000002 ADD #2,RO ;ADJUST POINTER TO NEXT DEVICE
031162 010037 001466 MOV RO,TSTQUE ;SAVE POINTER TO TSTQUE
031166 005710 TST (RO) ;ANY MORE DEVICES FOR TEST ?
031170 001402 BEQ 1$ ;BR IF NO
031172 000137 031422 JMP SHUT ;YES, JUMP TO 'SHUT' ROUTINE
031176 012737 001470 001466 1$: MOV #TSTQUE+2,TSTQUE ;INITIALIZE POINTER TO FIRST DEVICE IN
;TEST QUE TABLE
```

.SBTTL END OF PASS ROUTINE

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

```
031204 $EOP: NOP
031204 000240 CLR $TSTNM ;:ZERO THE TEST NUMBER
031206 005037 001116 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
031212 005037 001206 INC $PASS ;:INCREMENT THE PASS NUMBER
031216 005237 001230 BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
031222 042737 100000 001230 DEC (PC)+ ;:LOOP?
031230 005327 $EOPCT: .WORD 1
031232 000001 BGT $DOAGN ;:YES
031234 003066 MOV (PC)+,a(PC)+ ;:RESTORE COUNTER
031236 012737 $ENDCT: .WORD 1
031240 000001 $EOPCT
031242 031232 TYPE ,65$ ;:TYPE ASCIZ STRING
031244 104401 031252 BR 64$ ;:GET OVER THE ASCIZ
031250 000407 ;:65$: .ASCIZ <12><15>/END PASS #/
64$: MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
;:TYPE PASS NUMBER
031270 013746 001230 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
031274 104405 TST $ERTTL ;:SEE IF ANY ERRORS THIS PASS
031276 005737 001126 BEQ $GT42P ;:BR IF NO ERRORS TO REPORT
031302 001431 TYPE ,67$ ;:TYPE ASCIZ STRING
031304 104401 031312 BR 66$ ;:GET OVER THE ASCIZ
031310 000421 ;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$: MOV $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
;:TOTAL NUMBER OF ERRORS
031354 013746 001126 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
031360 104405 CLR $ERTTL ;:CLEAR ERROR TOTAL
031362 005037 001126
```

031366	104401	001217		SGT42P: TYPE	, \$CRLF	:::TYPE CARRIAGE RETURN, LINE FEED
031372	013700	000042		\$GET42: MOV	@#42,R0	:::GET MONITOR ADDRESS
031376	001405			BEQ	\$DOAGN	:::BRANCH IF NO MONITOR
031400	000005			RESET		:::CLEAR THE WORLD
031402	004710			SENDAD: JSR	PC,(R0)	:::GO TO MONITOR
031404	000240			NOP		:::SAVE ROOM
031406	000240			NOP		:::FOR
031410	000240			NOP		:::ACT11
031412				SDOAGN:		
031412	000137			JMP	@(PC)+	:::RETURN
031414	031422			SRTNAD: .WORD	SHUT	
031416	377	377	000	SENULL: .BYTE	-1,-1,0	:::NULL CHARACTER STRING
				.EVEN		
21						
22	031422	005737	001326	SHUT: TST	CTLFG	:::WAS CONTROL C FLAGGED ?
23	031426	001002		BNE	1\$:::BR IF YES
24	031430	000137	007524	JMP	READY	:::CONTINUE
25	031434	005737	000042	1\$: TST	@#42	:::ANY MONITOR PRESENT ?
26	031440	001002		BNE	2\$:::BR IF YES
27	031442	000137	005434	JMP	START	:::GO TO START
28	031446	005037	001300	2\$: CLR	\$DEVN	:::FUDGE NO DRIVES
29	031452	000137	031204	JMP	\$EOP	:::RETURN TO \$EOP

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.SBTTL TEST PREPARATION MODULE

: THIS MODULE PREPARES THE 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          TASK/VERIFY DESCRIPTOR
    .WORD  NNNNNN            RETURN HERE IF NO ERROR
    BR     ??                 RETURN HERE IF ERROR
    NOP
    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 OR SKI ERROR
    BIT 10 = 1  RECALIBRATE DRIVE
    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

```

01 031456 017637 000000 032402  MOV    @ (SP),39$      ;STORE DESCRIPTOR
02 031464 062716 000006          ADD    #6,(SP)        ;MOVE SP TO USERS ERROR CALL
03 031470 105076 000000          (LRB  @ (SP)         ;CLEAR ERROR CALL
04 031474 162716 000004          SUB    #4,(SP)        ;MOVE SP TO NO ERROR RETURN
05 031500 004737 035450          JSR    PC,GETSTS     ;SETUP TO READ ALL REGISTERS
06 031504 004737 035534          JSR    PC,GET        ;GET RMER2
07 031510 000411                  BR     2$            ;BR IF NO ERROR DETECTED
08 031512 000401                  BR     1$            ;GET OVER ERROR NUMBER
09 031514 000000                  .WORD  0             ;ERROR DEFINED BY GET SUBROUTINE
10 031516 062716 000004 1$:    ADD    #4,(SP)        ;XFER ERROR TO USER AND
11 031522 113776 031514 000000  MOVB  1$-2,@ (SP)    ;GET ERROR NUMBER.
12 031530 000137 032372          JMP    37$
13 031534 013737 001400 032404 2$:    MOV    RMER2I,40$    ;GET RMER2 AND SAVE FOR LATER
    
```

;;*****


```

58                                     ;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
59 031542 005737 032402                TST      39$                ;SELECT DEVICE??
60 031546 100014                        BPL      4$                ;NO!!
61
62 031550 004737 043762                JSR      PC,DEVSEL        ;GO SELECT DEVICE
63 031554 000411                        BR       4$                ;NO ERROR - CONTINUE
64 031556 000401                        BR       3$
65 031560 000000                        .WORD   0                ;ERROR NUMBER FROM DEVSEL
66 031562 062716 000004 000000 3$:   ADD      #4,(SP)          ;TRANSFER ERROR TO USER
67 031566 113776 031560 000000        MOVB    3$,2,a(SP)
68 031574 000137 032372                JMP      37$
69
70                                     ;*****
71                                     ;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
72 031600                                4$:   BIT      #BIT14,39$      ;CLEAR CONTROLLER??
73 031600 032737 040000 032402        BEQ     13$                ;NO!!
74 031606 001451
75
76 031610 004737 045454                JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
77 031614 000411                        BR       7$                ;CONTINUE - NO ERROR
78 031616 000401                        BR       6$
79 031620 000000                        .WORD   0                ;ERROR NUMBER FROM CNTCLR
80 031622 062716 000004 000000 5$:   ADD      #4,(SP)          ;TRANSFER ERROR TO USER
81 031626 113776 031620 000000 6$:   MOVB    5$,a(SP)
82 031634 000137 032372                JMP      37$
83
84                                     ;*****
85                                     ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
86 031640                                7$:   BIT      #BIT6,39$      ;VERIFY??
87 031640 032737 000100 032402        BEQ     13$                ;NO!!
88 031646 001431
89
90 031650 004737 035534                JSR      PC,GET           ;GO GET STATUS
91 031654 000411                        BR       10$              ;NO ERROR GETTING STATUS
92 031656 000401                        BR       9$
93 031660 000000                        .WORD   0                ;ERROR FROM GETTING STATUS
94 031662 062716 000004 000000 8$:   ADD      #4,(SP)          ;TRANSFER ERROR TO USER
95 031666 113776 031660 000000 9$:   MOVB    8$,a(SP)
96 031674 000137 032372                JMP      37$
97
98 031700 004737 045572                10$:   JSR      PC,CLRSTS    ;GO VERIFY STATUS CLEAR
99 031704 000412                        BR       13$              ;NO ERROR IN CLEAR
100 031706 000401                        BR       12$
101 031710 000000                        .WORD   0                ;ERROR IN STATUS CLEAR
102 031712 005726                        TST     (SP)+             ;STRIP RETURN ADDRESS TO
103 031714 062716 000004 000000 11$:   ADD      #4,(SP)          ;SUBROUTINE AND TRANSFER
104 031720 113776 031710 000000 12$:   MOVB    11$,a(SP)        ;ERROR TO USER
105 031726 000137 032372                JMP      37$
106
107                                     ;*****
108                                     ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
109                                     ;NOT VALID
110 031732                                13$:   BIT      #BIT12,39$     ;PACK ACKNOWLEDGE??
111 031732 032737 010000 032402        BEQ     25$                ;NO!!
112 031740 001501
113
114 031742 004737 035534                JSR      PC,GET
    
```

```

115 031746 000411 BR 16$ :NO ERROR GETTING RMDS
116 031750 000401 BR 15$
117 031752 000000 14$: .WORD 0
118 031754 062716 000004 15$: ADD #4,(SP) :TRANSFER ERROR TO USER
119 031760 113776 031752 000000 MOVB 14$,a(SP)
120 031766 000137 032372 JMP 37$
121
122 031772 032737 000100 001350 16$: BIT #VV,RMDSI :IS VOLUME VALID??
123 032000 001061 BNE 25$ :YES!!
124
125 032002 012737 000023 001412 MOV #PAKACK!GO,RMCS10 :LOAD PACK ACK COMMAND
126 032010 112737 000000 001555 MOVB #RMCS1,PUTINX :SETUP REGISTER INDEX TABLE
127 032016 112737 000200 001556 MOVB #200,PUTINX+1
128 032024 004737 036004 JSR PC,PUT :GO WRITE COMMAND
129 032030 000410 BR 19$ :NO ERROR LOADING REGISTER
130 032032 000401 BR 18$
131 032034 000000 17$: .WORD 0 :ERROR FROM PUT SUB
132 032036 062716 000004 18$: ADD #4,(SP) :TRANSFER ERROR TO USER
133 032042 113776 032034 000000 MOVB 17$,a(SP)
134 032050 000550 BR 37$
135
136 032052 004737 036356 19$: JSR PC,TIMOUT :WAIT FOR COMMAND TO COMPLETE
137
138 :*****
139 :VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
140 032056 032737 000020 032402 BIT #BIT4,39$ :VERIFY PACK ACKNOWLEDGE??
141 032064 001427 BEQ 25$ :NO!!
142
143 032066 004737 035534 JSR PC,GET :GO GET STATUS
144 032072 000410 BR 22$ :NO ERROR GETTING STATUS
145 032074 000401 BR 21$
146 032076 000000 20$: .WORD 0 :ERROR FROM GET SUB
147 032100 062716 000004 21$: ADD #4,(SP) :TRANSFER ERROR TO USER
148 032104 113776 032076 000000 MOVB 20$,a(SP)
149 032112 000527 BR 37$
150
151 032114 004737 046452 22$: JSR PC,ACKSTS :GO CHECK ACKNOWLEDGE
152 032120 000411 BR 25$ :NO ERROR
153 032122 000401 BR 24$
154 032124 000000 23$: .WORD 0 :PACK ACKNOWLEDGE ERROR
155 032126 005726 24$: TST (SP)+ :STRIP RETURN TO SUB AND
156 032130 062716 000004 ADD #4,(SP) :TRANSFER ERROR TO USER
157 032134 113776 032124 000000 MOVB 23$,a(SP)
158 032142 000513 BR 37$
159
160 :*****
161 :RECALIBRATE DRIVE IF BIT 11 IS SET IN TASK AND "SKI" IS SET
162 :OR "PIP" IS ACTIVE.
163 :RECALIBRATE DRIVE IF BIT 10 IS SET
164 032144 25$:
165 032144 032737 002000 032402 BIT #BIT10,39$ :RECALIBRATE DRIVE ?
166 032152 001027 BNE 29$ :YES!!
167 032154 032737 004000 032402 BIT #BIT11,39$ :RECALIBRATE??
168 032162 001505 BEQ 38$ :NO!!
169
170 032164 004737 035534 JSR PC,GET :GO GET RMDS
171 032170 000410 BR 28$ :NO ERROR GETTING RMDS
    
```

```

172 032172 000401          BR      27$
173 032174 000000          .WORD  0          ;ERROR FROM GET SUB
174 032176 062716 000004 26$:   ADD    #4,(SP)   ;TRANSFER ERROR TO USER
175 032202 113776 032174 000000 27$:   MOVB   26$,a(SP)
176 032210 000470          BR      37$
177
178 032212 032737 040000 032404 28$:   BIT    #SKI,40$   ;WAS SKI SET ?
179 032220 001004          BNE    29$        ;YES, GO RECALIBRATE
180 032222 032737 020000 001350          BIT    #PIP,RMDSI ;IS PIP ACTIVE??
181 032230 001462          BEQ    38$        ;NO!!
182
183 032232 012737 000007 001412 29$:   MOV    #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
184 032240 112737 000000 001555          MOVB   #RMCS1,PUTINX ;AND REGISTER INDEX
185 032246 112737 000200 001556          MOVB   #200,PUTINX+1 ;SET TERMINATOR
186 032254 004737 036004          JSR    PC,PUT     ;GO ISSUE RECALIBRATE
187 032260 000410          BR     31$        ;NO ERROR
188 032262 000401          BR     30$
189 032264 000000          .WORD  0          ;ERROR IN REGISTER TRANSFER
190 032266 062716 000004 30$:   ADD    #4,(SP)   ;TRANSFER ERROR TO USER
191 032272 113776 032264 000000          MOVB   30$-2,a(SP)
192 032300 000434          BR     37$
193
194 032302 004737 036356          31$:   JSR    PC,TIMOUT ;WAIT FOR COMPLETION
195
196                                     ;*****
197                                     ;VERIFY RECALIBRATE IF BIT 3 SET IN TASK
198 032306 032737 000010 032402          BIT    #BIT3,39$   ;VERIFY RECALIBRATE??
199 032314 001430          BEQ    38$        ;NO!!
200
201 032316 004737 035534          JSR    PC,GET     ;GO GET STATUS
202 032322 000410          BR     34$        ;NO ERROR GETTING STATUS
203 032324 000401          BR     33$
204 032326 000000          32$:   .WORD  0          ;ERROR FROM GET
205 032330 062716 000004 33$:   ADD    #4,(SP)   ;TRANSFER ERROR TO USER
206 032334 113776 032326 000000          MOVB   32$,a(SP)
207 032342 000413          BR     37$
208
209 032344 004737 047246          34$:   JSR    PC,RCLSTS ;GO CHECK RECALIBRATE
210 032350 000412          BR     38$        ;NO ERROR DURING RECALIBRATE
211 032352 000401          BR     36$
212 032354 000000          35$:   .WORD  0          ;ERROR DURING RECALIBRATE
213 032356 005726          36$:   TST    (SP)+    ;STRIP RETURN TO SUB AND
214 032360 062716 000004          ADD    #4,(SP)   ;TRANSFER ERROR TO USER
215 032364 113776 032354 000000          MOVB   35$,a(SP)
216 032372 162716 000002          37$:   SUB    #2,(SP)   ;MOVE SP BACK BEFORE ERROR
217 032376 000240          38$:   NOP
218 032400 000207          RTS    PC        ;RETURN TO USER
219
220 032402 000000          39$:   .WORD  0          ;TASK/VERIFY DESCRIPTOR
221 032404 000000          40$:   .WORD  0          ;CONTAINS RMER2
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

.SBTTL BAD SECTOR MODULE

: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 'ASNDA' AND 'ASNDC'
 :SO THAT A REFERENCE IS AVAILABLE TO THE TEST OUTSIDE OF THE PUT BUFFER.

:THE BAD SECTOR MODULE PERFORMS TWO MAJOR FUNCTIONS:

- (1) RECOVER THE BAD SECTOR FILES AND
- (2) APPROVE THE USAGE OF A SECTOR BASED ON INFORMATION IN
 THE BAD SECTOR FILES OR ASSIGN A NEW SECTOR IF THE ONE
 ELECTED IS NOT AVAILABLE FOR USE.

:INFORMATION REQUIRED BY THE MODULE INCLUDES:

- (1) .RMDCO - THE DESIRED CYLINDER,
- (2) .RMDAO - THE TRACK AND SECTOR ADDRESS,
- (3) .RMWCO - THE WORD COUNT,
- (4) .RMCS10 - THE COMMAND,
- (5) .RMOFO - THE FORMAT MODE AND SKIP SECTOR ERROR INHIBIT.

:CALL:

```
JSR    PC,BADSCT    ;CALL SUBROUTINE
BR     ???          ;RETURN HERE IF NO ERROR
TYPE  ,MESSAGE     ;RETURN HERE IF THE BAD SECTOR FILE
                        ;CANNOT BE RECOVERED.
ERROR N            ;THE EMT OFFSET NUMBER 'N' IS DEFINED
                        ;BY BAD SECTOR MODULE.
```

```
BADSCT: ADD    #6,(SP)    ;CLEAR ERROR NUMBER IN USER'S
CLR    @ (SP)        ;ERROR CALL.
SUB    #6,(SP)
```

:TEST 'MEDIA ENABLE' TO DETERMINE WHETHER OR NOT THE BAD SECTOR FILES
 :HAVE BEEN RECOVERED.

```
TST    MEDENB      ;HAVE BAD SECTOR FILES BEEN RECOVERED ?
BEQ    1$          ;BR IF NO
JMP    56$        ;YES, BAD SECTOR FILE IS AVAILABLE
```

:RECOVER SKIP SECTOR FILE FROM CYLINDER = 559., TRACK = 0 AND
 :RECOVER THE MANUFACTURES / USERS BAD SECTOR FILE FROM CYLINDER = 558.,
 :TRACK = LAST TRACK (RM80 = 13.). ALSO, SAVE THE USER'S PUT BUFFER

```
1$:
MOV    R0,-(SP)    ;;PUSH R0 ON STACK
CLR    R0          ;START WITH RMCS1
2$:    MOV    PUTBUF(R0),BUFFER(R0)
ADD    #2,R0      ;ADVANCE TO NEXT BUFFER POSITION
CMP    #46,R0     ;END OF BUFFER
BHS   2$          ;NO !!
```

:SET RETRY COUNT AND LOAD PUT BUFFER AND REGISTER INDEX TABLE
 :SETUP PARAMETERS TO READ SKIP SECTOR FILE FIRST (3 SECTORS)

```
MOV    #3,74$     ;RETRY COUNT
MOV    #559.,RMDCO ;DESIRED CYLINDER = 559.
```

```
032406 062716 000006
032412 105076 000000
032416 162716 000006
032422 005737 001512
032426 001402
032430 000137 033764
032434 010046
032436 005000
032440 016060 001412 101716
032446 062700 000002
032452 022700 000046
032456 103370
032460 012737 000003 034500
032466 012737 001057 001446
```

```

57 032474 012737 000000 001420      MOV      #0,RMDAO      ;STARTING TRACK = 0, SECTOR = 0
58 032502 012737 176372 001414      MOV      #-258.*3,RMWC0 ;258. * 3 SECTORS (2'S COMP)
59 032510 012737 010000 001444      MOV      #FMT16,RMOFO  ;16 BIT FORMAT
60 032516 012737 102722 001416      MOV      #BUFTWO,RMBAO ;SETUP BUS ADDRESS TO READ SKIP SECTOR FILE
61
62 032524 012700 001555      MOV      #PUTINX,R0     ;R0 POINTS TO REGISTER INDEX TABLE
63 032530 112720 000006      MOV      #RMDA,(R0)+
64 032534 112720 000034      MOV      #RMDC,(R0)+
65 032540 112720 000002      MOV      #RMWC,(R0)+
66 032544 112720 000032      MOV      #RMOF,(R0)+
67 032550 112720 000004      MOV      #RMBA,(R0)+
68 032554 112720 000000      MOV      #RMCS1,(R0)+
69 032560 112720 000200      MOV      #200,(R0)+
70 032564 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
71
72      ;SET GET INDEX TABLE FOR READING STATUS
73 032566      3$:
74 032566 004737 035450      JSR      PC,GETSTS     ;SETUP GET INDEX REGISTER FOR STATUS
75 032572 004737 035534      JSR      PC,GET       ;GET REGISTERS
76 032576 000411      BR      5$            ;BR IF NO ERROR
77 032600 000401      BR      4$            ;JUMP OVER ERROR NUMBER
78 032602 000000      .WORD   0            ;ERROR DEFINED BY GET SUB
79 032604 062716 000006      4$: ADD      #6,(SP)    ;XFER ERROR TO USER AND
80 032610 113776 032602 000000      MOV      4$-2,a(SP)  ;GET ERROR NUMBER.
81 032616 000137 033412      JMP      42$
82
83 032622 013737 001400 034512 5$: MOV      RMER2I,79$   ;GET RMER2 AND SAVE FOR LATER
84
85      ;CLEAR THE DEVICE USING DRIVE CLEAR COMMAND
86 03263C 012737 000011 001412      MOV      #DRVCLR!GO,RMCS10 ;LOAD COMMAND IN PUT BUFFER
87 032636 004737 036004      JSR      PC,PUT       ;OUTPUT COMMAND
88 032642 000411      BR      8$            ;RETURN HERE IF NO ERROR
89 032644 000401      BR      7$            ;GET AROUND ERROR #
90 032646 000000      6$: .WORD   0            ;ERROR # GOES HERE
91 032650 062716 000006      7$: ADD      #6,(SP)    ;MOVE SP TO USERS ERROR CALL
92 032654 113776 032646 000000      MOV      6$,a(SP)    ;MOVE ERROR NUMBER TO USER
93 032662 000137 033412      JMP      42$
94
95 032666 004737 036356      8$: JSR      PC,TIMOUT   ;WAIT FOR COMPLETION
96 032672 004737 035534      JSR      PC,GET       ;GO GET STATUS
97 032676 000411      BR      11$           ;RETURN HERE IF NO ERROR
98 032700 000401      BR      10$          ;GET AROUND ERROR #
99 032702 000000      9$: .WORD   0            ;ERROR # GOES HERE
100 032704 062716 000006      10$: ADD     #6,(SP)    ;MOVE SP TO USERS ERROR CALL
101 032710 113776 032702 000000      MOV      9$,a(SP)    ;MOVE ERROR # TO USERS ERROR CALL
102 032716 000137 033412      JMP      42$
103
104 032722 004737 051010      11$: JSR      PC,DRVSTS  ;GO VERIFY DRIVE CLEAR COMMAND
105 032726 000412      BR      14$           ;RETURN HERE IF NO ERROR
106 032730 000401      BR      13$           ;GET AROUND ERROR
107 032732 000000      12$: .WORD   0            ;ERROR # GOES HERE
108 032734 005726      13$: TST      (SP)+     ;STRIP RETURN TO SUBROUTINE
109 032736 062716 000006      ADD      #6,(SP)    ;MOVE SP TO USERS ERROR CALL
110 032742 113776 032732 000000      MOV      12$,a(SP)  ;MOVE ERROR # TO USER CALL
111 032750 000137 033412      JMP      42$
112
113      ;ISSUE A PACK ACKNOWLEDGE IF VOLUME VALID IS RESET
    
```

```

114 032754
115 032754 032737 000100 001350 14$: BIT #VV,RMDSI :IS VV RESET ??
116 032762 001052 BNE 23$ :NO !!
117
118 032764 012737 000023 001412 MOV #PACACK!GO,RMCS10 :LOAD COMMAND
119 032772 004737 036004 JSR PC,PUT :GO PUT COMMAND TO DRIVE
120 032776 000411 BR 17$ :RETURN HERE IF NO OUTPUT ERROR
121 033000 000401 BR 16$ :GET AROUND ERROR #
122 033002 000000 15$: .WORD 0 :ERROR # GOES HERE
123 033004 062716 000006 16$: ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
124 033010 113776 033002 000000 MOVB 15$,a(SP) :MOVE ERROR # TO ERROR CALL
125 033016 000137 033412 JMP 42$
126
127 033022 004737 036356 17$: JSR PC,TIMOUT :WAIT FOR COMPLETION
128 033026 004737 035534 JSR PC,GET :GO GET STATUS FOR PACK ACK
129 033032 000411 BR 20$ :RETURN HERE IF NO ERROR
130 033034 000401 BR 19$ :GET AROUND ERROR #
131 033036 000000 18$: .WORD 0 :ERROR # GOES HERE
132 033040 062716 000006 19$: ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
133 033044 113776 033036 000000 MOVB 18$,a(SP) :MOVE ERROR # TO CALL
134 033052 000137 033412 JMP 42$
135
136 033056 004737 046452 20$: JSR PC,ACKSTS :GO VERIFY ACKNOWLEDGE STATUS
137 033062 000412 BR 23$ :RETURN HERE IF NO ERROR
138 033064 000401 BR 22$ :GET AROUND ERROR #
139 033066 000000 21$: .WORD 0 :ERROR # GOES HERE
140 033070 005726 22$: TST (SP)+ :STRIP RETURN TO SUBROUTINE
141 033072 062716 000006 ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
142 033076 113776 033066 000000 MOVB 21$,a(SP) :MOVE ERROR # TO USERS ERROR CALL
143 033104 000137 033412 JMP 42$
144
145 :RECALIBRATE THE DRIVE IF "SKI" OR "PIP IS SET
23$:
146 033110
147 033110 032737 040000 034512 BIT #SKI,79$ :WAS SKI SET ?
148 033116 001004 BNE 24$ :YES, GO RECALIBRATE
149 033120 032737 020000 001350 BIT #PIP,RMDSI :IS PIP SET ??
150 033126 001452 BEQ 32$ :NO !!
151
152 033130 012737 000007 001412 24$: MOV #RECAL!GO,RMCS10 :LOAD RECALIBRATE COMMAND
153 033136 004737 036004 JSR PC,PUT :PUT THE RECAL COMMAND
154 033142 000411 BR 26$ :RETURN HERE IF NO ERROR
155 033144 000401 BR 25$ :GET AROUND ERROR #
156 033146 000000 .WORD 0 :ERROR # GOES HERE
157 033150 062716 000006 25$: ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
158 033154 113776 033146 000000 MOVB 25$-2,a(SP) :MOVE ERROR # TO USERS CALL
159 033162 000137 033412 JMP 42$
160
161 033166 004737 036356 26$: JSR PC,TIMOUT :WAIT FOR RECALIBRATE TO COMPLETE
162 033172 004737 035534 JSR PC,GET :GO GET RECAL STATUS
163 033176 000411 BR 29$ :RETURN HERE IF NO ERROR
164 033200 000401 BR 28$ :GET AROUND ERROR #
165 033202 000000 27$: .WORD 0 :ERROR # GOES HERE
166 033204 062716 000006 28$: ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
167 033210 113776 033202 000000 MOVB 27$,a(SP) :MOVE ERROR TO USERS CALL
168 033216 000137 033412 JMP 42$
169
170 033222 004737 047246 29$: JSR PC,RCLSTS :GO VERIFY RECALIBRATE STATUS
    
```

```

171 033226 000412 BR 32$ :RETURN HERE IF NO ERROR
172 033230 000401 BR 31$ :GET AROUND ERROR #
173 033232 000000 30$: .WORD 0 :ERROR # GOES HERE
174 033234 005726 31$: TST (SP)+ :STRIP RETURN TO SUBROUTINE
175 033236 062716 000006 ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
176 033242 113776 033232 000000 MOVB 30$,a(SP) :MOVE ERROR # TO USER CALL
177 033250 000137 033412 JMP 42$
178
179 :READ THE SECTOR IDENTIFIED BY RMDAO, INCLUDING HEADER AND DATA
180 033254 32$:
181 033254 012737 000073 001412 MOV #RH!GO,RMCS10 :LOAD READ HEADER AND DATA COMMAND
182 033262 004737 036004 JSR PC,PUT :PUT COMMAND
183 033266 000411 BR 35$ :RETURN HERE IF NO ERROR
184 033270 000401 BR 34$ :GET AROUND ERROR #
185 033272 000000 33$: .WORD 0 :ERROR # GOES HERE
186 033274 062716 000006 34$: ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
187 033300 113776 033272 000000 MOVB 33$,a(SP) :MOVE ERROR # TO USERS ERROR CALL
188 033306 000137 033412 JMP 42$
189
190 033312 004737 036356 35$: JSR PC,TIMOUT :WAIT FOR READ OPERATION TO COMPLETE
191 033316 004737 035534 JSR PC,GET :GO GET STATUS FOR READ OPERATION
192 033322 000411 BR 38$ :RETURN HERE IF NO ERROR
193 033324 000401 BR 37$ :GET AROUND ERROR #
194 033326 000000 36$: .WORD 0 :ERROR # GOES HERE
195 033330 062716 000006 37$: ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
196 033334 113776 033326 000000 MOVB 36$,a(SP) :MOVE ERROR # TO CALL
197 033342 000137 033412 JMP 42$
198
199 033346 004737 051612 38$: JSR PC,DTASTS :GO VERIFY RESULTS OF READ OPERATION
200 033352 000412 BR 41$ :RETURN HERE IF NO ERROR
201 033354 000401 BR 40$ :GET AROUND ERROR #
202 033356 000000 39$: .WORD 0 :ERROR # GOES HERE
203 033360 005726 40$: TST (SP)+ :STRIP RETURN ADDRESS TO SUBROUTINE
204 033362 062716 000006 ADD #6,(SP) :MOVE SP TO USERS ERROR CALL
205 033366 113776 033356 000000 MOVB 39$,a(SP) :MOVE ERROR # TO USERS CALL
206 033374 000137 033412 JMP 42$
207
208 033400 032737 040000 001336 41$: BIT #TRE,RMCS1I :ANY CONTROLLER ERRORS ?
209 033406 001001 BNE 42$ :BR IF YES
210 033410 000477 BR 51$ :NO ERRORS DETECTED
211
212 :*****
213 :AN ERROR HAS BEEN DETECTED IN TRYING TO READ THE BAD SECTOR FILE.
214 :THE SECTOR WILL BE RETRIED IF POSSIBLE.
215
216 033412 005337 034500 42$: DEC 74$ :YES, DECREMENT RETRY COUNT AND
217 033416 001057 BNE 48$ :RETRY IF COUNT IS POSITIVE
218
219 :THE RETRY COUNT HAS EXPIRED - SEE IF THE ERROR IS MEDIA RELATED
220
221 033420 032737 100720 001352 BIT #DCK!HCRC!HCE!FER!ECH,RMER1I :ANY MEDIA RELATED ERRORS ?
222 033426 001004 BNE 43$ :YES, GO TRY NEXT AVAILABLE SECTOR
223
224 033430 032737 100040 001400 BIT #BSE!SSE,RMER2I :ANY MEDIA RELATED ERRORS ?
225 033436 001453 BEQ 49$ :NO, EXIT AND REPORT ERROR ON RETURN
226
227 :THE ERRORS DETECTED WHILE TRYING TO RECOVER THE BAD SECTOR FILE ARE
    
```

```

228                                     :DUE TO THE MEDIA. SEE IF THE BAD SECTOR FILE CAN BE RECOVERED FROM
229                                     :ANOTHER AREA ON THE LAST TRACK
230
231 033440 032737 000040 001400 43$: BIT #SSE,RMER2I ;SSE ERROR ?
232 033446 001406 BEQ 44$ ;NO !!
233 033450 005237 001420 INC RMDAO ;DISPLACE SECTOR ADDRESS BY 1 TO
234 ;ACCOUNT FOR THE SKIP SECTOR ERROR
235 033454 052737 001000 001444 BIS #SSEI,RMOFO ;AND SET SSEI IN OFFSET REGISTER.
236 033462 000432 BR 47$ ;TRY AGAIN
237
238 033464 062737 000002 001420 44$: ADD #2,RMDAO ;ADVANCE SECTOR ADDRESS BY 2
239 033472 022737 001056 001446 CMP #558.,RMDCO ;READING DEC 144 BAD SECTOR FILE ?
240 033500 001407 BEQ 45$ ;YES !!
241 033502 005237 001420 INC RMDAO ;IF READING SSF, THEN ADVANCE SECTOR ADDRESS BY 1 MORE
242 033506 122737 000017 001420 CMPB #15.,RMDAO ;QUIT IF ALL SSF SECTORS HAVE BEEN
243 033514 101424 BLOS 49$ ;TRIED.
244 033516 000414 BR 47$
245
246 033520 022737 103726 001416 45$: CMP #MFGFIL,RMBAO ;READING MANUFACTURES BAD FILE ?
247 033526 001004 BNE 46$ ;NO !!
248 033530 122737 000012 001420 CMPB #10.,RMDAO ;QUIT IF ALL MFG SECTORS HAVE BEEN
249 033536 101413 BLOS 49$ ;TRIED.
250 033540 122737 000040 001420 46$: CMPB #32.,RMDAO ;QUIT IF ALL USER SECTORS HAVE BEEN
251 033546 101407 BLOS 49$ ;TRIED.
252
253 033550 012737 000003 034500 47$: MOV #3,74$ ;REINSTATE RETRY COUNT FOR THIS SECTOR
254 033556 162716 000006 48$: SUB #6,(SP) ;MOVE SP BACK TO NO ERROR
255 033562 000137 032566 JMP 3$ ;RETRY THE READ OPERATION
256
257                                     :THE BAD SECTOR FILE CANNOT BE READ
258 49$: NOP
259 033566 000240 BIT #SW13,@SWR ;INHIBIT MESSAGE ?
260 033570 032777 020000 145356 BNE 50$ ;YES
261 033576 001002 SUB #4,(SP) ;MOVE SP TO ERROR RETURN
262 033600 162716 000004 50$: JMP 73$ ;GO TO MODULE EXIT
263 033604 000137 034474
264
265                                     :THE SECTOR WAS RECOVERED WITHOUT ERROR - READ THE MFG/USER FILE IF
266                                     :THIS IS THE SSF FILE OR ELSE DONE.
267 51$: CMP #558.,RMDCO ;READING DEC 144 FILES ?
268 033610 022737 001056 001446 BEQ 52$ ;YES !!
269 033616 001422 JSR PC,GETSSF ;GET SKIP SECTOR FILE FOR USE
270 033620 004737 034514
271
272 033624 012737 001056 001446 MOV #558.,RMDCO ;READ DEC 144 FILE, CYLINDER 558.
273 033632 013737 001334 001420 MOV LSTRK,RMDAO ;MFG FILE LAST TRACK, SECTOR = 0
274 033640 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
275 033646 012737 103726 001416 MOV #MFGFIL,RMBAO ;POINT TO MANUFACTURES FILE BUFFER
276 033654 012737 177376 001414 MOV #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
277 033662 000415 BR 53$ ;GO READ MFG FILE
278 033664 52$: CMP #USRFIL,RMBAO ;WAS THE USER FILE READ ??
279 033664 022737 104734 001416 BEQ 54$ ;YES - READ IS COMPLETE
280 033672 001416 MOVB #10.,RMDAO ;READ THE USER FILE LAST TRACK, SECTOR = 10.
281 033674 112737 000012 001420 MOV #USRFIL,RMBAO ;POINT TO USERS FILE BUFFER
282 033702 012737 104734 001416 MOV #FMT16,RMOFO ;16 BIT FORMAT
283 033710 012737 010000 001444
284
    
```



```

338 034116 005037 001514          CLR      SSFENB          ;ASSUME 'SSF' BIT IS NOT TO BE WRITTEN
339                                ;IN THE 1ST HEADER WORD.
340 034122 004737 034132          JSR      PC,60$         ;SEARCH SKIP SECTOR MAPPED FILE
341 034126 000137 034452          JMP      71$           ;DONE WITH ALL FILE SEARCHES !!
342
343 034132 013737 001516 034504 60$:  MOV      ASNDC,76$      ;LOAD COMPARING CYLINDER ADDRESS
344 034140 013737 001520 034506      MOV      ASNDA,77$      ;LOAD COMPARING TRACK, SECTOR ADDRESS
345 034146 013737 034500 034502      MOV      74$,75$       ;LOAD NUMBER OF SECTORS TO CONFIRM
346
347                                ;SETUP FOR A BINARY SEARCH OF THE CURRENT FILE FOR THE COMPARING
348                                ;CYLINDER, TRACK AND SECTOR ADDRESS
349
350 034154 013700 034510          61$:  MOV      78$,R0      ;LOAD THE BASE ADDRESS IN R0
351 034160 022710 177777          62$:  CMP      #-1,(R0)   ;IS THIS FILE TERMINATOR ?
352 034164 001462                    BEQ      67$           ;BR IF YES
353 034166 023710 034504          CMP      76$,(R0)     ;IS THIS CYLINDER IN BAD TABLE ?
354 034172 001014                    BNE      63$           ;BR IF NO
355
356                                ;FILE ENTRY EQUALS COMPARING CYLINDER.  SEE IF THE NEXT ENTRY EQUALS
357                                ;THE COMPARING TRACK, AND SECTOR.
358
359 034174 123760 034507 000003      CMPB     77$+1,3(R0)   ;IS THIS TRACK IN BAD TABLE ?
360 034202 001010                    BNE      63$           ;BR IF NO
361
362 034204 123760 034506 000002      CMPB     77$,2(R0)    ;IS THIS SECTOR IN BAD TABLE ?
363 034212 001406                    BEQ      64$           ;BR IF YES
364 034214 002403                    BLT      63$           ;BR IF SECTOR IS LESS THAN TABLE ENTRY, ELSE
365 034216 012737 177777 001514      MOV      #-1,SSFENB   ;ENABLE 'SSF' BIT TO BE WRITTEN IN THE
366                                ;1ST HEADER WORD.
367
368 034224 022020          63$:  CMP      (R0)+,(R0)+ ;NO, ADJUST CYLINDER POINTER IN BAD FILE
369 034226 000754          BR       62$           ;AND CONTINUE SEARCH.
370
371                                ;THE COMPARING CYLINDER, TRACK AND SECTOR IS IN THE BAD SECTOR FILE.
372                                ;ADVANCE THE ASSIGNED SECTOR AND START THE SEARCH ALL OVER.
373
374 034230          64$:  INCB     ASNDA      ;INCREMENT SECTOR
375 034230 105237 001520          BIT      #SSEI,RMOFO  ;IS SSEI SET ?
376 034234 032737 001000 001444      BEQ      65$           ;NO !!
377 034242 001405                    CMPB     #31,,ASNDA    ;SECTOR OK ?
378 034244 122737 000037 001520      BHIS    60$           ;YES !!
379 034252 103327                    BR       66$
380 034254 000404          65$:  CMPB     #30,,ASNDA  ;SECTOR OK ??
381 034256 122737 000036 001520      BHIS    60$           ;YES !!
382 034264 103322          66$:  CLRB     ASNDA      ;CLEAR SECTOR AND ADVANCE TRACK
383 034266 105037 001520          INCB     ASNDA+1
384 034272 105237 001521          INCB     ASNDA+1
385 034276 123737 001335 001521      CMPB     LSTRK+1,ASNDA+1 ;TRACK OK ?
386 034304 103312                    BHIS    60$           ;YES !!
387 034306 005037 001520          CLR      ASNDA        ;CLEAR TRACK AND SECTOR
388 034312 005237 001516          INC      ASNDC         ;INCREMENT CYLINDER
389 034316 022737 001060 001516      CMP      #560,,ASNDC  ;CYLINDER OK ??
390 034324 103302                    BHIS    60$           ;YES !!
391 034326 005726          TST      (SP)+        ;RESTORE STACK AND
392 034330 000450          BR       71$         ;GET OUT !!
393
394                                ;THE COMPARING SECTOR IS NOT IN THE BAD SECTOR FILES.  DECREMENT THE
                                ;NUMBER OF SECTORS TO COMPARE AND SEARCH THE NEXT SECTOR IF THE NUMBER
    
```

```

395                                     :IS NOT ZERO.
396 034332                             67$:
397 034332 005337 034502               DEC 75$      ;DECREMENT NUMBER OF SECTORS TO COMPARE
398 034336 001456                       BEQ 73$      ;DONE IF ZERO
399
400 034340 105237 034506                 INCB 77$      ;INCREMENT THE COMPARING SECTOR
401 034344 032737 001000 001444         BIT #SSEI,RMOFO ;IS SSEI SET ?
402 034352 001405                       BEQ 68$      ;NO !!
403 034354 122737 000037 034506         CMPB #31.,77$  ;SECTOR OK ?
404 034362 103032                       BHIS 70$     ;YES !!
405 034364 000404                       BR 69$
406 034366 122737 000036 034506 68$:   CMPB #30.,77$  ;SECTOR OK ??
407 034374 103025                       BHIS 70$     ;YES !!
408 034376 105037 034506                 CLRB 77$     ;CLEAR SECTOR
409 034402 105237 034507                 INCB 77$+1   ;INCREMENT TRACK
410 034406 123737 001335 034507         CMPB LSTRK+1,77$+1 ;TRACK OK ??
411 034414 103015                       BHIS 70$     ;YES !!
412 034416 005037 034506                 CLR 77$     ;CLEAR SECTOR TRACK
413 034422 005237 034504                 INC 76$     ;INCREMENT CYLINDER
414 034426 022737 001060 034504         CMP #560.,76$ ;CYLINDER OK ??
415 034434 103005                       BHIS 70$     ;YES !!
416 034436 013737 034504 001446         MOV 76$,RMDCO ;LOAD CYLINDER
417 034444 005726                       TST (SP)+    ;RESTORE STACK AND
418 034446 000407                       BR 72$     ;GET OUT !!
419
420 034450 000641                         70$: BR 61$      ;SEARCH NEXT SECTOR
421
422                                     ;ASSIGN THE SECTOR AND RETURN TO USER
423 034452                             71$:
424 034452 013737 001516 001446         MOV ASNDC,RMDCO ;LOAD CYLINDER
425 034460 013737 001520 001420         MOV ASNDA,RMDAO ;LOAD TRACK AND SECTOR
426 034466                             72$:
427 034466 012602                       MOV (SP)+,R2  ;;POP STACK INTO R2
428 034470 012601                       MOV (SP)+,R1  ;;POP STACK INTO R1
429 034472 012600                       MOV (SP)+,R0  ;;POP STACK INTO R0
430
431                                     ;THE FOLLOWING ARE STORAGE LOCATIONS FOR THE MODULE
432
433 034500 000000                         74$: .WORD 0   ;RETRY COUNT/ NUMBER OF SECTORS REQUIRED
434 034502 000000                         75$: .WORD 0   ;NUMBER OF SECTORS TO COMPARE
435 034504 000000                         76$: .WORD 0   ;COMPARING CYLINDER
436 034506 000000                         77$: .WORD 0   ;COMPARING TRACK AND SECTOR
437 034510 000000                         78$: .WORD 0   ;BASE ADDRESS OF BAD SECTOR FILE BEING SEARCHED
438 034512 000000                         79$: .WORD 0   ;CONTAINS RMER2
439
440                                     ;THIS ROUTINE SEARCHES THE SKIP SECTOR FILE FOR ANY BAD SECTORS
441                                     ;WHICH OCCUR ON THE FE CYLINDERS. THE BAD SECTORS ARE THEN MAPPED
442                                     ;INTO A 'SSFIL' TABLE TO BE USED BY THE 'BADSECT' ROUTINE.
443
444 034514                             GETSSF:
445 034514 010046                       MOV R0,-(SP)  ;;PUSH R0 ON STACK
446 034516 010146                       MOV R1,-(SP)  ;;PUSH R1 ON STACK
447 034520 012700 102722                 MOV #BUFTWO,R0 ;R0 = POINTER TO BUFFER
448 034524 012701 105742                 MOV #SSFIL,R1 ;R1 = POINTER TO SKIP SECTOR TABLE
    
```

```

447 034530 012021      MOV      (R0)+,(R1)+      ;SAVE HEADER WORD 0(CYL ADDRESS)
448 034532 012021      MOV      (R0)+,(R1)+      ;SAVE HEADER WORD 1(TRK/SEC ADDRESS)
449
450 034534 012721 177777      1$:      MOV      #-1,(R1)+      ;INITIALIZE 'SSFIL' TABLE
451 034540 022701 106036      CMP      #SSFIL+60.,R1    ;DONE ?
452 034544 001373      BNE      1$              ;BR IF NO
453
454 034546 012701 105746      MOV      #SSFIL+4,R1      ;SETUP TO MAP SSFIL TABLE
455 034552 062700 000026      ADD      #22.,R0          ;GET OVER S/N(2), CREATE DATE, REV DATE, REV#,
456                                     ;FORMAT TYPE, UNUSED WORDS(3), ENTRY# AND
457                                     ;CHECKSUM (11. WORDS)
458 034556 021027 001057      2$:      CMP      (R0),#559.      ;IS CYLINDER IN FE AREA ?
459 034562 103410      BLO      4$              ;NO !!
460 034564 101004      BHI      3$              ;YES, ON CYLINDER 560.
461                                     ;YES, WHEN ON CYLINDER 559. DO NOT LOG
462                                     ;ANY BAD SECTORS THAT OCCUR ON THE SKIP
463                                     ;SECTOR TRACK(0) OR ALTERNATE DEC144 TRACK(1)
464 034566 126027 000003 000001      CMPB     3(R0),#1        ;IS IT ON TRACK 0 OR 1 ?
465 034574 003403      BLE      4$              ;YES !!
466 034576 011021      3$:      MOV      (R0),(R1)+      ;LOG THIS CYLINDER AND
467 034600 016021 000002      MOV      2(R0),(R1)+      ;TRACK/SECTOR IN 'SSFIL' TABLE.
468
469 034604 022020      4$:      CMP      (R0)+,(R0)+      ;ADJUST BUFFER TO NEXT CYLINDER ENTRY
470 034606 020127 106036      CMP      R1,#SSFIL+60.    ;ANY MORE ENTRIES AVAILABLE IN TABLE ?
471 034612 103003      BHIS     5$              ;NO, EXIT !!
472 034614 020027 105736      CMP      R0,#BUFTWO+1548. ;DONE WITH ALL 3 SECTORS ?
473 034620 103756      BLO      2$              ;NO, DO NEXT ENTRY
474
475 034622 012737 177777 104732 5$:      MOV      #-1,MFGFIL+516. ;RESTORE TERMINATOR TO MFGFIL
476 034630 000240      NOP
477 034632 012601      MOV      (SP)+,R1        ;;POP STACK INTO R1
478 034634 012600      MOV      (SP)+,R0        ;;POP STACK INTO R0
478 034636 000207      RTS      PC
    
```

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

.SBTTL BUFFER GENERATOR SUBROUTINE

:THIS SUBROUTINE GENERATES A DATA BUFFER FOR WRITE COMMANDS. THE
 :BUFFER STARTS AT RMBA AND IS RMWC WORDS LONG. THE BUFFER
 :CONTAINS A REPETITIVE DATA PATTERN CONSISTING OF \$TMP1 WORDS
 :FROM THE DATA PATTERN TABLE, BEGINNING AT ADDRESS \$TMP0.
 :HEADER INFORMATION FOR THE BUFFER IS EXTRACTED FROM RMDC,
 :RMDA AND RMOF.

:R0 = ADDRESS OF DATA BUFFER
 :R1 = LENGTH OF DATA BUFFER
 :R2 = ADDRESS OF DATA PATTERN
 :R3 = LENGTH OF DATA PATTERN
 :R4 = SECTOR COUNT

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

GENBUF:

```

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 RMBAO,R0      ;LOAD DATA BUFFER ADDRESS
MOV RMWCO,R1      ;LOAD WORD COUNT
MOV RMDCO,7$     ;LOAD STARTING CYLINDER ADDRESS
MOV RMDAO,8$     ;LOAD STARTING TRACK,SECTOR ADDRESS

26 034676 032737 000002 001412 1$: BIT #BIT1,RMCS10 ;WRITE HEADER & DATA??
27 034704 001451          BEQ 3$          ;NO!!
28 034706 013710 035102  MOV 7$(R0) ;WRITE HEADER WORD #1
29 034712 052710 140000  BIS #MSE!USE,(R0) ;SET BAD SECTOR FLAGS FOR GOOD SECTOR
30 034716 012702 000035  MOV #29.,R2 ;R2 = MAXIMUM SECTOR ADDRESS (29.)

32 034722 032737 010000 001444  BIT #FMT16,RMOFO ;18 BIT FORMAT??
33 034730 001410          BEQ 2$          ;YES !!
34 034732 052710 010000  BIS #FMT16,(R0) ;SET 16 FORMAT BIT IN HEADER
35 034736 005202          INC R2          ;CHANGE MAXIMUM SECTOR ADDRESS (30.)

37 034740 032737 001000 001444  BIT #SSEI,RMOFO ;IS SSEI SET TO ACCESS ALL SECTORS ?
38 034746 001401          BEQ 2$          ;NO !!
39 034750 005202          INC R2          ;CHANGE MAXIMUM SECTOR ADDRESS (31.)

41 034752 005201          2$: INC R1          ;INCREMENT WORD COUNT
42 034754 001443          BEQ 6$          ;EXIT IF DONE

44 034756 005720          TST (R0)+      ;MOVE R0 TO HEADER WORD #2
45 034760 013720 035104  MOV 8$(R0)+   ;WRITE HEADER WORD #2
46 034764 005201          INC R1          ;INCREMENT WORD COUNT AND
47 034766 001436          BEQ 6$          ;EXIT IF DONE
48 034770 012703 035104  MOV #8$,R3    ;ADVANCE SECTOR ADDRESS
49 034774 105213          INCB (R3)
50 034776 120213          CMPB R2,(R3)   ;SECTOR OVERFLOW ??
51 035000 103013          BHIS 3$       ;NO !!
52 035002 105013          CLRB (R3)   ;YES - CLEAR SECTOR ADDRESS
    
```

53	035004	105263	000001		INCB	1(R3)	:ADVANCE TRACK ADDRESS
54	035010	123763	001335	000001	CMPB	LSTRK+1,1(R3)	:TRACK OVERFLOW ??
55	035016	103004			BHIS	3\$:NO !!
56	035020	105063	000001		CLRB	1(R3)	:YES - CLEAR TRACK ADDRESS
57	035024	105237	035102		INCB	7\$:ADVANCE CYLINDER ADDRESS
58	035030	012704	000400	3\$:	MOV	#256,R4	:LOAD SECTOR DATA COUNT
59	035034	013702	001174	4\$:	MOV	\$TMP0,R2	:LOAD PATTERN ADDRESS
60	035040	013703	001176		MOV	\$TMP1,R3	:LOAD PATTERN COUNT
61	035044	012220		5\$:	MOV	(R2)+,(R0)+	:WRITE DATA PATTERN
62	035046	005201			INC	R1	:INCREMENT WORD COUNT AND
63	035050	001405			BEQ	6\$:EXIT IF DONE
64	035052	005304			DEC	R4	:DECREMENT SECTOR COUNT
65	035054	001710			BEQ	1\$:START NEXT SECTOR IF 0
66	035056	005303			DEC	R3	:DECREMENT PATTERN COUNT
67	035060	001765			BEQ	4\$:RESTART PATTERN IF 0
68	035062	000770			BR	5\$:CONTINUE DATA PATTERN
69	035064			6\$:	MOV	(SP)+,R4	::POP STACK INTO R4
	035066	012604			MOV	(SP)+,R3	::POP STACK INTO R3
	035070	012603			MOV	(SP)+,R2	::POP STACK INTO R2
	035072	012602			MOV	(SP)+,R1	::POP STACK INTO R1
	035074	012601			MOV	(SP)+,R0	::POP STACK INTO R0
70	035076	000240			NOP		
71	035100	000207			RTS	PC	
72							
73	035102	000000		7\$:	.WORD		:CYLINDER ADDRESS STORAGE
74	035104	000000		8\$:	.WORD		:TRACK, SECTOR ADDRESS STORAGE

```

1      .SBTTL COMPARE BUFFER SUBROUTINE
2
3      :THIS SUBROUTINE COMPARES THE CONTENTS OF BUFONE AND BUFTWO,
4      :ASSUMING THAT BUFONE IS THE BUFFER FROM WHICH DATA WAS WRITTEN
5      :AND BUFTWO IS THE BUFFER TO WHICH DATA WAS READ. ERRORS IN BUFFER
6      :COMPARISON ARE REPORTED TO THE USER VIA THE USER'S ERROR CALL.
7
8      :CALL:
9      : (1) JSR PC,CMPBUF
10     : (2) .WORD WRITE BUFFER ADDRESS
11     :      .WORD READ BUFFER ADDRESS
12     : (3) BR ?? RETURN HERE IF NO ERROR
13     : (4) NOP RETURN HERE IF ERROR
14     : (5) ERROR ERROR DEFINED BY SUBROUTINE
15     : (6) ???
16
17     CMPBUF:
18     MOV R0,-(SP) ;;PUSH R0 ON STACK
19     MOV R1,-(SP) ;;PUSH R1 ON STACK
20     MOV R2,-(SP) ;;PUSH R2 ON STACK
21     MOV R3,-(SP) ;;PUSH R3 ON STACK
22     CLR 14$ ;;CLEAR CORRECTION FLAG
23
24     :DETERMINE IF DATA SHOULD BE CORRECTED
25     BIT ECI,RMOFI ;:WAS ECC CORRECTION ALLOWED ??
26     BNE 9$ ;:NO !!
27     BIT #DCK,RMER1 ;:WAS THERE A DATA CHECK ??
28     BEQ 9$ ;:NO !!
29     BIT #ECH,RMER1 ;:IS ERROR CORRECTION HARD SET ?
30     BNE 9$ ;:YES !!
31     BIT #FMT16,RMOFI ;:IS THIS 16 BIT FORMAT ??
32     BEQ 9$ ;:NO !!
33
34     :CORRECT DATA USING ECC INFORMATION
35     MOV RMBAD,R0 ;:R0 = STARTING BUFFER ADDRESS
36     MOV RMEC11,R1 ;:R1 = ECC POSITION
37     BIS #BIT15,14$ ;:SET CORRECTION FLAG
38
39     :MOVE R0 TO WORD BOUNDARY OF ERROR BURST
40     1$: CMP #16.,R1 ;:IS BIT POSITION > 1 WORD
41     BHIS 2$ ;:NO !!
42     SUB #16.,R1 ;:SUBTRACT 1 WORDS WORTH
43     TST (R0)+ ;:ADVANCE BUFFER ADDRESS 1 WORD
44     BR 1$
45     2$: MOV #1,R2 ;:R2 = BIT POINTER
46     MOV R2,R3 ;:R3 = BIT NUMBER
47
48     :MOVE R2 TO STARTING BIT OF ERROR BURST
49     3$: CMP R3,R1 ;:IS R3 SAME AS R1 ??
50     BEQ 4$ ;:YES !!
51     ASL R2 ;:SHIFT BIT POINTER
52     INC R3 ;:INCREMENT BIT NUMBER
53     BR 3$
54     4$: MOV #11.,R3 ;:R3 = LENGTH OF ERROR BURST
55
56     :CORRECT THE ERROR BURST
57     5$: BIT R2,RMEC21 ;:IS THIS BIT SET IN ECC PATTERN ??
    
```

```

17 035106
18 035106 010046
19 035110 010146
20 035112 010246
21 035114 010346
22 035116 005037 035446
23 035122 033737 004000 001370
24 035130 001063
25 035132 032737 100000 001352
26 035140 001457
27 035142 032737 000100 001352
28 035150 001053
29 035152 032737 010000 001370
30 035160 001447
31 035162 013700 001416
32 035166 013701 001402
33 035172 052737 100000 035446
34
35 035200 022701 000020
36 035204 103004
37 035206 162701 000020
38 035212 005720
39 035214 000771
40 035216 012702 000001
41 035222 010203
42
43
44
45 035224 020301
46 035226 001403
47 035230 006302
48 035232 005203
49 035234 000773
50 035236 012703 000013
51
52
53 035242 030237 001404
    
```

```

54 035246 001405          BEQ      7$          :NO - DO NOT CORRECT THIS BIT
55 035250 030210          BIT      R2,(R0)     :IS THE BIT PRESENTLY SET ??
56 035252 001402          BEQ      6$          :NO
57 035254 040210          BIC      R2,(R0)     :RESET THE BIT
58 035256 000401          BR       7$
59 035260 050210          6$:    BIS      R2,(R0)     :SET THE BIT
60 035262 006302          7$:    ASL      R2          :SHIFT TO NEXT BIT
61 035264 001003          BNE      8$
62 035266 012702 000001    MOV      #1,R2      :CONTINUE WITH FIRST BIT OF NEXT WORD
63 035272 005720          TST      (R0)+
64 035274 005303          8$:    DEC      R3          :END OF BURST ??
65 035276 001361          BNE      5$          :NO !!
66
67          :COMPARE WRITE BUFFER TO READ BUFFER
68 035300 017600 000010    9$:    MOV      @10(SP),R0 :R0 = WRITE BUFFER
69 035304 062766 000002 000010    ADD      #2,10(SP) :MOVE SP TO READ ADDRESS
70 035312 017601 000010          MOV      @10(SP),R1 :R1 = READ BUFFER
71 035316 062766 000002 000010    ADD      #2,10(SP) :MOVE SP TO RETURN ADDRESS
72 035324 013702 001340          MOV      RMWCI,R2   :R2 = NUMBER OF WORDS TRANSFER
73 035330 163702 001414          SUB      RMWCO,R2
74 035334 022021          10$:   CMP      (R0)+,(R1)+ :COMPARE DATA WORDS
75 035336 001003          BNE      11$        :EXIT IF NOT EQUAL
76 035340 005302          DEC      R2          :DECREMENT WORD COUNT
77 035342 001374          BNE      10$        :CONTINUE IF NOT DONE
78 035344 000433          BR       13$        :DONE COMPARE - NO ERROR
79
80          :DATA COMPARE FAILED
81 035346 014037 001140    11$:   MOV      -(R0),%GDADR :STORE GOOD DATA FOR TYPEOUT
82 035352 014137 001142          MOV      -(R1),%BDADR :STORE BAD DATA FOR TYPEOUT
83 035356 010037 001134          MOV      R0,%GDADR   :STORE ADDRESS OF GOOD DATA
84 035362 010137 001136          MOV      R1,%BDADR   :STORE ADDRESS OF BAD DATA
85 035366 010237 001174          MOV      R2,%STMP0   :STORE WORD COUNT OF ERROR
86 035372 062766 000004 000010    ADD      #4,10(SP)   :MOVE SP TO USER'S ERROR CALL
87 035400 112776 000336 000010    MOV      #336,@10(SP) :WRITE ERROR NUMBER IN CALL
88
89          :CHANGE ERROR NUMBER IF ECC CORRECTION FAILED
90 035406 032737 100000 035446    BIT      #BIT15,14$ :WAS ECC CORRECTION USED ??
91 035414 001403          BEQ      12$        :NO !!
92 035416 112776 000163 000010    MOV      #163,@10(SP) :ECC CORRECTION FAILED
93 035424 162766 000002 000010    12$:   SUB      #2,10(SP) :MOVE SP TO RETURN IF ERROR
94 035432 000240          NOP
95 035434          13$:
96 035434 012603          MOV      (SP)+,R3   ::POP STACK INTO R3
97 035436 012602          MOV      (SP)+,R2   ::POP STACK INTO R2
98 035440 012601          MOV      (SP)+,R1   ::POP STACK INTO R1
99 035442 012600          MOV      (SP)+,R0   ::POP STACK INTO R0
100 035444 000207          RTS      PC         :RETURN TO USER
101 035446 000000          14$:   .WORD          :ECC CORRECTION FLAG
    
```



```

1
2
3
4
5
6
7
8
9
10 035450
    035450 010046
11 035452 010146
12 035454 010246
13 035456 012700 001526
14 035462 012701 001406
15 035466 012702 000046
16 035472 110220
17 035474 005041
18 035476 162702 000002
19 035502 100405
20 035504 022702 000022
21 035510 001370
22 035512 005041
23 035514 000770
24 035516 112720 000200
25 035522 012602
26 035524 012601
27 035526 012600
28 035530 000240
29 035532 000207

.SBTTL GET STATUS SUBROUTINE
:THIS SUBROUTINE SETS UP THE "GET INDEX TABLE" AND THE "GET
:BUFFER" FOR READING ALL SUBSYSTEM REGISTERS VIA THE GET SUBROUTINE
:AND THEN RETURNS TO THE USER.
:CALL: JSR      PC,GETSTS
:      ???
:      RETURN HERE

GETSTS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      #GETINX,R0    ;;R0 = ADDRESS OF INDEX TABLE
MOV      #RMEC2I+2,R1  ;;R1 = ADDRESS OF GET BUFFER
MOV      #RMEC2,R2     ;;R2 = REGISTER INDE
1$:      MOVB      R2,(R0)+  ;;WRITE REGISTER INDEX IN TABLE
        CLR      -(R1)    ;;CLEAR CORRESPONDING LOCATION
2$:      SUB      #2,R2    ;;DECREMENT TO NEXT INDEX
        BMI      3$      ;;BRANCH OUT IF DONE
        CMP      #RMDB,R2 ;;DONT WRITE RMDB INDEX
        BNE      1$
        CLR      -(R1)
        BR      2$
3$:      MOVB      #200,(R0)+  ;;WRITE TERMINATOR
        MOV      (SP)+,R2    ;;POP STACK INTO R2
        MOV      (SP)+,R1    ;;POP STACK INTO R1
        MOV      (SP)+,R0    ;;POP STACK INTO R0
        NOP
        RTS      PC
  
```

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

.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 "RMBA" 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
 ???

:R0 = REGISTER BASE ADDRESS
 :R1 = REGISTER ADDRESS
 :R2 = BUFFER BASE ADDRESS
 :R3 = BUFFER ADDRESS
 :R4 = POINTER TO REGISTER INDEX

```

GET:  NOP
      ADD    #4,(SP)           ;CLEAR ERROR NUMBER IN USER'S
      CLRB  @ (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    $BASE,R0
      MOV    #GETBUF,R2
      MOV    #GETINX,R4
      MOV    #3$,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    2$              ;YES!!
      ADD    #4,16(SP)        ;WRITE ERROR NUMBER IN USER'S
      MOVB  #112,@16(SP)     ;ERROR CALL
      BR    4$
2$:   TSTB  (R4)              ;DONE??
      BMI   6$              ;YES!!
      MOVB  (R4),R1          ;R1 = REGISTER ADDRESS
      BIC   #^CIDXMSK,R1     ;CLEAR ANY SIGN EXTENSION
    
```

```

31 035534 000240
32 035536 062716 000004
33 035542 105076 00C000
34 035546 162716 000004
35 035552 010046
   035554 010146
   035556 010246
   035560 010346
   035562 010446
   035564 013746 000004
   035570 013746 000006
36 035574 013700 001276
37 035600 012702 001336
38 035604 012704 001526
39 035610 012737 035716 000004
40 035616 012737 000300 000006
41 035624 016037 000010 001174 1$:
42 035632 016037 000000 001176
43 035640 032737 004000 001176
44 035646 001007
45 035650 062766 000004 000016
46 035656 112776 000112 000016
47 035664 000423
48 035666 105714 2$:
49 035670 100433
50 035672 111401
51 035674 042701 177700
    
```

```

52 035700 060001          ADD      R0,R1
53 035702 112403          MOVVB   (R4)+,R3      ;R3 = STORAGE ADDRESS FOR REGISTER
54 035704 042703 177700  BIC     #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
55 035710 060203          ADD     R2,R3
56 035712 011113          MOV     (R1),(R3)   ;READ REGISTER
57 035714 000764          BR     2$
58
59 035716 022626          3$:    CMP     (SP)+,(SP)+ ;RESTORE STACK
60 035720 062766 000004 000016  ADD     #4,16(SP)    ;WRITE ERROR NUMBER IN
61 035726 112776 000007 000016  MOVVB  #7,@16(SP)   ;USER'S ERROR CALL
62 035734 162766 000002 000016  4$:    SUB     #2,16(SP)
63 035742 105714          5$:    TSTB   (R4)       ;DONE CLEARING??
64 035744 100405          BMI     6$         ;YES!!
65 035746 005003          CLR     R3         ;CLEAR REMAINING STORAGE
66 035750 112403          MOVVB  (R4)+,R3    ;LOCATIONS
67 035752 060203          ADD     R2,R3
68 035754 005013          CLR     (R3)
69 035756 000771          BR     5$
70 035760          6$:
70 035760 012637 000006  MOV     (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
70 035764 012637 000004  MOV     'SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
70 035770 012604          MOV     (SP)+,R4     ;;POP STACK INTO R4
70 035772 012603          MOV     (SP)+,R3     ;;POP STACK INTO R3
70 035774 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
70 035776 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
70 036000 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
71 036002 000207          RTS     PC           ;RETURN
    
```

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 036004 000240
29 036006 010046
036010 010146
036012 010246
036014 010346
036016 010446
036020 013746 000004
036024 013746 000006
30 036030 013700 001276
31 036034 012702 001412
32 036040 012704 001555
33 036044 012737 036164 000004
34 036052 012737 000300 000006
35 036060 016037 000010 001174
36 036066 016037 000000 001176
37 036074 032737 004000 001176
38 036102 001007
39 036104 062766 000004 000016
40 036112 112776 000112 000016
41 036120 000430
42 036122 105714
43 036124 100431
44 036126 111401
45 036130 042701 177700
46 036134 060001
47 036136 111403
48 036140 042703 177700
49 036144 060203
50 036146 011311
51 036150 122714 000032

.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
???

:R0 = REGISTER BASE ADDRESS
 :R1 = REGISTER ADDRESS
 :R2 = BUFFER BASE ADDRESS
 :R3 = BUFFER ADDRESS
 :R4 = POINTER TO REGISTER INDEX

```

PUT:  NOP
      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 $BASE,R0
      MOV #PUTBUF,R2
      MOV #PUTINX,R4
      MOV #4$,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 2$            ;:YES!!
      ADD #4,16(SP)     ;:WRITE ERROR NUMBER IN
      MOVB #112,a16(SP) ;:USER'S ERROR CALL
2$:   BR 5$
      TSTB (R4)         ;:DONE??
      BMI 6$            ;:YES!!
      MOVB (R4),R1     ;:R1 = REGISTER ADDRESS
      BIC #^CIDXMSK,R1 ;:CLEAR ANY SIGN EXTENSION
      ADD R0,R1
      MOVB (R4),R3     ;:R3 = STORAGE ADDRESS
      BIC #^CIDXMSK,R3 ;:CLEAR ANY SIGN EXTENSION
      ADD R2,R3
      MOV (R3),(R1)    ;:WRITE REGISTER
      CMPB #RMOF,(R4) ;:WAS RMOF JUST LOADED ?
  
```

```
52 036154 001001      BNE      3$
53 036156 011311      MOV      (R3), (R1)
54
55 036160 105724      3$:     TSTB   (R4)+
56 036162 000757      BR      2$
57
58 036164 022626      4$:     CMP    (SP)+, (SP)+
59 036166 062766      ADD    #4, 16(SP)
60 036174 112776      MOVB   #7, @16(SP)
61 036202 162766      5$:     SUB    #2, 16(SP)
62
63 036210      6$:
   036210 012637 000006      MOV    (SP)+, ERRVEC+2
   036214 012637 000004      MOV    (SP)+, ERRVEC
   036220 012604      MOV    (SP)+, R4
   036222 012603      MOV    (SP)+, R3
   036224 012602      MOV    (SP)+, R2
   036226 012601      MOV    (SP)+, R1
   036230 012600      MOV    (SP)+, R0
64 036232 000207      RTS    PC

      ::POP STACK INTO ERRVEC+2
      ::POP STACK INTO ERRVEC
      ::POP STACK INTO R4
      ::POP STACK INTO R3
      ::POP STACK INTO R2
      ::POP STACK INTO R1
      ::POP STACK INTO R0
      :RETURN
```

```

1      .SBTTL  SIZE CLOCK SUBROUTINE
2
3      SIZCLK:
4      036234 013746 000004      MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
5      036240 013746 000006      MOV      ERRVEC+2,-(SP)    ;;PUSH ERRVEC+2 ON STACK
6      036244 012737 036302 000004      MOV      #1$,ERRVEC      ;;SET UP FOR BUS TIMEOUT
7      036252 012737 000300 000006      MOV      #PR6,ERRVEC+2
8      036260 012737 177546 001522      MOV      #177546,CLKADR   ;;LOAD ADDRESSES FOR KW11-L
9      036266 012737 000100 001524      MOV      #100,CLKVCT
10     036274 005777 143222      TST      @CLKADR         ;;TEST FOR KW11-L PRESENT
11     036300 000421      BR      3$              ;;YES - KW11-L IS PRESENT
12     036302 022626      1$:  CMP      (SP)+,(SP)+    ;;RESTORE SP
13     036304 012737 036334 000004      MOV      #2$,ERRVEC      ;;SET UP FOR BUS TIMEOUT
14     036312 012737 172540 001522      MOV      #172540,CLKADR  ;;LOAD ADDRESSES FOR KW11-P CLOCK
15     036320 012737 000104 001524      MOV      #104,CLKVCT
16     036326 005777 143170      TST      @CLKADR         ;;TEST FOR KW11-P PRESENT
17     036332 000404      BR      3$              ;;YES - KW11-P IS PRESENT
18     036334 022626      2$:  CMP      (SP)+,(SP)+    ;;RESTORE SP
19     036336 062766 000002 000004      ADD      #2,4(SP)        ;;MOVE RETURN TO ERROR
20     036344      3$:
21     036344 012637 000006      MOV      (SP)+,ERRVEC+2   ;;POP STACK INTO ERRVEC+2
22     036350 012637 000004      MOV      (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
23     036354 000207      RTS      PC              ;;RETURN TO USER
    
```

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

.SBTTL TIMEOUT SUBROUTINE

:THIS SUBROUTINE WAITS FOR RDY = 1 AND GO = 0 OR FOR A TIMEOUT
 :GREATER THAN APPROX. 500 MSEC AND THEN RETURNS.

:CALL: JSR PC,TIMOUT
 : ??? RETURN HERE

TIMOUT:

```

MOV R0,-(SP)      ;;PUSH R0 ON STACK
MOV R2,-(SP)      ;;PUSH R2 ON STACK
MOV ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
MOV #3$,ERRVEC    ;SETUP FOR BUS TIMEOUT - 04 TRAP
MOV #PR6,ERRVEC+2
MOV $BASE,R0      ;R0=BASE ADDRESS
MOV #30,R2        ;R2=NUMBER OF CYCLES
1$: JSR PC,$TIMER ;START CLOCK TIMER

2$: MOV RMCS1(R0),-(SP) ;GET STATUS
    BIC #^C<RDY!GO>,(SP)
    CMP #RDY,(SP)+    ;RDY=1,GO=0??
    BEQ 4$           ;YES!!
    BIT #BIT7,@CLKADR ;TIMER DONE??
    BEQ 2$           ;NO!!
    DEC R2           ;DEC NUMBER OF CYCLES
    BNE 1$          ;CONTINUE IF NOT DONE
    BR 4$           ;"RDY" DID NOT SET OR "GO" DID NOT RESET
                    ;WITHIN 500 MSEC AFTER THE COMMAND WAS ISSUED.

3$: CMP (SP)+,(SP)+ ;ADJUST STACK
    ADD #4,12(SP)   ;MOVE SP TO USER'S CALL
    MOVB #7,@12(SP) ;WRITE ERROR NUMBER
    SUB #2,12(SP)

4$: MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
    MOV (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
    MOV (SP)+,R2      ;;POP STACK INTO R2
    MOV (SP)+,R0      ;;POP STACK INTO R0
    RTS PC           ;RETURN TO USER
    
```

:THIS ROUTINE IS USED START THE KW11-L OR THE KW11-P CLOCK. THE CLOCK
 :IS STARTED TO RUN IN FLAG MODE. A CLOCK WILL BE 16.667MS FOR A 60 HZ CPU
 :AND 20MS FOR A 50 HZ CPU.

\$TIMER:

```

MOV R1,-(SP)      ;;PUSH R1 ON STACK
MOV CLKADR,R1     ;R1=CLOCK ADDRESS
CMP R1,#172540    ;KW11-P CLOCK??
BNE 1$           ;NO!!
MOV #1,2(R1)     ;SET COUNTER
1$: MOV #BIT2!BIT0,(R1) ;START COUNTER
    MOV (SP)+,R1   ;POP STACK INTO R1
    RTS PC        ;RETURN
    
```

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 036552
40
41
42 036552 062716 000004
43 036556 105076 000000
44 036562 162716 000004
45
46
47 036566 013737 001346 001142
48 036574 042737 177770 001142
49 036602 013737 001234 001140
50 036610 042737 177770 001140
51 036616 123737 001140 001142
52
53 036624 001415
54 036626 062716 000004
55 036632 112776 000001 000000
56 036640 162716 000002
57 036644 004736

.SBTTL PRIMARY ERROR CHECK SUBROUTINE

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

:CORRECT 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 RMDS) 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 RMDS 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
    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
```

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,$BDDAT ;CORRECT UNIT SELECTED??
BIC #^CUNTMSK,$BDDAT
MOV $UNIT,$GDDAT ;GOOD DATA FOR TYPEOUT
BIC #^CUNTMSK,$GDDAT
CMPB $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
;DRIVE NUMBERS
;YES!!
```

```
BEQ 1$
ADD #4,(SP)
MOVB #1,@(SP) ;ERROR 1
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;REPORT WRONG UNIT SELECTED
```



```

58 036646 162716 000010          SUB      #10,(SP)          ;RESTORE (SP)
59 036652 000240          NOP
60 036654 000137 037374          JMP      10$          ;SKIP OTHER CHECKS
61 036660
62
63
64
65 036660 032737 004000 001336          BIT      #DVA,RMCS1I      ;DEVICE AVAILABLE??
66 036666 001045          BNE      3$          ;YES!!
67 036670 013737 001336 001140          MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
68 036676 052737 004000 001140          BIS      #DVA,$GDDAT
69 036704 013737 001336 001142          MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
70 036712 062716 000004          ADD      #4,(SP)
71 036716 112776 000002 000000          MOV      #2,@(SP)        ;ERROR #2
72 036724 032737 010000 001346          BIT      #NED,RMCS2I      ;WAS NED SET??
73 036732 001414          BEQ      2$          ;NO!!
74 036734 013737 001346 001140          MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
75 036742 013737 001346 001142          MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
76 036750 042737 010000 001140          BIC      #NED,$GDDAT
77 036756 112776 000003 000000          MOV      #3,@(SP)        ;YES - CHANGE ERROR NUMBER
78 036764 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
79 036770 004736          JSR      PC,@(SP)+      ;REPORT DEVICE NOT AVAILABLE
80 036772 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
81 036776 000240          NOP
82 037000 000575          BR       10$          ;SKIP OTHER CHECKS
83 037002
84
85
86 037002 032737 000200 001336          ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
87 037010 001030          BIT      #RDY,RMCS1I      ;CONTROLLER READY??
88 037012 013737 001336 001140          BNE      4$          ;YES!!
89 037020 052737 000200 001140          MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
90 037026 042737 160001 001140          BIS      #RDY,$GDDAT
91 037034 013737 001336 001142          BIC      #SC!TRÉ!MCPE!GO,$GDDAT
92 037042 062716 000004          MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
93 037046 112776 000004 000000          ADD      #4,(SP)
94 037054 162716 000002          MOV      #4,@(SP)        ;ERROR #4
95 037060 004736          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
96 037062 162716 000010          JSR      PC,@(SP)+      ;REPORT CONTROLLER NOT READY
97 037066 000240          SUB      #10,(SP)        ;RESTORE (SP)
98 037070 000541          NOP
99 037072          BR       10$          ;SKIP OTHER CHECKS
100
101
102 037072 032737 000001 001336          ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
103 037100 001431          BIT      #GO,RMCS1I      ;GO RESET??
104 037102 032737 000200 001350          BEQ      5$          ;YES!!
105 037110 001025          BIT      #DRY,RMDSI      ;DRIVE READY??
106 037112 013737 001336 001140          BNE      5$          ;YES!!
107 037120 042737 160001 001140          MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
108 037126 013737 001336 001142          BIS      #SC!TRÉ!MCPE!GO,$GDDAT
109 037134 062716 000004          MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
110 037140 112776 000005 000000          ADD      #4,(SP)
111 037146 162716 000002          MOV      #5,@(SP)        ;ERROR #5
112 037152 004736          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
113 037154 162716 000010          JSR      PC,@(SP)+      ;REPORT DRIVE NOT READY
114 037160 000240          SUB      #10,(SP)        ;RESTORE (SP)
    
```

```

115 037162 000504          BR      10$
116 037164          5$:
117
118          :REPORT AN ERROR IF THE RH CONTROLLER DETECTED BAD
119          :PARITY ON THE MASSBUS CONTROL BUS
120 037164 032737 020000 001336  BIT      #MCPE,RMCS1I  :PARITY ERROR ??
121 037172 001425          BEQ      6$          :NO!!
122 037174 013737 001336 001140  MOV      RMCS1I,$GDDAT :EXPECTED STATUS
123 037202 042737 160001 001140  BIC      #SC!TRE!MCPE!GO,$GDDAT
124 037210 013737 001336 001142  MOV      RMCS1I,$BDDAT :RECEIVED STATUS
125 037216 062716 000004          ADD      #4,(SP)      :MOVE STACK TO USER'S ERROR
126 037222 112776 000013 000000  MOVB    #13,@(SP)    :ERROR #13
127 037230 162716 000002          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
128 037234 004736          JSR     PC,@(SP)+    :REPORT ERROR VIA USER
129 037236 162716 000010          SUB      #10,(SP)    :RESTORE STACK
130 037242 000240          NOP
131 037244 000453          BR      10$
132 037246          6$:
133
134          :REPORT AN ERROR IF THE RM80 DETECTED A CONTROL BUS PARITY ERROR
135 037246 032737 000010 001352  BIT      #PAR,RMER1I  :WAS THERE A PARITY ERROR??
136 037254 001451          BEQ      11$         :NO!!
137 037256 032737 000010 001400  BIT      #DPE,RMER2I  :WAS IT THE CONTROL BUS??
138 037264 001045          BNE     11$         :NOT SURE!!
139 037266 032737 000010 001426  BIT      #PAR,RMER10  :DID TEST SET PAR ??
140 037274 001413          BEQ      9$          :NO!!
141 037276 010046          MOV     R0,-(SP)     :PUSH R0 ON STACK
142 037300 012700 001555          MOV     #PUTINX,R0   :R0 POINTS TO INDEX TABLE
143 037304 122710 000014          7$:  CMPB   #RMER1,(R0)  :SEARCH TABLE FOR RMER1
144 037310 001002          BNE     8$          :
145 037312 012600          MOV     (SP)+,R0    :POP STACK INTO R0
146 037314 000431          BR     11$         :PAR WAS SET BY TEST
147 037316 105720          8$:  TSTB  (R0)+        :END OF TABLE??
148 037320 100371          BPL    7$          :NO!!
149 037322 012600          MOV     (SP)+,R0    :POP STACK INTO R0
150 037324 013737 001352 001140  9$:  MOV     RMER1I,$GDDAT :EXPECTED STATUS
151 037332 042737 000010 001140  BIC     #PAR,$GDDAT
152 037340 013737 001352 001142  MOV     RMER1I,$BDDAT :RECEIVED STATUS
153 037346 062716 000004          ADD     #4,(SP)     :MOVE SP TO USER'S ERROR CALL
154 037352 112776 000050 000000  MOVB   #50,@(SP)    :WRITE THE ERROR NUMBER
155 037360 162716 000002          SUB     #2,(SP)     :MOVE SP TO RETURN FOR ERROR
156 037364 004736          JSR    PC,@(SP)+   :REPORT THE ERROR
157 037366 162716 000010          SUB     #10,(SP)    :MOVE SP TO NO ERROR RETURN
158 037372 000240          NOP
159 037374 062716 000010          10$: ADD     #10,(SP)    :RETURN TO ERROR
160 037400 000240          11$: NOP
161 037402 000207          RTS     PC
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.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
      ERROR          RETURN HERE TO REPORT AN ERROR
      JSR    PC,@(SP)+   ERROR NUMBER DEFINED BY SUB
      ???          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    RMCS10,81$      :STORE FUNCTION CODE
BIC    #^C<F0!F1!F2!F3!F4>,81$
ADD    #4,(SP)        :MOVE (SP) TO ERROR CALL
CLRB   @(SP)          :CLEAR ERROR NUMBER
SUB    #4,(SP)        :MOVE (SP) TO NO ERROR RETURN
```

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

```
:REPORT ERROR IF GO BIT IS NOT RESET
1$: BIT    #GO,RMCS11  :GO BIT RESET??
    BEQ    2$         :YES!!
    MOV    RMCS11,$BDDAT :BAD DATA FOR TYPEOUT
    BIC    #^CGO,$BDDAT
    CLR    $GDDAT      :GOOD DATA FOR TYPEOUT
    ADD    #4,(SP)
    MOVB   #11,@(SP)  :ERROR NUMBER
    SUB    #2,(SP)    :MOVE SP TO RETURN FOR ERROR
    JSR    PC,@(SP)+  :REPORT DEVICE NOT AVAILABLE
    SUB    #10,(SP)   :RESTORE (SP)
```

```
037404
037404 013737 001412 043544
037412 042737 177701 043544
037420 062716 000004
037424 105076 000000
037430 162716 000004
037434 032737 000200 001350
037442 001024
037444 013737 001350 001142
037452 042737 177577 001142
037460 012737 000200 001140
037466 062716 000004
037472 112776 000010 000000
037500 162716 000002
037504 004736
037506 162716 000010
037512 000240
037514 032737 000001 001336
037522 001423
037524 013737 001336 001142
037532 042737 177776 001142
037540 005037 001140
037544 062716 000004
037550 112776 000011 000000
037556 162716 000002
037562 004736
037564 162716 000010
037570 000240
```

```

58
59
60 037572 013737 001336 001142 :REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
61 037600 042737 177701 001142 2$: MOV RMCS1I,$BDDAT ;IS FUNCTION CODE CORRECT??
62 037606 013737 043544 001140 BIC #^C76,$BDDAT
63 037614 023737 001142 001140 MOV 81,$SGDDAT ;EXPECTED FUNCTION CODE
64 037622 001413 CMP $BDDAT,$SGDDAT
65 037624 062716 000004 BEQ 3$ ;YES!!
66 037630 112776 000012 000000 ADD #4,(SP)
67 037636 162716 000002 MOVB #12,@(SP) ;ERROR NUMBER
68 037642 004736 JSR PC,@(SP)+ ;MOVE SP TO RETURN FOR ERROR
69 037644 162716 000010 SUB #10,(SP) ;REPORT WRONG FUNCTION CODE
70 037650 000240 NOP ;RESTORE (SP)
71 037652
72
73
74
75 037652 005037 001140 3$: CLR $GDDAT ;EXPECT 'ERR' = 0
76 037656 005737 001352 TST RMER1I ;IS RMER1 = 0??
77 037662 001003 BNE 4$ ;NO!!
78 037664 005737 001400 TST RMER2I ;IS RMERZ = 0??
79 037670 001403 BEQ 5$ ;YES!!
80 037672 052737 040000 001140 4$: BIS #ERR,$GDDAT ;'ERR' SOULD BE SET
81 037700 013737 001350 001142 5$: MOV RMDSI,$BDDAT
82 037706 042737 137777 001142 BIC #^CERR,$BDDAT
83 037714 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS 'ERR' OK??
84 037722 001412 BEQ 6$ ;YES!!
85 037724 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
86 037730 112776 000047 000000 MOVB #47,@(SP) ;WRITE ERROR NUMBER
87 037736 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
88 037742 004736 JSR PC,@(SP)+ ;REPORT INVALID COMP ERROR
89 037744 162716 000010 SUB #10,(SP)
90
91
92
93
94 037750 005037 001140 :REPORT AN ERROR IF 'TRE' IS SET AND NONE OF THE BITS WHICH SET
95 037754 013746 001346 :TRE IS SET, OR IF TRE IS NOT SET AND ONE OR MORE BITS WHICH
96 037760 042726 000377 :SET TRE IS SET
97 037764 001010 6$: CLR $GDDAT ;EXPECT 'TRE' = 0
98 037766 032737 040000 001350 MOV RMCS2I,-(SP) ;WAS DLT, WCE, UPE, NED, NEM
99 037774 001407 BIC #377,(SP)+ ;PGE, MXF OR MDPE SET
100 037776 022737 000030 043544 BNE 7$ ;YES!!
101 040004 103003 BIT #ERR,RMDSI ;WAS EXCEPTION RECEIVED??
102 040006 052737 040000 001140 7$: BEQ 8$ ;NO!!
103 040014 013737 001336 001142 8$: CMP #SEARCH,81$ ;WAS DATA TRANSFERRED??
104 040022 042737 137777 001142 BHIS 8$ ;NO!!
105 040030 023737 001140 001142 BIS #TRE,$GDDAT ;'TRE' SHOULD BE SET
106 040036 001413 BEQ 9$ ;BAD DATA FOR TYPEOUT
107 040040 062716 000004 ADD #4,(SP) ;IS 'TRE' OK??
108 040044 112776 000014 000000 MOVB #14,@(SP) ;YES!!
109 040052 162716 000002 SUB #2,(SP) ;MOVE SP TO USER'S ERROR CALL
110 040056 004736 JSR PC,@(SP)+ ;WRITE ERROR NUMBER
111 040060 162716 000010 SUB #10,(SP) ;MOVE SP TO RETURN FOR ERROR
112 040064 000240 NOP ;REPORT TRE ERROR
113 040066
114 9$:
    
```

```

115
116
117
118
119
120
121
122
123
124 040066 010046
125 040070 013700 043544
126 040074 016037 064704 043536
127 040102 012600
128
129
130
131 040104 013737 043536 001140
132 040112 032737 040000 001350
133 040120 001403
134 040122 052737 100000 001140
135 040130 042737 077777 001140 10$:
136 040136 013737 001350 001142
137 040144 042737 077777 001142
138 040152 023737 001140 001142
139 040160 001413
140 040162 062716 000004
141 040166 112776 000006 000000
142 040174 162716 000002
143 040200 004736
144 040202 162716 000010
145 040206 000240
146 040210
147
148
149
150 040210 013737 043536 001140
151 040216 042737 177776 001140
152 040224 013737 001352 001142
153 040232 042737 177776 001142
154 040240 023737 001140 001142
155 040246 001412
156 040250 062716 000004
157 040254 112776 000254 000000
158 040262 162716 000002
159 040266 004736
160 040270 162716 000010
161 040274 005037 001140 12$:
162
163
164 040300 013746 043536
165 040304 052716 137777
166 040310 013737 001346 001142
167 040316 042637 001142
168 040322 001412
169 040324 062716 000004
170 040330 112776 000026 000000
171 040336 162716 000002
    
```

 :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 R0,-(SP) ;:PUSH R0 ON STACK
 MOV 81\$,R0 ;:GET FUNCTION CODE
 MOV FNCDTB(R0),78\$;:STORE ENTRY
 MOV (SP)+,R0 ;:POP STACK INTO R0
 :REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
 :ATA IS NOT SET AND SHOULD BE SET.
 MOV 78\$, \$GDDAT ;:GET EXPECTED ATA STATUS
 BIT #ERR,RMDSI ;:IS COMPOSITE ERROR SET ??
 BEQ 10\$;:NO !!
 BIS #ATA,\$GDDAT ;:EXPECT AN ATTENTION
 BIC #^CATA,\$GDDAT ;:STRIP DONT CARES
 MOV RMDSI,\$BDDAT ;:GET RECEIVED ATA
 BIC #^CATA,\$BDDAT ;:STRIP DONT CARES
 CMP \$GDDAT,\$BDDAT ;:IS ATA OK ??
 BEQ 11\$;:YES !!
 ADD #4,(SP) ;:MOVE SP TO USERS ERROR CALL
 MOVB #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
 11\$:
 :REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
 :WITH FUNCTION CODE TABLE
 MOV 78\$, \$GDDAT ;:GET EXPECTED ILF
 BIC #^CILF,\$GDDAT ;:CLEAR ALL OTHER BITS
 MOV RMER11,\$BDDAT ;:GET RECEIVED ILF
 BIC #^CILF,\$BDDAT ;:CLEAR ALL OTHER BITS
 CMP \$GDDAT,\$BDDAT ;:IS ILF OK ??
 BEQ 12\$;:YES !!
 ADD #4,(SP) ;:MOVE SP TO USERS ERROR CALL
 MOVB #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
 CLR \$GDDAT ;:CLEAR EXPECTED STATUS
 12\$:
 :REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
 MOV 78\$,-(SP) ;:GET WCE STATUS ENABLE
 BIS #^CWCE,(SP) ;:SET ALL OTHER BITS
 MOV RMCS21,\$BDDAT ;:RECEIVED STATUS
 BIC (SP)+,\$BDDAT ;:CLEAR WCE IF ENABLED
 BEQ 13\$;:BRANCH IF WCE OK
 ADD #4,(SP) ;:MOVE SP TO USER'S ERROR CALL
 MOVB #26,@(SP) ;:WRITE ERROR NUMBER
 SUB #2,(SP) ;:MOVE SP TO ERROR RETURN

```

172 040342 004736          JSR      PC,@(SP)+      ;REPORT ERROR
173 040344 162716 000010  SUB      #10,(SP)        ;RESTORE ERROR
174 040350          13$:
175
176          ;REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
177 040350 013746 043536    MOV      78$,-(SP)      ;GET OPI STATUS ENABLE
178 040354 052716 157777    BIS      #^COPI,(SP)   ;SET ALL OTHER BITS
179 040360 013737 001352 001142  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
180 040366 042637 001142    BIC      (SP)+,$BDDAT  ;CLEAR OPI IF ENABLED
181 040372 001412          BEQ      14$           ;BRANCH IF OPI OK
182 040374 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
183 040400 112776 000164 000000  MOVB     #164,@(SP)    ;WRITE ERROR NUMBER IN CALL
184 040406 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
185 040412 004736          JSR      PC,@(SP)+      ;REPORT ERROR
186 040414 162716 000010  SUB      #10,(SP)      ;RESTORE SP
187 040420          14$:
188
189          ;REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
190          ;SET AND VV IS NOT RESET
191 040420 013746 043536    MOV      78$,-(SP)      ;GET IVC STATUS ENABLE
192 040424 032737 000100 001350  BIT      #VV,RMDSI     ;IS VV SET
193 040432 001402          BEQ      15$           ;NO !!
194 040434 042716 010000    BIC      #IVC,(SP)     ;YES - IVC SHOULD BE 0
195 040440 052716 167777    BIS      #^CIVC,(SP)  ;SET ALL OTHER BITS
196 040444 013737 001400 001142  MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
197 040452 042637 001142    BIC      (SP)+,$BDDAT  ;CLEAR IVC IF ENABLED
198 040456 001412          BEQ      16$           ;BRANCH IF IVC OK
199 040460 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
200 040464 112776 000165 000000  MOVB     #165,@(SP)    ;WRITE ERROR NUMBER IN CALL
201 040472 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
202 040476 004736          JSR      PC,@(SP)+      ;REPORT ERROR
203 040500 162716 000010  SUB      #10,(SP)      ;RESTORE SP TO NO ERROR
204 040504          16$:
205
206          ;BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
207          ; ALL WRITE ERRORS, I.E.,
208          ;       RMER1 - WLE, WCF
209          ;       RMER2 - DPE
210          ;       RMCS2 - UPE.
211          ;EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
212          ;WRITE ERROR ENABLE BIT IS RESET.
213
214          ;REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
215          ;THE DRIVE IS NOT WRITE PROTECTED
216 040504 012746 177777    MOV      #-1,-(SP)     ;ASSUME WRITE ERRORS ENABLED
217 040510 032737 004000 043536  BIT      #WLE,78$     ;ARE WRITE ERRORS ENABLED ??
218 040516 001404          BEQ      17$           ;NO !!
219 040520 032737 004000 001350  BIT      #WRL,RMDSI   ;IS THE DRIVE WRITE PROTECTED ??
220 040526 001002          BNE      18$           ;YES !!
221 040530 042716 004000 17$:    BIC      #WLE,(SP)     ;RESET WLE ENABLE
222 040534 013737 001352 001142 18$:    MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
223 040542 042637 001142    BIC      (SP)+,$BDDAT  ;CLEAR WLE IF ENABLED
224 040546 001412          BEQ      19$           ;BRANCH IF WLE OK
225 040550 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
226 040554 112776 000023 000000  MOVB     #23,@(SP)    ;WRITE ERROR NUMBER IN CALL
227 040562 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
228 040566 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
    
```

```

229 040570 162716 000010          SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
230 040574          19$:
231
232          ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
233 040574 012746 177777          MOV      #-1,-(SP)          ;ASSUME WRITE ERRORS ENABLED
234 040600 032737 004000 043536  BIT      #WLE,78$          ;ARE WRITE ERRORS ENABLED ??
235 040606 001002          BNE      20$              ;YES !!
236 040610 042716 000040          BIC      #WCF,(SP)          ;DISABLE WCF ERROR
237 040614 013737 001352 001142 20$:  MOV      RMER1I,$BDDAT      ;GET RECEIVED STATUS
238 040622 042637 001142          BIC      (SP)+,$BDDAT      ;RESET WCF IF ENABLED
239 040626 001412          BEQ      21$              ;BRANCH IF WCF OK
240 040630 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
241 040634 112776 000025 000000  MOVB     #25,@(SP)          ;WRITE ERROR NUMBER IN CALL
242 040642 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
243 040646 004736          JSR      PC,@(SP)+        ;REPORT ERROR
244 040650 162716 000010          SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
245 040654          21$:
246
247          ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
248 040654 012746 177777          MOV      #-1,-(SP)          ;ASSUME WRITE ERRORS ARE ENABLED
249 040660 032737 004000 043536  BIT      #WLE,78$          ;ARE WRITE ERRORS ENABLED ??
250 040666 001002          BNE      22$              ;YES !!
251 040670 042716 000010          BIC      #DPE,(SP)          ;RESET DPE ENABLE
252 040674 013737 001400 001142 22$:  MOV      RMER2I,$BDDAT      ;GET RECEIVED STATUS
253 040702 042637 001142          BIC      (SP)+,$BDDAT      ;RESET DPE IF ENABLED
254 040706 001412          BEQ      23$              ;BRANCH IF DPE OK
255 040710 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
256 040714 112776 000040 000000  MOVB     #40,@(SP)          ;WRITE ERROR NUMBER IN CALL
257 040722 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
258 040726 004736          JSR      PC,@(SP)+        ;REPORT ERROR
259 040730 162716 000010          SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
260 040734          23$:
261
262          ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
263 040734 012746 177777          MOV      #-1,-(SP)          ;ASSUME WRITE ERRORS ARE ENABLED
264 040740 032737 004000 043536  BIT      #WLE,78$          ;ARE WRITE ERRORS ENABLED ??
265 040746 001002          BNE      24$              ;YES !!
266 040750 042716 020000          BIC      #UPE,(SP)          ;DISABLE UPE ERROR
267 040754 013737 001346 001142 24$:  MOV      RMCS2I,$BDDAT      ;GET RECEIVED STATUS
268 040762 042637 001142          BIC      (SP)+,$BDDAT      ;RESET UPE IF ENABLED
269 040766 001412          BEQ      25$              ;BRANCH IF UPE OK
270 040770 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
271 040774 112776 000024 000000  MOVB     #24,@(SP)          ;WRITE ERROR NUMBER IN CALL
272 041002 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
273 041006 004736          JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
274 041010 162716 000010          SUB      #10,(SP)         ;MOVE SP TO NO ERROR
275 041014          25$:
276
277          ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
278 041014 013746 043536          MOV      78$,-(SP)          ;GET IAE ENABLE
279 041020 052716 175777          BIS      #^CIAE,(SP)       ;SET ALL OTHER BITS
280 041024 013737 001352 001142  MOV      RMER1I,$BDDAT      ;GET RECEIVED STATUS
281 041032 042637 001142          BIC      (SP)+,$BDDAT      ;CLEAR IAE IF ENABLED
282 041036 001412          BEQ      26$              ;BRANCH IF IAE IS OK
283 041040 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
284 041044 112776 000166 000000  MOVB     #166,@(SP)        ;WRITE ERROR NUMBER
285 041052 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
    
```

```

286 041056 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
287 041060 162716 000010  SUB      #10,(SP)        ;MOVE SP TO NO ERROR
288 041064          26$:
289
290          ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
291          ; ALL READ/WRITE ERRORS, I.E.,
292          ;
293          ; RMCS1 - TRE
294          ; RMCS2 - DLT,NEM,MXF
295          ; RMDS  - LBT
296          ; RMER1 - AOE
297          ;NOTE:
298          ; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
299          ; CYLINDER REGISTER IS WRITTEN
300          ;NOTE:
301          ; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
302          ;NOTE:
303          ; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
304
305          ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
306 041064 012746 177777    MOV      #-1,-(SP)      ;ASSUME ERRORS ARE ENABLED
307 041070 032737 001000 043536 BIT      #AOE,78$      ;ARE ERRORS ENABLED ??
308 041076 001002          BNE      27$           ;YES !!
309 041100 042716 100000    BIC      #DLT,(SP)     ;RESET DLT ENABLE
310 041104 013737 001346 001142 27$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
311 041112 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
312 041116 001412          BEQ      28$           ;BRANCH IF DLT IS OK
313 041120 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
314 041124 112776 000032 000000 MOVB     #32,@(SP)     ;WRITE ERROR NUMBER IN CALL
315 041132 162716 000002    SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
316 041136 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
317 041140 162716 000010  SUB      #10,(SP)        ;MOVE SP TO NO ERROR
318 041144          28$:
319
320          ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
321 041144 012746 177777    MOV      #-1,-(SP)      ;ASSUME ERRORS ARE ENABLED
322 041150 032737 001000 043536 BIT      #AOE,78$      ;ARE ERRORS ENABLED ??
323 041156 001002          BNE      29$           ;YES !!
324 041160 042716 004000    BIC      #NEM,(SP)    ;DISABLE NEM
325 041164 013737 001346 001142 29$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
326 041172 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
327 041176 001412          BEQ      30$           ;BRANCH IF NEM IS OK
328 041200 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
329 041204 112776 000167 000000 MOVB     #167,@(SP)   ;WRITE ERROR NUMBER IN CALL
330 041212 162716 000002    SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
331 041216 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
332 041220 162716 000010  SUB      #10,(SP)        ;MOVE SP TO NO ERROR
333 041224          30$:
334
335          ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
336 041224 012746 177777    MOV      #-1,-(SP)      ;ASSUME ERRORS ARE ENABLED
337 041230 032737 001000 043536 BIT      #AOE,78$      ;ARE DATA ERRORS ENABLED ??
338 041236 001002          BNE      31$           ;YES !!
339 041240 042716 001000    BIC      #MXF,(SP)    ;DISABLE MXF ERROR
340 041244 013737 001346 001142 31$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
341 041252 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR MXF IF ENABLED
342 041256 001412          BEQ      32$           ;BRANCH IF MXF IS OK
    
```



```

343 041260 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
344 041264 112776 000033 000000  MOVB    #33,@(SP)       ;WRITE ERROR NUMBER IN CALL
345 041272 162716 000002          SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
346 041276 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
347 041300 162716 000010          SUB      #10,(SP)       ;MOVE SP TO NO ERROR
348 041304          32$:
349
350          ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
351 041304 012746 177777          MOV      #-1,-(SP)      ;ASSUME DATA ERRORS ARE ENABLED
352 041310 032737 001000 043536  BIT      #AOE,78$      ;ARE DATA ERRORS EAMBLD ??
353 041316 001404          BEQ     33$            ;NO !!
354 041320 032737 002000 001350  BIT      #LBT,RMDSI    ;IS LBT ALSO SET ??
355 041326 001002          BNE     34$            ;YES !!
356 041330 042716 001000 33$:      BIC      #AOE,(SP)      ;DISABLE AOE
357 041334 013737 001352 001142 34$:      MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
358 041342 042637 001142          BIC      (SP)+,$BDDAT  ;CLEAR AOE IF ENABLED
359 041346 001412          BEQ     35$            ;BRANCH IF AOE IS OK
360 041350 062716 000004          ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
361 041354 112776 000020 000000  MOVB    #20,@(SP)      ;WRITE ERROR NUMBER
362 041362 162716 000002          SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
363 041366 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
364 041370 162716 000010          SUB      #10,(SP)       ;MOVE SP TO NO ERROR
365 041374          35$:
366
367          ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
368          ;HEADER ERRORS, I.E.,
369          ;
370          ;
371          ;
372          ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
373 041374 032737 002000 001370  BIT      #HCI,RMOFI    ;IS HCI SET ??
374 041402 001403          BEQ     36$            ;NO !!
375 041404 042737 000200 043536  BIC      #HCE,78$      ;YES - DISABLE ALL HEADER ERRORS
376 041412          36$:
377
378          ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
379 041412 012746 177777          MOV      #-1,-(SP)      ;ASSUME ERRORS ENABLED
380 041416 032737 000200 043536  BIT      #HCE,78$      ;ARE HEADER ERRORS ENABLED ??
381 041424 001002          BNE     37$            ;YES !!
382 041426 042716 000400          BIC      #HCRC,(SP)    ;DISABLE HCRC
383 041432 013737 001352 001142 37$:      MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
384 041440 042637 001142          BIC      (SP)+,$BDDAT  ;RESET HCRC IF ENABLED
385 041444 001412          BEQ     38$            ;BRANCH IF HCRC IS OK
386 041446 062716 000004          ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
387 041452 112776 000035 000000  MOVB    #35,@(SP)      ;WRITE ERROR NUMBER IN CALL
388 041460 162716 000002          SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
389 041464 004736          JSR     PC,@(SP)+      ;REPORT ERROR AND RETURN
390 041466 162716 000010          SUB      #10,(SP)       ;MOVE SP TO NO ERROR
391 041472          38$:
392
393          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
394 041472 012746 177777          MOV      #-1,-(SP)      ;ASSUME ERRORS ENABLED
395 041476 032737 000200 043536  BIT      #HCE,78$      ;ARE ERRORS ENABLED ??
396 041504 001002          BNE     39$            ;YES !!
397 041506 042716 000200          BIC      #HCE,(SP)     ;DISABLE HCE
398 041512 013737 001352 001142 39$:      MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
399 041520 042637 001142          BIC      (SP)+,$BDDAT  ;CLEAR HCE IF ENABLED
    
```

```

400 041524 001412          BEQ      40$          ;BRANCH IF HCE IS OK
401 041526 062716 000004  ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
402 041532 112776 000036 000000  MOVVB   #36,@(SP)    ;WRITE ERROR NUMBER IN CALL
403 041540 162716 000002  SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
404 041544 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
405 041546 162716 000010  SUB      #10,(SP)     ;MOVE SP TO NO ERROR
406 041552          40$:
407
408          ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
409 041552 012746 177777  MOV      #-1,-(SP)    ;ASSUME FER IS ENABLED
410 041556 032737 000200 043536  BIT      #HCE,78$    ;ARE HEADER ERRORS ENABLED ??
411 041564 001002          BNE      41$          ;YES !!
412 041566 042716 000020  BIC      #FER,(SP)    ;DISABLE FER
413 041572 013737 001352 001142 41$:  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
414 041600 042637 001142  BIC      (SP)+,$BDDAT ;RESET FER IF ENABLED
415 041604 001412          BEQ      42$          ;BRANCH IF FER OK
416 041606 062716 000004  ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
417 041612 112776 000037 000000  MOVVB   #37,@(SP)    ;WRITE ERROR NUMBER IN CALL
418 041620 162716 000002  SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
419 041624 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
420 041626 162716 000010  SUB      #10,(SP)     ;MOVE SP TO NO ERROR
421 041632          42$:
422
423          ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
424 041632 012746 177777  MOV      #-1,-(SP)    ;ASSUME ERRORS ENABLED
425 041636 032737 000200 043536  BIT      #HCE,78$    ;ARE THEY ENABLED ??
426 041644 001002          BNE      43$          ;YES !!
427 041646 042716 100000  BIC      #BSE,(SP)    ;DISABLE BSE
428 041652 013737 001400 001142 43$:  MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
429 041660 042637 001142  BIC      (SP)+,$BDDAT ;CLEAR BSE IF ENABLED
430 041664 001412          BEQ      44$          ;BRANCH IF BSE OK
431 041666 062716 000004  ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
432 041672 112776 000354 000000  MOVVB   #354,@(SP)   ;WRITE ERROR NUMBER
433 041700 162716 000002  SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
434 041704 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
435 041706 162716 000010  SUB      #10,(SP)     ;MOVE SP TO NO ERROR
436 041712          44$:
437
438          ;HCE IS USED AS THE ENABLING BIT FOR DETECTING SSE, SO REESTABLISH ENTRY
439          ;FROM THE FUNCTION CODE TABLE, TO CHECK THE "SSE" BIT
440 041712 010046          MOV      R0,-(SP)    ;;PUSH R0 ON STACK
441 041714 013700 043544          MOV      81$,R0      ;GET FUNCTION CODE
442 041720 016037 064704 043536  MOV      FNCDTB(R0),78$ ;STORE FUNCTION TABLE ENTRY
443 041726 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
444
445          ;IF SKIP SECTOR ERROR INHIBIT (SSEI) IS SET OR FORMAT 16 (FMT) IS CLEAR,
446          ;RESET THE ENABLING BIT (HCE), OTHERWISE LEAVE ENABLING BIT (HCE) SET.
447 041730 032737 010000 001370  BIT      #FMT16,RMOFI ;18 BIT FORMAT MODE ?
448 041736 001404          BEQ      45$          ;BR IF YES
449 041740 032737 001000 001370  BIT      #SSEI,RMOFI  ;IS SSEI SET ?
450 041746 001403          BEQ      46$          ;BR IF NO
451 041750 042737 000200 043536 45$:  BIC      #HCE,78$    ;DISABLE HEADER ERROR
452 041756          46$:
453
454          ;REPORT ERROR IF SSE IS SET AND HEADER ERROR IS NOT ENABLED
455 041756 012746 177777  MOV      #-1,-(SP)    ;ASSUME ERRORS ENABLED
456 041762 032737 000200 043536  BIT      #HCE,78$    ;IS HCE ENABLED ?
    
```

```

457 041770 001002          BNE      47$      :YES !!
458 041772 042716 000040    BIC      #SSE,(SP) :DISABLE SSE
459 041776 013737 001400 001142 47$:  MOV      RMER2I,$BDDAT :GET RECEIVED STATUS
460 042004 042637 001142    BIC      (SP)+,$BDDAT :CLEAR SSE IF NOT VALID FOR FUNCTION
461 042010 001412          BEQ      48$      :BRANCH IF SSE OK
462 042012 062716 000004    ADD      #4,(SP)   :MOVE SP TO USERS ERROR CALL
463 042016 112776 000063 000000  MOVB     #63,@(SP) :WRITE ERROR NUMBER
464 042024 162716 000002    SUB      #2,(SP)   :MOVE SP TO ERROR RETURN
465 042030 004736          JSR      PC,@(SP)+ :REPORT ERROR AND RETURN
466 042032 162716 000010    SUB      #10,(SP)  :MOVE SP TO NO ERROR
467 042036
468
469
470
471
472
473
474
475
476
477
478 042036 012746 177777          :REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
479 042042 032737 000100 043536  MOV      #-1,-(SP) :ASSUME ENABLED
480 042050 001002          BIT      #ECH,78$  :ARE DATA FIELD ERRORS ENABLED ??
481 042052 042716 000400    BNE      49$      :YES !!
482 042056 013737 001346 001142 49$:  BIC      #MDPE,(SP) :DISBALE MDPE
483 042064 042637 001142    MOV      RMCS2I,$BDDAT :GET RECEIVED STATUS
484 042070 001412          BIC      (SP)+,$BDDAT :CLEAR MDPE IF ENABLED
485 042072 062716 000004    BEQ      50$      :BRANCH IF MDPE OK
486 042076 112776 000027 000000  ADD      #4,(SP)   :MOVE SP TO USERS ERROR CALL
487 042104 162716 000002    MOVB     #27,@(SP) :WRITE ERROR NUMBER IN CALL
488 042110 004736          SUB      #2,(SP)   :MOVE SP TO ERROR RETURN
489 042112 162716 000010    JSR      PC,@(SP)+ :REPORT ERROR AND RETURN
490 042116          SUB      #10,(SP)  :MOVE SP TO NO ERROR
491
492
493 042116 012746 177777          :REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
494 042122 032737 000100 043536  MOV      #-1,-(SP) :ASSUME ENABLED
495 042130 001002          BIT      #ECH,78$  :ARE THEY ENABLED ??
496 042132 042716 100000    BNE      51$      :YES !!
497 042136 013737 001352 001142 51$:  BIC      #DCK,(SP) :DISABLE DCK
498 042144 042637 001142    MOV      RMER1I,$BDDAT :GET RECEIVED STATUS
499 042150 001412          BIC      (SP)+,$BDDAT :CLEAR DCK IF ENABLED
500 042152 062716 000004    BEQ      52$      :BRANCH IF DCK IS OK
501 042156 112776 000030 000000  ADD      #4,(SP)   :MOVE SP TO USERS ERROR CALL
502 042164 162716 000002    MOVB     #30,@(SP) :WRITE ERROR NUMBER IN CALL
503 042170 004736          SUB      #2,(SP)   :MOVE SP TO ERROR RETURN
504 042172 162716 000010    JSR      PC,@(SP)+ :REPORT ERROR AND RETURN
505 042176          SUB      #10,(SP)  :MOVE SP TO NO ERROR
506
507
508
509
510
511 042176 012746 177777          :REPORT ERROR IF ECH IS SET AND,
512 042202 032737 000100 043536  :DATA FIELD ERRORS ARE NOT ENABLED, OR
513 042210 001410          :ECI IS SET, OR
                    :DCK IS NOT SET.
                    MOV      #-1,-(SP)  :ASSUME ENABLED
                    BIT      #ECH,78$  :ARE ERRORS ENABLED ??
                    BEQ      53$      :NO !!

```

```
514 042212 032737 004000 001370 BIT #ECI,RMOFI ;IS ECI SET ??
515 042220 001004 BNE 53$ ;YES !!
516 042222 032737 100000 001352 BIT #DCK,RMER1I ;IS DCK ALSO SET ??
517 042230 001002 BNE 54$ ;YES !!
518 042232 042716 000100 53$: BIC #ECH,(SP) ;DISABLE ECH
519 042236 013737 001352 001142 54$: MOV RMER1I,$BDDAT ;GET RECEIVED STATUS
520 042244 042637 001142 BIC (SP)+,$BDDAT ;CLEAR ECH IF ENABLED
521 042250 001412 BEQ 55$ ;BRANCH IF ECH IS OK
522 042252 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
523 042256 112776 000031 000000 MOVB #31,@(SP) ;WRITE ERROR NUMBER IN CALL
524 042264 162716 000002 SUB #2,(SP) ;MOVE SP TO ERROR RETURN
525 042270 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
526 042272 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
527 042276 55$:
528
529
530 ::*****
531 ::*PERFORM THE REMAINING ERROR CHECKS FOR DATA TRANSFER COMMANDS
532 ::*****
533 042276 022737 000030 043544 CMP #SEARCH,81$ ;WAS DATA TRANSFERRED ?
534 042304 103402 BLO 56$ ;BR IF YES
535 042306 000137 043510 JMP 75$ ;NO - EXIT
536
537 ;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
538 042312 013737 001340 001142 56$: MOV RMWCI,$BDDAT ;WORD COUNT ZERO??
539 042320 001421 BEQ 57$ ;YES
540 042322 032737 040000 001336 BIT #TRE,RMCS1I ;TRANSFER ERROR DETECTED??
541 042330 001015 BNE 57$ ;YES!!
542 042332 062716 000004 ADD #4,(SP)
543 042336 112776 000015 000000 MOVB #15,@(SP) ;ERROR NUMBER
544 042344 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
545 042350 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
546 042354 004736 JSR PC,@(SP)+ ;REPORT WORD COUNT NOT ZERO
547 042356 162716 000010 SUB #10,(SP) ;RESTORE (SP)
548 042362 000240 NOP
549
550 ;REPORT ERROR IF RMBA IS NOT CORRECT
551 042364 013737 001340 001140 57$: MOV RMWCI,$GDDAT ;GET WORD COUNT AT END OF TRANSFER AND
552 042372 163737 001414 001140 SUB RMWCO,$GDDAT ;SUBTRACT STARTING WORD COUNT.
553 042400 006337 001140 ASL $GDDAT ;* 2
554 042404 063737 001416 001140 ADD RMBAO,$GDDAT ;ADD STARTING BUS ADDRESS
555
556 042412 032737 000010 001346 BIT #BAI,RMCS2I ;WAS BUS ADDRESS INHIBIT (BAI) SET ??
557 042420 001403 BEQ 58$ ;NO !!
558 042422 013737 001416 001140 MOV RMBAO,$GDDAT ;ADDRESS SHOULD NOT HAVE CHANGED
559
560 042430 023737 001140 001342 58$: CMP $GDDAT,RMBAI ;BUS ADDRESS OK??
561 042436 001416 BEQ 59$ ;YES!!
562 042440 013737 001342 001142 MOV RMBAI,$BDDAT ;BAD DATA FOR TYPEOUT
563 042446 062716 000004 ADD #4,(SP)
564 042452 112776 000016 000000 MOVB #16,@(SP) ;ERROR NUMBER
565 042460 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
566 042464 004736 JSR PC,@(SP)+ ;REPORT UNEXPECTED ADDRESS
567 042466 162716 000010 SUB #10,(SP) ;RESTORE (SP)
568 042472 000240 NOP
569
570 ;COMPUTE NUMBER OF SECTORS TRANSFERRED FROM WORD COUNT
```

```

571 042474 005046          59$: CLR      -(SP)      ;NUMBER OF SECTORS TRANSFERRED
572 042476 013746 001340  MOV     RMWCI,-(SP)  ;GET WORD COUNT AT END OF TRANSFER AND
573 042502 163716 001414  SUB     RMWCO,(SP)  ;SUBTRACT STARTING WORD COUNT.
574
575 042506 012746 0004C0          MOV     #256,-(SP)  ;ASSUME 256. WORDS PER SECTOR
576 042512 032737 000002 001412  BIT     #BIT1,RMCS10 ;HEADER & DATA COMMAND ??
577 042520 001402          BEQ     60$        ;NO !!
578 042522 062716 000002          ADD     #2,(SP)    ;CHANGE TO 258. WORDS PER SECTOR
579
580 042526 005266 000004 60$: INC     4(SP)      ;INCREMENT SECTOR COUNT
581 042532 161666 000002          SUB     (SP),2(SP) ;SUBTRACT ONE SECTOR'S WORTH
582 042536 003373          BGT     60$        ;CONTINUE IF NOT DONE
583 042540 022626          CMP     (SP)+,(SP)+ ;RESTORE STACK
584
585 ;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM
586 ;NUMBER OF SECTORS
587 042542 013737 001446 043536  MOV     RMDCO,78$   ;STORE ORIGINAL CYLINDER
588 042550 013737 001420 043540  MOV     RMDAO,79$   ;STORE ORIGINAL TRACK
589 042556 013737 001420 043542  MOV     RMDAO,80$   ;STORE ORIGINAL SECTOR
590 042564 013737 001334 043546  MOV     LSTRK,82$   ;STORE LAST TRACK,
591 042572 000337 043546          SWAB    82$        ;GET TRACK ADDRESS TO LO BYTE AND
592 042576 005237 043546          INC     82$        ;INCREMENT TO GET TOTL # OF TRACKS.
593
594 042602 042737 000377 043540  BIC     #^CTADMSK,79$ ;SAVE TRACK ADDRESS BITS AND
595 042610 000337 043540          SWAB    79$        ;GET TRACK ADDRESS TO LO BYTE.
596 042614 042737 177400 043542  BIC     #^CSADMSK,80$ ;SAVE SECTOR ADDRESS BITS
597 042622 062637 043542          ADD     (SP)+,80$
598
599 042626 032737 001000 001444  BIT     #SSEI,RMOFO ;WAS SSEI SET ?
600 042634 001411          BEQ     61$        ;NO !!
601
602 042636 023727 043542 000040  CMP     80$,#32.   ;SECTOR OVEFLOWED??
603 042644 103417          BLO     63$        ;NO!!
604 042646 005237 043540          INC     79$        ;INCREMENT TRACK
605 042652 162737 000040 043542  SUB     #32.,80$   ;ADJUST SECTOR
606
607 042660 023727 043542 000037 61$: CMP     80$,#31.   ;SECTOR OVERFLOWED ?
608 042666 103406          BLO     63$        ;NO !!
609 042670 005237 043540          INC     79$        ;INCREMENT TRACK
610 042674 162737 000037 043542  SUB     #31.,80$   ;ADJUST SECTOR
611 042702 000766          BR      61$        ;TRY AGAIN
612
613 042704 023737 043540 043546 63$: CMP     79$,82$   ;TRACK OVERFLOWED??
614 042712 103407          BLO     64$        ;NO!!
615 042714 005237 043536          INC     78$        ;INCREMENT CYLINDER
616 042720 163737 043546 043540  SUB     82$,79$   ;ADJUST TRACK
617 042726 000766          BR      63$        ;TRY AGAIN
618 042730 000240          NOP
619
620 ;REPORT ERROR IF "SSEI" IS SET AFTER TRACK SWITCHING OCCURED
621 ;THRU DATA TRANSFER
622 042732 005037 001140 64$: CLR     $GDDAT    ;SETUP EXPECTED SSEI = 0
623 042736 013737 001370 001142  MOV     RMOFI,$BDDAT ;GET RECIEVED DATA AND
624 042744 042737 176777 001142  BIC     #^CSSEI,$BDDAT ;SAVE SSEI BIT FOR COMPARE.
625
626 042752 032737 001000 001444  BIT     #SSEI,RMOFO ;WAS SSEI SET BEFORE DATA TRANSFER ?
627 042760 001420          BEQ     66$        ;NO
    
```

```

628 042762 123737 001421 001345      CMPB   RMDAO+1,RMDAI+1 ;DID TRACK SWITCH OCCUR ?
629 042770 001411                      BEQ    65$              :NO
630 042772 005737 001142                      TST   $BDDAT          :IS SSEI STILL SET ?
631 042776 001430                      BEQ    68$              :NO
632 043000 062716 000004                      ADD   #4,(SP)         :POINT TO ERROR
633 043004 112776 000225 000000                      MOVB  #225,a(SP)     :DEFINE ERROR NUMBER
634 043012 000414                      BR    67$              :
635
636                                     :REPORT ERROR IF "SSEI" IS INCORRECT
637 043014 012737 001000 001140 65$:  MOV   #SSEI,$GDDAT   :SETUP EXPECTED SSEI = 1
638 043022 023737 001140 001142 66$:  CMP   $GDDAT,$BDDAT  :IS SSEI CORRECT ?
639 043030 001413                      BEQ    68$              :YES
640 043032 062716 000004                      ADD   #4,(SP)         :POINT TO ERROR
641 043036 112776 000226 000000                      MOVB  #226,a(SP)     :DEFINE ERROR NUMBER
642 043044 162716 000002                      67$:  SUB   #2,(SP)      :ADJUST RETURN TO ERROR
643 043050 004736                      JSR   PC,a(SP)+       :REPORT ERROR
644 043052 162716 000010                      SUB   #10,(SP)       :RESTORE (SP)
645 043056 000240                      NOP
646
647                                     :REPORT ERROR IF "LBT" IS NOT CORRECT
648 043060 005037 001140 001060 68$:  CLR   $GDDAT         :SET GOOD DATA FOR LBT = 0
649 043064 023727 043536 001060                      CMP   78$,#560.      :SHOULD LBT BE SET??
650 043072 101407                      BLOS  69$              :NO!!
651 043074 032737 002000 001352                      BIT   #IAE,RMER1I    :WAS IAE SET ??
652 043102 001003                      BNE   69$              :YES - LBT SHOULD NOT BE SET
653 043104 012737 002000 001140                      MOV   #LBT,$GDDAT    :SET GOOD DATA FOR LBT = 1
654 043112 013737 001350 001142 69$:  MOV   RMDSI,$BDDAT   :BAD DATA FOR TYPEOUT
655 043120 042737 175777 001142                      BIC   #^CLBT,$BDDAT
656 043126 023737 001140 001142                      CMP   $GDDAT,$BDDAT :IS LBT CORRECT??
657 043134 001413                      BEQ    70$              :YES!!
658 043136 062716 000004                      ADD   #4,(SP)
659 043142 112776 000017 000000                      MOVB  #17,a(SP)     :ERROR NUMBER
660 043150 162716 000002                      SUB   #2,(SP)        :MOVE SP TO RETURN FOR ERROR
661 043154 004736                      JSR   PC,a(SP)+       :REPORT LBT IS WRONG
662 043156 162716 000010                      SUB   #10,(SP)       :RESTORE (SP)
663 043162 000240                      NOP
664
665                                     :REPORT ERROR IF "AOE" IS INCORRECT
666 043164 005037 001140 001352 70$:  CLR   $GDDAT         :SET FOR AOE = 0
667 043170 032737 002000 001352                      BIT   #IAE,RMER1I    :WAS "IAE" DETECTED??
668 043176 001031                      BNE   72$              :YES-"AOE" SHOULD BE ZERO
669 043200 023727 043536 001060                      CMP   78$,#560.      :SHOULD AOE BE SET??
670 043206 101425                      BLOS  72$              :NO!!
671 043210 005737 043540                      TST   79$              :MAYBE
672 043214 001012                      BNE   71$              :YES
673 043216 005737 043542                      TST   80$
674 043222 001007                      BNE   71$              :YES !!
675 043224 032737 000010 043544                      BIT   #F2,81$ ;WAS THIS READ OR WRITE CHECK ??
676 043232 001413                      BEQ    72$              :NO !!
677 043234 005737 001340                      TST   RMWCI          :WAS ALL DATA TRANSFERRED ??
678 043240 001410                      BEQ    72$              :YES !!
679 043242 012737 001000 001140 71$:  MOV   #AOE,$GDDAT    :SET FOR AOE = 1
680 043250 005037 043540                      CLR   79$              :CLEAR EXPECTED TRACK
681 043254 012737 000001 043542                      MOV   #1,80$         :EXPECT SECTOR = 1
682 043262 013737 001352 001142 72$:  MOV   RMER1I,$BDDAT  :BAD DATA FOR TYPEOUT
683 043270 042737 176777 001142                      BIC   #^CAOE,$BDDAT
684 043276 023737 001140 001142                      CMP   $GDDAT,$BDDAT :IS AOE CORRECTY??
    
```

```

685 043304 001413          BEQ      73$          ;YES!!
686 043306 062716 000004    ADD      #4,(SP)
687 043312 112776 000020 000000    MOVEB   #20,@(SP)    ;ERROR NUMBER
688 043320 162716 000002    SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
689 043324 004736          JSR      PC,@(SP)+   ;REPORT AOE IS WRONG
690 043326 162716 000010    SUB      #10,(SP)   ;RESTORE (SP)
691 043332 000240          NOP
692
693
694 043334 032737 002000 001352  ;REPORT ERROR IF RMDA IS NOT CORRECT
73$: BIT      #IAE,RMER1I ;WAS THERE AN IAE ERROR ??
695 043342 001062          BNE      75$          ;YES - DONT CHECK RMDA,RMDC
696 043344 013737 043540 001140    MOV      79$,SGDDAT ;SETUP EXPECTED DISK ADDRESS
697 043352 000337 001140          SWAB    $GDDAT
698 043356 113737 043542 001140    MOVEB   80$,SGDDAT
699 043364 013737 001344 001142    MOV      RMDAI,$BDDAT ;SETUP RECEIVED DISK ADDRESS
700 043372 023737 001140 001142    CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
701 043400 001413          BEQ      74$          ;BRANCH IF EQUAL
702 043402 062716 000004    ADD      #4,(SP)
703 043406 112776 000021 000000    MOVEB   #21,@(SP)   ;ERROR NUMBER
704 043414 162716 000002    SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
705 043420 004736          JSR      PC,@(SP)+   ;REPORT BAD DISK ADDRESS
706 043422 162716 000010    SUB      #10,(SP)   ;RESTORE (SP)
707 043426 000240          NOP
708
709
710 043430 013737 043536 001140  ;REPORT ERROR IF RMDC IS INCORRECT
74$: MOV      78$,SGDDAT ;SETUP EXPECTED CYLINDER
711 043436 042737 176000 001140    BIC      #^C1777,$GDDAT
712 043444 013737 001372 001142    MOV      RMDCI,$BDDAT ;SETUP RECEIVED CYLINDER
713 043452 023737 001140 001142    CMP      $GDDAT,$BDDAT ;COMPARE CYLINDERS
714 043460 001413          BEQ      75$          ;BRANCH IF EQUAL
715 043462 062716 000004    ADD      #4,(SP)
716 043466 112776 000022 000000    MOVEB   #22,@(SP)   ;ERROR NUMBER
717 043474 162716 000002    SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
718 043500 004736          JSR      PC,@(SP)+   ;REPORT BAD CYLINDER
719 043502 162716 000010    SUB      #10,(SP)   ;RESTORE (SP)
720 043506 000240          NOP
721
722 043510 062716 000004 75$:  ADD      #4,(SP)     ;MOVE (SP) TO ERROR CALL
723 043514 105776 000000    TSTB    @(SP)        ;WAS ERROR FOUND??
724 043520 001403          BEQ      76$
725 043522 062716 000004    ADD      #4,(SP)     ;MOVE (SP) TO ERROR RETURN
726 043526 000402          BR      77$
727 043530 162716 000004 76$:  SUB      #4,(SP)     ;MOVE (SP) TO NO ERROR RETURN
728 043534 000207 77$:  RTS      PC
729
730 043536 000000 78$:  .WORD   0           ;CYLINDER
731 043540 000000 79$:  .WORD   0           ;TRACK
732 043542 000000 80$:  .WORD   0           ;SECTOR
733 043544 000000 81$:  .WORD   0           ;FUNCTION CODE
734 043546 000000 82$:  .WORD   0           ;TOTAL # OF TRACKS = LAST TRACK +1
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.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      MASK FOR ERROR REGISTER 1
      .WORD                MASK FOR ERROR REGISTER 2
      .WORD                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
      ???
```

: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,$TMP1      ;STORE RMER1 AT TEMP STORAGE
BIC @($P),$TMP1      ;MASK RMER1
ADD #2,$P            ;MOVE SP TO NEXT MASK
MOV RMER2I,$TMP2      ;STORE RMER2 AT TEMP STORAGE
BIC @($P),$TMP2      ;MASK RMER2
```

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

```
:SEE IF THERE WERE ANY ERRORS IN RMER1, I.E., $TMP1
TST $TMP1           ;ANY ERRORS TO REPORT??
BEQ 1$              ;NO !!
MOV $TMP1,$BDDAT    ;RECEIVED STATUS FOR TYPEOUT
CLR $GDDAT          ;EXPECTED STATUS FOR TYPEOUT
ADD #4,$P           ;MOVE SP TO USER'S ERROR CALL
MOVB #6,@($P)      ;CORRECTABLE DATA CHECK ERROR #
SUB #2,$P           ;MOVE SP TO RETURN FOR ERROR
JSR PC,@($P)+      ;REPORT ERROR VIA USER
SUB #10,$P         ;MOVE SP BACK TO BRANCH
NOP
```

1\$:

```
:SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 ($TMP2)
TST $TMP2           ;ANY ERRORS IN RMER2?
BEQ 2$              ;NO!!
MOV $TMP2,$BDDAT    ;RECEIVED STATUS FOR TYPEOUT
CLR $GDDAT          ;EXPECTED STATUS FOR TYPEOUT
ADD #4,$P           ;MOVE SP TO USER'S ERROR CALL
MOVB #6,@($P)      ;WRITE ERROR NUMBER IN USER'S CALL
SUB #2,$P           ;MOVE SP TO RETURN FOR ERROR
```

```
043550
043550 013737 001352 001176
043556 047637 000000 001176
043564 062716 000002
043570 013737 001400 001200
043576 047637 000000 001200
043604 062716 000006
043610 105076 000000
043614 162716 000004
043620 005737 001176
043624 001420
043626 013737 001176 001142
043634 005037 001140
043640 062716 000004
043644 112776 000066 000000
043652 162716 000002
043656 004736
043660 162716 000010
043664 000240
043666
043666 005737 001200
043672 001420
043674 013737 001200 001142
043702 005037 001140
043706 062716 000004
043712 112776 000067 000000
043720 162716 000002
```


58	043724	004736		JSR	PC,@(SP)+	:REPORT ERROR VIA USER	
59	043726	162716	000010	SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN	
60	043732	000240		NOP			
61	043734			2\$:			
62							
63				:AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED			
64	043734	062716	000004	ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL	
65	043740	105776	000000	TSTB	@(SP)	:WAS THERE AN ERROR CALLED??	
66	043744	001403		BEQ	3\$:NO!!	
67	043746	062716	000004	ADD	#4,(SP)	:YES - MOVE SP TO ERROR RETURN	
68	043752	000402		BR	4\$		
69	043754	162716	000004	3\$:	SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
70	043760	000207		4\$:	RTS	PC	:RETURN TO USER

```

1      .SBTTL  DEVICE SELECT SUBROUTINE
2
3      ; THIS SUBROUTINE SELECTS THE DEVICE, GETTING THE DEVICE NUMBER FROM THE
4      ; TEST QUEUE.
5
6      ;CALL:
7      ;(1)  JSR      PC,DEVSEL
8      ;(2)  BR       ??          RETURN IF NO ERROR
9      ;(3)  NOP
10     ;(4)  ERROR          RETURN IF ERROR
11                                ERROR DEFINED BY SUBROUTINE
12 043762
13
14     ;CLEAR USER'S ERROR CALL
15 043762 062716 000004      ADD     #4,(SP)          ;MOVE SP TO USER'S ERROR
16 043766 105076 000000      CLR    @10(SP)         ;CLEAR LOW ORDER BYTE OF CALL
17 043772 162716 000004      SUB     #4,(SP)          ;MOVE SP BACK
18
19     ;SAVE USER'S INFORMATION AND SETUP REGISTERS
20 043776 013746 000004      MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
21 044002 013746 000006      MOV     ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
22 044006 010046              MOV     R0,-(SP)       ;;PUSH R0 ON STACK
23 044010 010146              MOV     R1,-(SP)       ;;PUSH R1 ON STACK
24 044012 012737 044132 000004  MOV     #2$,ERRVEC     ;SETUP FOR BUS TIMEOUT
25 044020 012737 000300 000006  MOV     #PR6,ERRVEC+2
26 044026 013700 001276      MOV     $BASE,R0      ;R0 = UNIBUS ADDRESS
27 044032 013701 001466      MOV     TSTQUE,R1     ;R1 POINTS TO DEVICE NUMBER
28
29     ;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
30
31 044036 111160 000010      MOV    (R1),RMCS2(R0) ;WRITE UNIT SELECT BITS
32 044042 016037 000000 001176  MOV     RMCS1(R0),$TMP1 ;GET 'DVA' STATUS
33 044050 016037 000010 001174  MOV     RMCS2(R0),$TMP0 ;GET 'NED' STATUS
34
35 044056 032737 010000 001174  BIT     #NED,$TMP0     ;IS DEVICE NONEXISTENT ?
36 044064 001407              BEQ     1$             ;NO!!
37 044066 062766 000004 000010  ADD     #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
38 044074 112776 000111 000010  MOV    #111,@10(SP)   ;WRITE ERROR NUMBER
39 044102 000422              BR      3$
40
41 044104 032737 004000 001176 1$: BIT     #DVA,$TMP1     ;IS DEVICE AVAILABLE ?
42 044112 001021              BNE     4$             ;YES!!
43 044114 062766 000004 000010  ADD     #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
44 044122 112776 000112 000010  MOV    #112,@10(SP)   ;WRITE ERROR NUMBER
45 044130 000407              BR      3$
46
47     ;HANDLE BUS TIMEOUT
48
49 044132 022626 2$: CMP     (SP)+,(SP)+  ;ADJUST SP
50 044134 062766 000004 000010  ADD     #4,10(SP)      ;MOVE SP TO USERS ERROR CALL
51 044142 112776 000113 000010  MOV    #113,@10(SP)   ;WRITE BUS TIMEOUT ERROR NUMBER
52 044150 162766 000002 000010 3$: SUB     #2,10(SP)     ;ADJUST RETURN TO 'NOP' PRECEDING
53                                ;THE ERROR CALL
54
55     ;RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
56
57 044156 012601 4$: MOV     (SP)+,R1      ;;POP STACK INTO R1
58 044160 012600      MOV     (SP)+,R0      ;;POP STACK INTO R0
59 044162 012637 000006      MOV     (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
    
```

044166 012637 000004
52 044172 000207

MOV
RTS

(SP)+,ERRVEC
PC

::POP STACK INTO ERRVEC
:EXIT

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.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   ???
      NOP
      ERROR
      JSR PC,@(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
    
```

SEKSTS:

:CLEAR USERS' 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 20$ ;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, CYLINDER,
 :AND THE STATUS OF "SSEI" BIT DURING A SEEK OPERATION. ALSO, SET "SKI"
 :IF CYLINDER ADDRESS IS TOO LARGE.

```

1$: MOV #IAE,$GDDAT ;SETUP FOR IAE = 1
    BIS #SKI,20$ ;SETUP FOR SKI = 1
    CMP RMDCO,#560. ;GREATER THAN LAST CYLINDER ?
    BHI 3$ ;YES - CYLINDER IS INVALID
    BIC #SKI,20$ ;CLEAR SKI ERROR FLAG

    CMPB RMDAO+1,LSTRK+1 ;GREATER THAN LAST TRACK ?
    BHI 3$ ;YES - TRACK IS INVALID

    CMPB RMDAO,#29. ;SECTOR > 29. ?
    
```

```

19 044174
22 044174 000240
23 044176 062716 000004
24 044202 105076 000000
25 044206 162716 000004
26 044212 005037 045452
30 044216 032737 000010 001352
31 044224 001424
32 044226 032737 000010 001400
33 044234 001020
36 044236 005037 001140
37 044242 013737 001352 001142
38 044250 062716 000004
39 044254 112776 000050 000000
40 044262 162716 000002
41 044266 004736
42 044270 162716 000010
43 044274 000447
48 044276 012737 002000 001140
49 044304 052737 040000 045452
50 044312 023727 001446 001060
51 044320 101035
52 044322 042737 040000 045452
54 044330 123737 001421 001335
55 044336 101026
57 044340 123727 001420 000035
    
```

```

58 044346 101420          BLOS 2$          ;BR IF NO
59 044350 032737 010000 001444  BIT  #FMT16,RMOFO ;18 BIT FORMAT ?
60 044356 001416          BEQ  3$          ;YES - SECTOR IS INVALID FOR 18 BIT MODE
61 044360 123727 001420 000036  CMPB RMDAO,#30.  ;SECTOR > 30. ?
62 044366 101410          BLOS 2$          ;BR IF NO
63 044370 032737 001000 001444  BIT  #SSEI,RMOFO ;IS SSEI CLEAR ?
64 044376 001406          BEQ  3$          ;YES - SECTOR IS INVALID FOR 16 BIT MODE
65 044400 123727 001420 000037  CMPB RMDAO,#31.  ;SECTOR > 31. ?
66 044406 101002          BHI  3$          ;YES - SECTOR IS INVALID
67
68 044410 005037 001140          2$: CLR  $GDDAT    ;"IAE" SHOULD = 0
69
70          ;COMPARE EXPECTED AND RECIEVED "IAE" STATUS
71 044414 013737 001352 001142  3$: MOV  RMER1I,$BDDAT ;IS IAE OK??
72 044422 042737 175777 001142  BIC  #^CIAE,$BDDAT ;SAVE IAE BIT FOR COMPARE
73 044430 023737 001140 001142  CMP  $GDDAT,$BDDAT ;CORRECT "IAE" STATUS ?
74 044436 001004          BNE  4$          ;BR IF NO
75 044440 042737 040000 045452  BIC  #SKI,20$    ;CLEAR SKI FLAG
76 044446 000413          BR   5$          ;GO CHECK NEXT ERROR
77 044450
78          4$: ;REPORT INCORRECT "IAE" STATUS VIA USER'S ERROR CALL
79 044450 062716 000004          ADD  #4,(SP)
80 044454 112776 000051 000000  MOVB #51,a(SP)   ;ERROR 51
81 044462 162716 000002          SUB  #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
82 044466 004736          JSR  PC,a(SP)+   ;REPORT INCORRECT IAE
83 044470 162716 000010          SUB  #10,(SP)   ;RESTORE (SP)
84 044474 000240
85 044476          5$:
86
87          ;REPORT ANY IVC ERROR AS
88          ; IVC ERROR WITH VOLUME VALID ZERO
89          ; ERRONEOUS IVC ERROR, VOLUME VALID IS SET
90 044476 032737 010000 001400  BIT  #IVC,RMER2I ;IVC ERROR??
91 044504 001427          BEQ  7$          ;NO!!
92 044506 005037 001140          CLR  $GDDAT     ;EXPECTED STATUS
93 044512 013737 001400 001142  MOV  RMER2I,$BDDAT ;RECEIVED STATUS
94 044520 062716 000004          ADD  #4,(SP)   ;MOVE SP TO USER'S ERROR
95 044524 112776 000060 000000  MOVB #60,a(SP)   ;ERROR 60 IF VV = 0
96 044532 032737 000100 001350  BIT  #VV,RMDSI
97 044540 001403          BEQ  6$          ;ERROR 61 IF VV = 1
98 044542 112776 000061 000000  MOVB #61,a(SP)   ;MOVE SP TO RETURN FOR ERROR
99 044550 162716 000002          6$: SUB  #2,(SP)     ;REPORT ERROR VIA USER
100 044554 004736          JSR  PC,a(SP)+  ;RESTORE SP
101 044556 162716 000010          SUB  #10,(SP)
102 044562 000240          NOP
103
104 044564 013737 001400 001142  7$: MOV  RMER2I,$BDDAT ;RECEIVED STATUS
105 044572 042737 137777 001142  BIC  #^CSKI,$BDDAT ;CLEAR DONT CARES
106 044600 013737 045452 001140  MOV  20$,$GDDAT  ;GET EXPECTED SKI STATUS
107 044606 042737 137777 001140  BIC  #^CSKI,$GDDAT ;CLEAR DONT CARES
108 044614 001417          BEQ  8$          ;BRANCH IF 0 EXPECTED
109
110          ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
111 044616 032737 040000 001142  BIT  #SKI,$BDDAT ;WAS SKI DETECTED ??
112 044624 001032          BNE  9$          ;YES !!
113 044626 062716 000004          ADD  #4,(SP)   ;MOVE SP TO USERS ERROR CALL
114 044632 112776 000267 000000  MOVB #267,a(SP)  ;WRITE ERROR NUMBER
    
```

```

115 044640 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
116 044644 004736                JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
117 044646 162716 000010          SUB      #10,(SP)       ;MOVE SP TO NO ERROR
118 044652 000443                BR       10$           ;GO TO NEXT ERROR CHECK
119 044654
120
121                                ;REPORT ERROR IF SKI IS SET
122 044654 032737 040000 001142          BIT      #SKI,$BDDAT    ;IS SKI SET ??
123 044662 001413                BEQ      9$            ;NO - SKI IS OK
124 044664 062716 000004                ADD      #4,(SP)        ;MOVE (SP) TO ERROR
125 044670 112776 000054 000000          MOVB    #54,@(SP)      ;LOAD ERROR NUMBER
126 044676 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
127 044702 004736                JSR      PC,@(SP)+      ;REPORT SEEK ERROR
128 044704 162716 000010          SUB      #10,(SP)       ;RESTORE (SP)
129 044710 000240                NOP
130
131                                ;REPORT ANY DEVICE CHECK
132 044712 032737 000200 001400          9$: BIT      #DVC,RMER2I  ;WAS THERE DVC DURING SEEK??
133 044720 001420                BEQ      10$           ;NO!!
134 044722 005037 001140                CLR      $GDDAT        ;EXPECTED STATUS
135 044726 013737 001400 001142          MOV      RMER2I,$BDDAT ;RECEIVED STATUS
136 044734 062716 000004                ADD      #4,(SP)
137 044740 112776 000055 000000          MOVB    #55,@(SP)      ;ERROR #55
138 044746 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
139 044752 004736                JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
140 044754 162716 000010          SUB      #10,(SP)       ;RESTORE SP
141 044760 000240                NOP
142
143                                ;REPORT ANY 'OPI' ERROR AS OPI WITH MOL = 0, OR OPI
144                                ;BECAUSE ON CYLINDER LATCH DIDN'T RESET
145 044762 032737 020000 001352          10$: BIT      #OPI,RMER1I ;'OPI' ERROR??
146 044770 001427                BEQ      12$           ;NO!!
147 044772 005037 001140                CLR      $GDDAT        ;EXPECTED STATUS
148 044776 013737 001352 001142          MOV      RMER1I,$BDDAT ;RECEIVED STATUS
149 045004 062716 000004                ADD      #4,(SP)
150 045010 112776 000052 000000          MOVB    #52,@(SP)      ;LOAD ERROR NUMBER
151 045016 032737 010000 001350          BIT      #MOL,RMDSI    ;WAS MEDIUM ON LINE??
152 045024 001403                BEQ      11$           ;NO!!
153 045026 112776 000053 000000          MOVB    #53,@(SP)      ;YES - CHANGE ERROR NUMBER
154 045034 162716 000002          11$: SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
155 045040 004736                JSR      PC,@(SP)+      ;REPORT 'OPI' ERROR
156 045042 162716 000010          SUB      #10,(SP)       ;RESTORE (SP)
157 045046 000240                NOP
158
159                                ;SEE IF 'PIP' = 0, AND 'ATA', 'MOL' AND 'VV' = 1
160 045050 013746 001350          12$: MOV      RMDSI,-(SP)
161 045054 042716 047677          BIC     #^C<ATA!PIP!MOL!VV>,(SP)
162 045060 022726 110100          CMP     #ATA!MOL!VV,(SP)+
163 045064 001002          BNE     13$           ;ERROR IN RMDS
164 045066 000137 045422          JMP     17$           ;RMDS IS OK
165
166                                ;REPORT ERROR IF MOL = 0 AND OPI = 0
167 045072 032737 010000 001350          13$: BIT      #MOL,RMDSI  ;IS MOL RESET??
168 045100 001030                BNE     14$           ;NO - MOL IS SET
169 045102 032737 020000 001352          BIT      #OPI,RMER1I  ;WAS OPI SET
170 045110 001024                BNE     14$           ;YES - DONT REPORT ERROR
171 045112 013737 001350 001140          MOV      RMDSI,$GDDAT  ;EXPECTED STATUS
    
```

```

172 045120 052737 010000 001140      BIS      #MOL,$GDDAT
173 045126 013737 001350 001142      MOV      RMDSI,$BDDAT      ;RECEIVED STATUS
174 045134 062716 000004                ADD      #4,(SP)
175 045140 112776 000062 000000      MOVVB   #62,@(SP)
176 045146 162716 000002                SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
177 045152 004736                JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
178 045154 162716 000010                SUB      #10,(SP)
179 045160 000240                NOP
180
181                                     ;REPORT AN ERROR IF 'PIP' IS STIL SET AND SKI NOT SET
182 045162 032737 020000 001350      14$: BIT      #PIP,RMDSI      ;IS 'PIP' STILL SET??
183 045170 001430                BEQ      15$            ;NO!!
184 045172 032737 040000 001400      BIT      #SKI,RMER2I     ;WAS 'SKI' SET??
185 045200 001024                BNE      15$            ;YES-DONT REPORT PIP
186 045202 013737 001350 001140      MOV      RMDSI,$GDDAT     ;EXPECTED STATUS
187 045210 042737 020000 001142      BIC      #PIP,$BDDAT
188 045216 013737 001350 001142      MOV      RMDSI,$BDDAT     ;RECEIVED STATUS
189 045224 062716 000004                ADD      #4,(SP)      ;MOVE (SP) TO ERROR
190 045230 112776 000056 000000      MOVVB   #56,@(SP)      ;LOAD ERROR NUMBER
191 045236 162716 000002                SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
192 045242 004736                JSR      PC,@(SP)+      ;REPORT 'PIP' SET AFTER SEEK
193 045244 162716 000010                SUB      #10,(SP)      ;RESTORE (SP)
194 045250 000240                NOP
195
196                                     ;REPORT AN ERROR IF 'ATA' IS NOT SET
197 045252 032737 100000 001350      15$: BIT      #ATA,RMDSI     ;WAS 'ATA' SET ??
198 045260 001024                BNE      16$            ;YES!!
199 045262 013737 001350 001140      MOV      RMDSI,$GDDAT     ;EXPECTED STATUS
200 045270 052737 110600 001140      BIS      #ATA!MOL!DPR!DRY,$GDDAT
201 045276 013737 001350 001142      MOV      RMDSI,$BDDAT     ;RECEIVED STATUS
202 045304 062716 000004                ADD      #4,(SP)      ;MOVE (SP) TO ERROR
203 045310 112776 000057 000000      MOVVB   #57,@(SP)      ;LOAD ERROR NUMBER
204 045316 162716 000002                SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
205 045322 004736                JSR      PC,@(SP)+      ;REPORT ATTENTION NOT SET DURING
206                                     ;SEEK TEST
207 045324 162716 000010                SUB      #10,(SP)      ;RESTORE (SP)
208 045330 000240                NOP
209
210                                     ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
211 045332 032737 000100 001350      16$: BIT      #VV,RMDSI     ;IS VV = 0 ??
212 045340 001030                BNE      17$            ;NO!!
213 045342 032737 010000 001400      BIT      #IVC,RMER2I     ;IS IVC ALSO 0 ??
214 045350 001024                BNE      17$            ;NO - IVC IS SET
215 045352 013737 001350 001140      MOV      RMDSI,$GDDAT     ;EXPECTED STATUS
216 045360 052737 000100 001140      BIS      #VV,$GDDAT
217 045366 013737 001350 001142      MOV      RMDSI,$BDDAT     ;RECEIVED STATUS
218 045374 062716 000004                ADD      #4,(SP)
219 045400 112776 000064 000000      MOVVB   #64,@(SP)      ;ERROR #64
220 045406 162716 000002                SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
221 045412 004736                JSR      PC,@(SP)+
222 045414 162716 000010                SUB      #10,(SP)
223 045420 000240                NOP
224 045422
225
226                                     ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
227 045422 000240                NOP
228 045424 062716 000004                ADD      #4,(SP)      ;MOVE (SP) TO ERROR CALL
    
```

229	045430	105776	000000		TSTB	@(SP)	:WAS ERROR CALLED??
230	045434	001403			BEQ	18\$:NO!!
231	045436	062716	000004		ADD	#4,(SP)	:MOVE TO ERROR RETURN
232	045442	000402			BR	19\$	
233							
234	045444	162716	000004	18\$:	SUB	#4,(SP)	:MOVE (SP) TO NO ERROR RETURN
235	045450	000207		19\$:	RTS	PC	:RETURN
236							
237	045452	000000		20\$:	.WORD	0	:ERROR FLAGS


```

1      .SBTTL  CONTROLLER CLEAR SUBROUTINE
2
3      ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
4      ;AND DRIVES, THEN SELECTS THE DRIVE.
5
6      :CALL:  JSR      PC,CNTCLR
7              BR      ???
8              NOP
9              ERROR
10             ???
11
12     CNTCLR:  MOV      R0,-(SP)      ;;PUSH R0 ON STACK
13             MOV      R1,-(SP)      ;;PUSH R1 ON STACK
14             MOV      ERRVEC,-(SP)   ;;PUSH ERRVEC ON STACK
15             MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
16             MOV      #1$,ERRVEC    ;SETUP FOR BUS TIMEOUT
17             MOV      #PR6,ERRVEC+2
18             MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
19             MOV      #CLR, RMCS2(R0) ;CLEAR MASSBUS
20             MOV      TSTQUE,R1     ;GET DEVICE UNDER TEST
21             MOVB     (R1),RMCS2(R0) ;SELECT DEVICE
22             BR      2$
23
24     1$:      CMP      (SP)+,(SP)+   ;ADJUST STACK
25             ADD      #4,10(SP)     ;MOVE SP TO USER'S ERROR CALL
26             MOVB     #7,@10(SP)    ;WRITE THE ERROR NUMBER
27             SUB      #2,10(SP)     ;ADJUST SP TO RETURN TO ERROR
28
29     2$:      MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
30             MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
31             MOV      (SP)+,R1      ;;POP STACK INTO R1
32             MOV      (SP)+,R0      ;;POP STACK INTO R0
33             RTS      PC
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.SBTTL CONTROLLER CLEAR STATUS CHECK SUBROUTINE

: THIS SUBROUTINE VERIFIES THAT THE 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      RETURN HERE IF NO ERROR
      BR   ???          RETURN HERE TO REPORT AN ERROR
      NOP
      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

```

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

:REPORT ERROR IF RMCS1 NOT INITIALIZED

```

1$: MOV RMCS1I,$BDDAT ;VERIFY RMCS1
    BIC #SC,$BDDAT    ;IGNORE SPECIAL CONDITION
    MOV #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
    CMP $GDDAT,$BDDAT ;COMPARE EXPECTED, RECEIVED
    BEQ 2$           ;BRANCH IF EQUAL
    ADD #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
    MOVB #126,@(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 NO ERROR
    NOP
    
```

:REPORT ERROR IF RMBA NOT RESET

```

2$: CLR $GDDAT      ;VERIFY RMBA IS ZERO
    MOV RMBAI,$BDDAT
    BEQ 3$         ;BRANCH IF ZERO
    ADD #4,(SP)    ;MOVE SP TO USER'S ERROR CALL
    MOVB #127,@(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 NO ERROR
    NOP
    
```

:REPORT ERROR IF RMCS2 NOT INITIALIZED

```

3$: MOV RMCS2I,$BDDAT ;VERIFY RMCS2
    MOV R1,-(SP)      ;PUSH R1 ON STACK
    CLR -(SP)         ;EXPECT IR & UNIT NUMBER
    MOV TSTQUE,R1     ;R1 = ADDRESS OF TEST QUE
    MOVB (R1),(SP)
    BIS #IR,(SP)
    MOV (SP)+,$GDDAT
    
```

045572

045572 062716 000004

045576 105076 000000

045602 162716 000004

045606 013737 001336 001142

045614 042737 100000 001142

045622 012737 004200 001140

045630 023737 001140 001142

045636 001413

045640 062716 000004

045644 112776 000126 000000

045652 162716 000002

045656 004736

045660 162716 000010

045664 000240

045666 005037 001140

045672 013737 001342 001142

045700 001413

045702 062716 000004

045706 112776 000127 000000

045714 162716 000002

045720 004736

045722 162716 000010

045726 000240

045730 013737 001346 001142

045736 010146

045740 005046

045742 013701 001466

045746 111116

045750 052716 000100

045754 012637 001140

```

58 045760 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
59 045762 023737 001140 001142  CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED & RECEIVED
60 045770 001413          BEQ      4$              ;;BRANCH IF EQUAL
61 045772 062716 000004          ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
62 045776 112776 000130 000000  MOVB    #130,a(SP)       ;;WITE ERROR NUMBER IN CALL
63 046004 162716 000002          SUB      #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
64 046010 004736          JSR     PC,a(SP)+        ;;REPORT ERROR VIA USER
65 046012 162716 000010          SUB      #10,(SP)       ;;MOVE SP BACK TO NO ERROR
66 046016 000240          NOP
67          ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
68 046020 005037 001140 4$: CLR      $GDDAT          ;;VERIFY RMER1
69 046024 013737 001352 001142  MOV     RMER1I,$BDDAT
70 046032 042737 040000 001142  BIC     #UNS,$BDDAT      ;;IGNORE UNSAFE
71 046040 001413          BEQ     5$              ;;BRANCH IF ZERO
72 046042 062716 000004          ADD     #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
73 046046 112776 000131 000000  MOVB   #131,a(SP)       ;;WITE ERROR NUMBER IN CALL
74 046054 162716 000002          SUB     #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
75 046060 004736          JSR    PC,a(SP)+        ;;REPORT ERROR VIA USER
76 046062 162716 000010          SUB     #10,(SP)       ;;MOVE SP BACK TO NO ERROR
77 046066 000240          NOP
78          ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
79 046070 013737 001362 001142  5$: MOV     RMMR1I,$BDDAT  ;;VERIFY RMMR
80 046076 042737 000046 001142  BIC     #WC!LS!LST,$BDDAT ;;IGNORE WORD CLOCK, SCT, TRK
81 046104 012737 000010 001140  MOV     #MWD,$GDDAT      ;;EXPECT WRITE DATA BIT
82 046112 023737 001140 001142  CMP     $GDDAT,$BDDAT    ;;COMPARE EXPECTED AND RECEIVED
83 046120 001413          BEQ     6$              ;;BRANCH IF 0
84 046122 062716 000004          ADD     #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
85 046126 112776 000133 000000  MOVB   #133,a(SP)       ;;WITE ERROR NUMBER IN CALL
86 046134 162716 000002          SUB     #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
87 046140 004736          JSR    PC,a(SP)+        ;;REPORT ERROR VIA USER
88 046142 162716 000010          SUB     #10,(SP)       ;;MOVE SP BACK TO NO ERROR
89 046146 000240          NOP
90          ;REPORT AN ERROR IF RMEC2 IS NOT RESET
91 046150 005037 001140 6$: CLR      $GDDAT          ;;EXPECT ZEROS
92 046154 013737 001404 001142  MOV     RMEC2I,$BDDAT    ;;VERIFY RMEC2=0
93 046162 001413          BEQ     7$              ;;BRANCH IF 0
94 046164 062716 000004          ADD     #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
95 046170 112776 000135 000000  MOVB   #135,a(SP)       ;;WITE ERROR NUMBER IN CALL
96 046176 162716 000002          SUB     #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
97 046202 004736          JSR    PC,a(SP)+        ;;REPORT ERROR VIA USER
98 046204 162716 000010          SUB     #10,(SP)       ;;MOVE SP BACK TO NO ERROR
99 046210 000240          NOP
100          ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
101 046212 013737 001376 001142  7$: MOV     RMMR2I,$BDDAT  ;;VERIFY RMMR2
102 046220 042737 140000 001142  BIC     #RQA!RQB,$BDDAT
103 046226 012737 011777 001140  MOV     #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
104 046234 023737 001140 001142  CMP     $GDDAT,$BDDAT
105 046242 001413          BEQ     8$              ;;BRANCH IF 0
106 046244 062716 000004          ADD     #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
107 046250 112776 000136 000000  MOVB   #136,a(SP)       ;;WITE ERROR NUMBER IN CALL
108 046256 162716 000002          SUB     #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
109 046262 004736          JSR    PC,a(SP)+        ;;REPORT ERROR VIA USER
110 046264 162716 000010          SUB     #10,(SP)       ;;MOVE SP BACK TO NO ERROR
111 046270 000240          NOP
112          ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC
113 046272 005037 001140 8$: CLR      $GDDAT          ;;EXPECT ALL ZEROS
114 046276 013737 001400 001142  MOV     RMER2I,$BDDAT    ;;VERIFY RMER2
    
```

115	046304	042737	040200	001142	BIC	#SKI!DVC,\$BDDAT	:IGNORE DEVICE ERRORS
116	046312	001413			BEQ	9\$:BRANCH IF OTHER BITS 0
117	046314	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
118	046320	112776	000174	000000	MOVB	#174,@(SP)	:WRITE ERROR NUMBER IN CALL
119	046326	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
120	046332	004736			JSR	PC,@(SP)+	:REPORT ERROR VIA USER
121	046334	162716	000010		SUB	#10,(SP)	:MOVE SP BACK TO NO ERROR
122	046340	000240			NOP		
123							:REPORT ERROR IF RMDS NOT INITIALIZED
124	046342	013737	001350	001142	9\$: MOV	RMDSI,\$BDDAT	:TEST DRIVE STATUS REGISTER
125	046350	042737	177177	001142	BIC	#^C<DRY!DPR>,\$BDDAT	
126	046356	012737	000600	001140	MOV	#DPR!DRY,\$GDDAT	:EXPECTED DRIVE STATUS
127	046364	023737	001140	001142	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED & RECEIVED
128	046372	001413			BEQ	10\$:BRANCH IF EQUAL
129	046374	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
130	046400	112776	000134	000000	MOVB	#134,@(SP)	:WRITE ERROR NUMBER
131	046406	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
132	046412	004736			JSR	PC,@(SP)+	:REPORT ERROR TO USER
133	046414	162716	000010		SUB	#10,(SP)	:MOVE SP BACK TO NO ERROR
134	046420	000240			NOP		
135	046422	062716	000004		10\$: ADD	#4,(SP)	:MOVE SP TO ERROR CALL
136	046426	105776	000000		TSTB	@(SP)	:WAS AN ERROE DETECTED??
137	046432	001403			BEQ	11\$:NO!!
138	046434	062716	000004		ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
139	046440	000402			BR	12\$	
140	046442	162716	000004		11\$: SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
141	046446	000240			12\$: NOP		
142	046450	000207			RTS	PC	

CZRNEAO RM80 FCTNL PT2 MACRO V04.00 28-JAN-82 16:38:23 PAGE 30-2

M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.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 ??? 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

ACKSTS:

:CLEAR USER'S ERROR CALL
ADD #4,(SP) :MOVE SP TO ERROR CALL
CLRB @(SP) :CLEAR LOW ORDER BYTE
SUB #4,(SP) :MOVE SP BACK

:REPORT AN ERROR IF 'VV' IS 0
BIT #VV,RMDSI :IS VOLUME VALID SET??
BNE 1\$:YES!!
MOV RMDSI,\$GDDAT :EXPECTED STATUS
BIS #VV,\$GDDAT
MOV RMDSI,\$BDDAT :RECEIVED STATUS
ADD #4,(SP) :MOVE SP TO ERROR CALL
MOVB #155,@(SP) :WRITE NUMBER IN ERROR CALL
SUB #2,(SP) :MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ :REPORT THE ERROR
SUB #10,(SP) :MOVE SP BACK TO BRANCH
NOP

1\$:

:REPORT AN ERROR IF 'MOL' IS 0
BIT #MOL,RMDSI :IS MOL SET??
BNE 2\$:YES!!
MOV RMDSI,\$GDDAT :EXPECTED STATUS
BIS #MOL,\$GDDAT
MOV RMDSI,\$BDDAT :RECEIVED STATUS
ADD #4,(SP) :MOVE SP TO ERROR CALL
MOVB #41,@(SP) :WRITE NUMBER OF ERROR IN CALL
SUB #2,(SP) :MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ :REPORT TH ERROR
SUB #10,(SP) :MOVE SP TO BRANCH
NOP

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 :WAS UNS SET??
BEQ 3\$:NO!!
MOV RMER1I,\$BDDAT :RECEIVED STATUS

CZRNEAO RM80 FCTNL PT2 MACRO V04.00 28-JAN-82 16:38:23 PAGE 31-1
PACK ACKNOWLEDGE STATUS CHECK

58	046654	013737	001352	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
59	046662	042737	040000	001140	BIC	#UNS,\$GDDAT	
60	046670	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
61	046674	112776	000042	000000	MOVB	#42,@(SP)	:WRITE NUMBER OF ERROR IN CALL
62	046702	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
63	046706	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
64	046710	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
65	046714	000240			NOP		
66	046716						
67							
68							
69	046716	032737	020000	001352	:REPORT ANY OPI ERROR		
70	046724	001424			BIT	#OPI,RMER1I	:WAS OPI SET ??
71	046726	013737	001352	001142	BEQ	4\$:NO!!
72	046734	013737	001352	001140	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
73	046742	042737	020000	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
74	046750	062716	000004		BIC	#OPI,\$GDDAT	
75	046754	112776	000043	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
76	046762	162716	000002		MOVB	#43,@(SP)	:WRITE NUMBER OF ERROR IN CALL
77	046766	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
78	046770	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
79	046774	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
80	046776				NOP		
81							
82							
83	046776	032737	000004	001352	:REPORT ANY RMR ERROR		
84	047004	001424			BIT	#RMR,RMER1I	:WAS RMR SET??
85	047006	013737	001352	001142	BEQ	5\$:NO!!
86	047014	013737	001352	001140	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
87	047022	042737	000004	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
88	047030	062716	000004		BIC	#RMR,\$GDDAT	
89	047034	112776	000044	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
90	047042	162716	000002		MOVB	#44,@(SP)	:WRITE NUMBER OF ERROR IN CALL
91	047046	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
92	047050	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
93	047054	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
94	047056				NOP		
95							
96							
97	047056	032737	000002	001352	:REPORT ANY ILR ERROR		
98	047064	001424			BIT	#ILR,RMER1I	:WAS ILR SET??
99	047066	013737	001352	001142	BEQ	6\$:NO!!
100	047074	013737	001352	001140	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
101	047102	042737	000002	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
102	047110	062716	000004		BIC	#ILR,\$GDDAT	
103	047114	112776	000045	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
104	047122	162716	000002		MOVB	#45,@(SP)	:WRITE NUMBER OF ERROR IN CALL
105	047126	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
106	047130	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
107	047134	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
108	047136				NOP		
109							
110							
111	047136	032737	000001	001352	:REPORT ANY ILF ERROR		
112	047144	001424			BIT	#ILF,RMER1I	:WAS ILF SET??
113	047146	013737	001352	001142	BEQ	7\$:NO!!
114	047154	013737	001352	001140	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
					MOV	RMER1I,\$GDDAT	:EXPECTED STATUS

115	047162	042737	000001	001140	BIC	#ILF,\$GDDAT	
116	047170	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
117	047174	112776	000046	000000	MOVB	#46,@(SP)	:WRITE NUMBER OF ERROR IN CALL
118	047202	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
119	047206	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
120	047210	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
121	047214	000240			NOP		
122	047216						
123							
124							
125	047216	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
126	047222	105776	000000		TSTB	@(SP)	:WAS ERROR FOUND??
127	047226	001403			BEQ	8\$:NO!!
128	047230	062716	000004		ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
129	047234	000402			BR	9\$	
130	047236	162716	000004		SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
131	047242	000240			NOP		
132	047244	000207			RTS	PC	

7\$:

:AUGMENT RETURN ADDRESS IF ERROR WAS FOUND

8\$:

9\$:

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```

.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 5$ ;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$:
    
```

047246
 047246 062716 000004
 047252 105076 000000
 047256 162716 000004
 047262 032737 022011 001352
 047270 001553
 047272 032737 000010 001352
 047300 001430
 047302 032737 000010 001400
 047310 001024
 047312 013737 001352 001140
 047320 042737 000010 001140
 047326 013737 001352 001142
 047334 062716 000004
 047340 112776 000050 000000
 047346 162716 000002
 047352 004736
 047354 162716 000010
 047360 000240
 047362
 047362 032737 000001 001352
 047370 001424
 047372 013737 001352 001140
 047400 042737 000001 001140
 047406 013737 001352 001142
 047414 062716 000004
 047420 112776 000071 000000
 047426 162716 000002
 047432 004736
 047434 162716 000010
 047440 000240
 047442


```

58                                     ;REPORT ANY "OPI" ERROR AS
59                                     . OPI DUE TO "MOL" = 0
60                                     . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
61 047442 032737 020000 001352      BIT    #OPI,RMER1I      ;WAS OPI SET??
62 047450 001433                    BEQ    4$              ;NO!!
63 047452 013737 001352 001140      MOV    RMER1I,$GDDAT  ;EXPECTED STATUS
64 047460 042737 020000 001140      BIC    #OPI,$GDDAT
65 047466 013737 001352 001142      MOV    RMER1I,$BDDAT ;RECEIVED STATUS
66 047474 062716 000004              ADD    #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
67 047500 112776 000072 000000      MOVB  #72,a(SP)     ;WRITE ERROR NUMBER IN USER'S CALL
68 047506 032737 010000 001350      BIT    #MOL,RMDSI   ;WAS "MOL" = 0??
69 047514 001403                    BEQ    3$              ;YES!!
70 047516 112776 000073 000000      MOVB  #73,a(SP)     ;NO - CHANGE ERROR NUMBER
71 047524 162716 000002              SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
72 047530 004736                    JSR    PC,a(SP)+     ;REPORT ERROR VIA USER
73 047532 162716 000010              SUB    #10,(SP)     ;MOVE SP BACK TO BRANCH
74 047536 000240                    NOP
75 047540                          4$:
76
77                                     ;REPORT AN ERROR IF "IAE" IS SET
78 047540 032737 002000 001352      BIT    #IAE,RMER1I  ;IS "IAE" SET??
79 047546 001424                    BEQ    5$              ;NO!!
80 047550 013737 001352 001140      MOV    RMER1I,$GDDAT ;EXPECTED STATUS
81 047556 042737 002000 001140      BIC    #IAE,$GDDAT
82 047564 013737 001352 001142      MOV    RMER1I,$BDDAT ;RECEIVED STATUS
83 047572 062716 000004              ADD    #4,(SP)       ;MOVE SP TO ERROR CALL
84 047576 112776 000070 000000      MOVB  #70,a(SP)     ;WRITE ERROR NUMBER IN USER'S CALL
85 047604 162716 000002              SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
86 047610 004736                    JSR    PC,a(SP)+     ;REPORT ERROR
87 047612 162716 000010              SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR RETURN
88 047616 000240                    NOP
89 047620                          5$:
90
91                                     ;SEE IF "SKI" OR "IVC" OR "DVC" IS SET
92 047620 032737 050200 001400      BIT    #SKI!IVC!DVC,RMER2I
93 047626 001517                    BEQ    9$              ;NONE OF THE BITS ARE SET
94
95
96                                     ;REPORT ANY "IVC" ERROR AS
97                                     . IVC WITH VV = 0
98                                     . ERRONEOUS IVC ERROR
99 047630 032737 010000 001400      BIT    #IVC,RMER2I  ;WAS IVC SET??
100 047636 001433                    BEQ    7$              ;NO!!
101 047640 013737 001400 001140      MOV    RMER2I,$GDDAT ;EXPECTED STATUS
102 047646 042737 010000 001140      BIC    #IVC,$GDDAT
103 047654 013737 001400 001142      MOV    RMER2I,$BDDAT ;RECEIVED STATUS
104 047662 062716 000004              ADD    #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
105 047666 112776 000074 000000      MOVB  #74,a(SP)     ;WRITE ERROR NUMBER IN CALL
106 047674 032737 000100 001350      BIT    #VV,RMDSI   ;WAS VV = 0??
107 047702 001403                    BEQ    6$              ;YES!!
108 047704 112776 000075 000000      MOVB  #75,a(SP)     ;NO - CHANGE ERROR NUMBER
109 047712 162716 000002              SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
110 047716 004736                    JSR    PC,a(SP)+     ;REPORT ERROR VIA USER
111 047720 162716 000010              SUB    #10,(SP)     ;MOVE SP BACK TO BRANCH
112 047724 000240                    NOP
113 047726                          7$:
114

```

```

115                                     ;REPORT ANY "SKI" ERROR
116 047726 032737 040000 001400      BIT      #SKI,RMER2I      ;WAS SKI SET??
117 047734 001424                                     BEQ      8$              ;NO!!
118 047736 013737 001400 001140      MOV      RMER2I,$GDDAT   ;EXPECTED STATUS
119 047744 042737 040000 001140      BIC      #SKI,$GDDAT
120 047752 013737 001400 001142      MOV      RMER2I,$BDDAT   ;RECEIVED STATUS
121 047760 062716 000004                                     ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
122 047764 112776 000076 000000      MOVVB   #76,@(SP)       ;WRITE ERROR NUMBER
123 047772 162716 000002                                     SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
124 047776 004736                                     JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
125 050000 162716 000010                                     SUB      #10,(SP)       ;MOVE SP TO BRANCH
126 050004 000240                                     NOP
127 050006                                     8$:
128
129                                     ;REPORT ANY "DVC" ERROR
130 050006 032737 000200 001400      BIT      #DVC,RMER2I    ;WAS "DVC" SET??
131 050014 001424                                     BEQ      9$              ;NO!!
132 050016 013737 001400 001140      MOV      RMER2I,$GDDAT   ;EXPECTED STATUS
133 050024 042737 000200 001140      BIC      #DVC,$GDDAT
134 050032 013737 001400 001142      MOV      RMER2I,$BDDAT   ;RECEIVED STATUS
135 050040 062716 000004                                     ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
136 050044 112776 000077 000000      MOVVB   #77,@(SP)       ;WRITE ERROR NUMBER
137 050052 162716 000002                                     SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
138 050056 004736                                     JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
139 050060 162716 000010                                     SUB      #10,(SP)       ;MOVE SP TO USER'S BRANCH
140 050064 000240                                     NOP
141 050066                                     9$:
142
143                                     ;SEE IF "PIP" AND "OM" ARE 0, AND "ATA","MOL" AND "VV" ARE 1
144 050066 013746 001350                                     MOV      RMDSI,-(SP)     ;PUT RMDI ON STACK
145 050072 042716 047676                                     BIC      #^C<PIP!MOL!VV!OM!ATA>,(SP)
146 050076 022726 110100                                     CMP      #ATA!MOL!VV,(SP)+
147 050102 001002                                     BNE      10$
148 050104 000137 050520                                     JMP      15$
149 050110                                     10$:
150
151                                     ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
152                                     ;LINE AFTER RECALIBRATE WAS INITIATED
153 050110 032737 010000 001350      BIT      #MOL,RMDSI     ;DID MOL DROP??
154 050116 001030                                     BNE      11$            ;NO!!
155 050120 032737 020000 001352      BIT      #OPI,RMER1I    ;WAS OPI ERROR REPORTED??
156 050126 001024                                     BNE      11$            ;YES - DON'T REPORT MOL=0
157 050130 013737 001350 001140      MOV      RMDSI,$GDDAT   ;EXPECTED STATUS
158 050136 052737 010000 001140      BIS      #MOL,$GDDAT
159 050144 013737 001350 001142      MOV      RMDSI,$BDDAT   ;RECEIVED STATUS
160 050152 062716 000004                                     ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
161 050156 112776 000100 000000      MOVVB   #100,@(SP)      ;WRITE ERROR NUMBER
162 050164 162716 000002                                     SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
163 050170 004736                                     JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
164 050172 162716 000010                                     SUB      #10,(SP)       ;MOVE SP BACK TO USER'S BRANCH
165 050176 000240                                     NOP
166 050200                                     11$:
167
168                                     ;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0
169 050200 032737 000100 001350      BIT      #VV,RMDSI      ;DID "VV" DROP??
170 050206 001030                                     BNE      12$            ;NO!!
171 050210 032737 010000 001400      BIT      #IVC,RMER2I    ;WAS THERE A IVC ERROR??
    
```

```

172 050216 001024          BNE      12$          :YES - DONT REPORT VV=0
173 050220 013737 001350 001140  MOV      RMDSI,$GDDAT :EXPECTED STATUS
174 050226 013737 001350 001142  MOV      RMDSI,$BDDAT :RECEIVED STATUS
175 050234 052737 000100 001140  BIS      #VV,$GDDAT
176 050242 062716 000004          ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
177 050246 112776 000101 000000  MOVB    #101,a(SP)   :WRITE ERROR NUMBER IN CALL
178 050254 162716 000002          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
179 050260 004736          JSR     PC,a(SP)+
180 050262 162716 000010          SUB      #10,(SP)    :MOVE SP BACK TO USER'S BRANCH
181 050266 000240          NOP
182 050270          12$:
183
184          :REPORT AN ERROR IF ATA IS NOT SET
185 050270 032737 100000 001350  BIT      #ATA,RMDSI   :WAS ATA SET DURING RECALIBRATE??
186 050276 001024          BNE      13$          :YES!!
187 050300 013737 001350 001140  MOV      RMDSI,$GDDAT :EXPECTED STATUS
188 050306 052737 100000 001140  BIS      #ATA,$GDDAT
189 050314 013737 001350 001142  MOV      RMDSI,$BDDAT :RECEIVED STATUS
190 050322 062716 000004          ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
191 050326 112776 000102 000000  MOVB    #102,a(SP)   :WRITE ERROR NUMBER IN CALL
192 050334 162716 000002          SUB      #2,(SP)      :MOVE SP TO USER'S BRANCH
193 050340 004736          JSR     PC,a(SP)+
194 050342 162716 000010          SUB      #10,(SP)
195 050346 000240          NOP
196
197 050350          13$:
198
199          :REPORT AN ERROR IF 'OM' IS NOT ZERO BECAUSE RECALIBRATE SHOULD
200          :ALWAYS CLEAR OFFSET MODE
201 050350 032737 000001 001350  BIT      #OM,RMDSI   :WAS 'OM' RESET??
202 050356 001424          BEQ     14$          :YES!!
203 050360 013737 001350 001140  MOV      RMDSI,$GDDAT :EXPECTED STATUS
204 050366 042737 000001 001140  BIC      #OM,$GDDAT
205 050374 013737 001350 001142  MOV      RMDSI,$BDDAT :RECEIVED STATUS
206 050402 062716 000004          ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
207 050406 112776 000103 000000  MOVB    #103,a(SP)   :WRITE ERROR NUMBER
208 050414 162716 000002          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
209 050420 004736          JSR     PC,a(SP)+   :REPORT ERROR VIA USER
210 050422 162716 000010          SUB      #10,(SP)    :MOVE SP TO USER'S BRANCH
211 050426 000240          NOP
212 050430          14$:
213
214          :REPORT AN ERROR IF 'PIP' IS STIL ON, I.E., DRIVE NOT ON
215          :CYLINDER
216 050430 032737 020000 001350  BIT      #PIP,RMDSI  :IS DRIVE OFF CYLINDER??
217 050436 001430          BEQ     15$          :NO!!
218 050440 032737 040000 001400  BIT      #SKI,RMER2I :WAS 'SKI' DETECTED??
219 050446 001024          BNE      15$          :YES-DONT REPORT 'PIP'
220 050450 013737 001350 001140  MOV      RMDSI,$GDDAT :EXPECTED STATUS
221 050456 042737 020000 001140  BIC      #PIP,$GDDAT
222 050464 013737 001350 001142  MOV      RMDSI,$BDDAT :RECEIVED STATUS
223 050472 062716 000004          ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
224 050476 112776 000104 000000  MOVB    #104,a(SP)   :WRITE ERROR NUMBER
225 050504 162716 000002          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
226 050510 004736          JSR     PC,a(SP)+
227 050512 162716 000010          SUB      #10,(SP)    :MOVE SP BACK TO USER'S BRANCH
228 050516 000240          NOP
    
```

```

229 050520          15$:
230
231                :SEE IF "ILR" OR "RMR" OR "UNS" IS SET
232 050520 032737 040006 001352      BIT    #ILR!RMR!UNS,RMER1I
233 050526 001514                BEQ    18$
234
235                :REPORT AN ERROR IF "ILR" IS SET
236 050530 032737 000002 001352      BIT    #ILR,RMER1I      :WAS ILR SET DURING RECALIBRATE??
237 050536 001424                BEQ    16$                :NO!!
238 050540 013737 001352 001140      MOV    RMER1I,$GDDAT    :EXPECTED STATUS
239 050546 042737 000002 001140      BIC    #ILR,$GDDAT
240 050554 013737 001352 001142      MOV    RMER1I,$BDDAT    :RECEIVED STATUS
241 050562 062716 000004                ADD    #4,(SP)          :MOVE SP TO USER'S ERROR CALL
242 050566 112776 000105 000000      MOV    #105,a(SP)      :WRITE ERROR NUMBER IN CALL
243 050574 162716 000002                SUB    #2,(SP)          :MOVE SP TO RETURN FOR ERROR
244 050600 004736                JSR    PC,a(SP)+
245 050602 162716 000010                SUB    #10,(SP)         :MOVE SP TO USER'S BRANCH
246 050606 000240                NOP
247
248
249                :REPORT AN ERROR IF "RMR" IS SET
250 050610 032737 000004 001352      BIT    #RMR,RMER1I     :WAS RMR SET??
251 050616 001424                BEQ    17$                :NO!!
252 050620 013737 001352 001140      MOV    RMER1I,$GDDAT    :EXPECTED STATUS
253 050626 042737 000004 001140      BIC    #RMR,$GDDAT
254 050634 013737 001352 001142      MOV    RMER1I,$BDDAT    :RECEIVED STATUS
255 050642 062716 000004                ADD    #4,(SP)          :MOVE SP TO USER'S ERROR CALL
256 050646 112776 000106 000000      MOV    #106,a(SP)      :WRITE ERROR NUMBER IN USER'S CALL
257 050654 162716 000002                SUB    #2,(SP)          :MOVE SP TO RETURN FOR ERROR
258 050660 004736                JSR    PC,a(SP)+
259 050662 162716 000010                SUB    #10,(SP)         :REPORT ERROR VIA USER
260 050666 000240                NOP
261 050670
262
263                :REPORT AN ERROR IF "UNS" IS SET AND "DVC" IS 0
264 050670 032737 040000 001352      BIT    #UNS,RMER1I     :WAS UNSAFE ON??
265 050676 001430                BEQ    18$                :NO!!
266 050700 032737 000200 001400      BIT    #DVC,RMER2I     :WAS THERE A DEVICE CHECK??
267 050706 001024                BNE    18$                :YES - DON'T REPORT UNSAFE
268 050710 013737 001352 001140      MOV    RMER1I,$GDDAT    :EXPECTED STATUS
269 050716 042737 040000 001140      BIC    #UNS,$GDDAT
270 050724 013737 001352 001142      MOV    RMER1I,$BDDAT    :RECEIVED STATUS
271 050732 062716 000004                ADD    #4,(SP)          :MOVE SP TO USER'S ERROR CALL
272 050736 112776 000107 000000      MOV    #107,a(SP)      :WRITE ERROR NUMBER
273 050744 162716 000002                SUB    #2,(SP)          :MOVE SP TO RETURN FOR ERROR
274 050750 004736                JSR    PC,a(SP)+
275 050752 162716 000010                SUB    #10,(SP)         :REPORT ERROR VIA USER
276 050756 000240                NOP
277 050760
278
279                :AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
280 050760 062716 000004                ADD    #4,(SP)          :MOVE SP TO USER'S ERROR CALL
281 050764 105776 000000                TSTB   a(SP)            :WAS AN ERROR REPORTED??
282 050770 001403                BEQ    19$                :NO!!
283 050772 062716 000004                ADD    #4,(SP)          :YES - AUGMENT SP RETURN
284 050776 000402                BR     20$
285 051000 162716 000004      19$:  SUB    #4,(SP) ;NO ERROR - RETURN SP TO BRANCH
    
```

286 051004 000240
287 051006 000207

20\$: NOP
RTS PC

;STATUS CECK IS COMPLETE

1
 2
 3
 4
 5
 6
 7
 8
 9 051010
 10
 11
 12 051010 062716 000004
 13 051014 105076 000000
 14 051020 162716 000004
 15
 16 051024 013737 001336 001142
 17 051032 042737 173700 001142
 18 051040 012737 004010 001140
 19 051046 023737 001140 001142
 20 051054 001443
 21 051056 062716 000004
 22 051062 112776 000141 000000
 23 051070 162716 000002
 24 051074 004736
 25 051076 162716 000010
 26 051102 000240
 27
 28 051104 013737 001350 001142
 29 051112 042737 021101 001142
 30 051120 012737 010600 001140
 31 051126 023737 001140 001142
 32 051134 001413
 33 051136 062716 000004
 34 051142 112776 000142 000000
 35 051150 162716 000002
 36 051154 004736
 37 051156 162716 000010
 38 051162 000240
 39
 40 051164 005037 001140
 41 051170 013737 001352 001142
 42 051176 001413
 43 051200 062716 000004
 44 051204 112776 000143 000000
 45 051212 162716 000002
 46 051216 004736
 47 051220 162716 000010
 48 051224 000240
 49
 50 051226 013737 001354 001142
 51 051234 010146
 52 051236 010246
 53 051240 013701 001466
 54 051244 116102 000001
 55 051250 042702 177400
 56 051254 005102
 57 051256 040237 001142

```

.SBTTL DRIVE CLEAR STATUS CHECK SUBROUTINE
:
: BR      ???      RETURN HERE IF NO ERROR
: NOP
: ERROR   RETURN HERE TO REPORT AN ERROR
: JSR    PC,@(SP)+ ERROR NUMBER DEFINED BY SUB
:      ???      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
CLRB   @ (SP)      :CLEAR ERROR CALL
SUB    #4,(SP)      :MOVE SP TO USER'S BRANCH

: REPORT ERROR IF RMCS1 NOT INITIALIZED
1$:   MOV    RMCS1I,$BDDAT :CHECK RMCS1
      BIC   #^C<DVA!FNCMSK>,$BDDAT :CLEAR DONT CARES
      MOV   #DVA!DRVCLR,$GDDAT :EXPECT DVA
      CMP   $GDDAT,$BDDAT :COMPARE EXPECTED & RECEIVED
      BEQ   3$      :BRANCH IF EQUAL
      ADD   #4,(SP) :MOVE SP TO ERROR CALL
      MOVB #141,@(SP) :WRITE NUMBER OF ERROR IN CALL
      SUB   #2,(SP) :MOVE SP TO RETURN FOR ERROR
      JSR   PC,@(SP)+ :REPORT THE ERROR VIA USER
      SUB   #10,(SP) :MOVE SP TO NO ERROR RETURN
      NOP

: REPORT ERROR IF RMDS NOT INITIALIZED
2$:   MOV    RMDSI,$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   3$      :BRANCH IF EQUAL
      ADD   #4,(SP) :MOVE SP TO ERROR CALL
      MOVB #142,@(SP) :WRITE NUMBER OF ERROR IN CALL
      SUB   #2,(SP) :MOVE SP TO RETURN FOR ERROR
      JSR   PC,@(SP)+ :REPORT THE ERROR VIA USER
      SUB   #10,(SP) :MOVE SP TO NO ERROR RETURN
      NOP

: REPORT ERROR IF RMER1 NOT INITIALIZED
3$:   CLR    $GDDAT :EXPECT 0'S
      MOV   RMER1I,$BDDAT :CHECK RMER1
      BEQ   4$      :BRANCH IF EQUAL
      ADD   #4,(SP) :MOVE SP TO ERROR CALL
      MOVB #143,@(SP) :WRITE NUMBER OF ERROR IN CALL
      SUB   #2,(SP) :MOVE SP TO RETURN FOR ERROR
      JSR   PC,@(SP)+ :REPORT THE ERROR VIA USER
      SUB   #10,(SP) :MOVE SP TO NO ERROR RETURN
      NOP

: REPORT ERROR IF ATA NOT INITIALIZED
4$:   MOV    RMASI,$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   #^CATNMSK,R2
      COM   R2
      BIC   R2,$BDDAT
    
```

```

58 051262 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
59 051264 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
60 051266 005737 001142  TST      $BDDAT      ;;IS ATTENTION CLEARED??
61 051272 001413      BEQ      5$          ;;BRANCH IF ATTENTION CLEARED
62 051274 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
63 051300 112776 000144 000000  MOVB    #144,a(SP)    ;;WRITE NUMBER OF ERROR IN CALL
64 051306 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
65 051312 004736      JSR     PC,a(SP)+     ;;REPORT THE ERROR VIA USER
66 051314 162716 000010      SUB     #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
67 051320 000240      NOP
68                                     :REPORT ERROR IF RMMR1 NOT INITIALIZED
69 051322 013737 001362 001142 5$:      MOV     RMMR1I,$BDDAT  ;;CHECK RMMR
70 051330 042737 000046 001142      BIC    #WC!LS!LST,$BDDAT ;;CLEAR DONT CARES
71 051336 012737 000010 001140      MOV    #MWD,$GDDAT   ;;EXPECT WRITE DATA ON
72 051344 023737 001140 001142      CMP    $GDDAT,$BDDAT ;;COMPARE EXPECTED AND RECEIVED
73 051352 001413      BEQ    6$          ;;BRANCH IF ZERO
74 051354 062716 000004  ADD     #4,(SP)      ;;MOVE SP TO ERROR CALL
75 051360 112776 000145 000000  MOVB   #145,a(SP)    ;;WRITE NUMBER OF ERROR IN CALL
76 051366 162716 000002      SUB    #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
77 051372 004736      JSR   PC,a(SP)+     ;;REPORT THE ERROR VIA USER
78 051374 162716 000010      SUB   #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
79 051400 000240      NOP
80                                     :REPORT ERROR IF RMMR2 NOT INITIALIZED
81 051402 013737 001376 001142 6$:      MOV     RMMR2I,$BDDAT  ;;CHECK RMMR2
82 051410 042737 140000 001142      BIC    #RQA!RQB,$BDDAT ;;CLEAR REQA, REQB
83 051416 012737 011777 001140      MOV    #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
84 051424 023737 001140 001142      CMP    $GDDAT,$BDDAT ;;COMPARE EXPECTED & RECEIVED
85 051432 001413      BEQ    7$          ;;BRANCH IF EQUAL
86 051434 062716 000004  ADD     #4,(SP)      ;;MOVE SP TO ERROR CALL
87 051440 112776 000146 000000  MOVB   #146,a(SP)    ;;WRITE NUMBER OF ERROR IN CALL
88 051446 162716 000002      SUB    #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
89 051452 004736      JSR   PC,a(SP)+     ;;REPORT THE ERROR VIA USER
90 051454 162716 000010      SUB   #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
91 051460 000240      NOP
92 051462 005037 001140 7$:      CLR    $GDDAT       ;;EXPECT ZEROS
93                                     :REPORT ERROR IF RMEC2 NOT RESET
94 051466 013737 001404 001142      MOV     RMEC2I,$BDDAT  ;;CHECK RMEC2
95 051474 001413      BEQ    8$          ;;BRANCH IF 0
96 051476 062716 000004  ADD     #4,(SP)      ;;MOVE SP TO ERROR CALL
97 051502 112776 000150 000000  MOVB   #150,a(SP)    ;;WRITE NUMBER OF ERROR IN CALL
98 051510 162716 000002      SUB    #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
99 051514 004736      JSR   PC,a(SP)+     ;;REPORT THE ERROR VIA USER
100 051516 162716 000010      SUB   #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
101 051522 000240      NOP
102                                     :REPORT ERROR IF RMER2 NOT RESET
103 051524 013737 001400 001142 8$:      MOV     RMER2I,$BDDAT  ;;CHECK RMER2
104 051532 001413      BEQ    9$          ;;BRANCH IF NO ERROR
105 051534 062716 000004  ADD     #4,(SP)      ;;MOVE SP TO ERROR CALL
106 051540 112776 000147 000000  MOVB   #147,a(SP)    ;;WRITE NUMBER OF ERROR IN CALL
107 051546 162716 000002      SUB    #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
108 051552 004736      JSR   PC,a(SP)+     ;;REPORT THE ERROR VIA USER
109 051554 162716 000010      SUB   #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
110 051560 000240      NOP
111 051562      9$:
112
113 051562      10$:
114

```

				:AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND	
115					
116	051562	062716	000004	ADD #4,(SP)	:MOVE SP TO ERROR CALL
117	051566	105776	000000	TSTB @ (SP)	:WAS AN ERROR DETECTED??
118	051572	001403		BEQ 11\$:NO!!
119	051574	062716	000004	ADD #4,(SP)	:YES - MOVE SP TO ERROR RETURN
120	051600	000402		BR 12\$	
121	051602	162716	000004	11\$: SUB #4,(SP)	:MOVE SP BACK TO NO ERROR RETURN
122	051606	000240		12\$: NOP	
123	051610	000207		RTS PC	:RETURN TO USER

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.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 49\$: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 1\$: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 4\$

1\$:

:REPORT ANY CONTROL BUS PARITY ERROR WHILE WRITTING REMOTE REGISTERS,
 :I.E., PAR = 1 AND DPE = 0

BIT #PAR,RMER1I :WAS THERE A PARITY ERROR??
 BEQ 3\$:NO!!
 BIT #DPE,RMER2I :DATA PARITY ERROR ?
 BNE 3\$:YES!!
 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,RMCS2I :DID MXF GET SET??
 BNE 2\$: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 4\$

2\$:

:LOOK FOR ANY ERRORS WHICH MAY HAVE OCCURRED DURING COMMAND INITIATION OR

051612
 051612 062716 000004
 051616 105076 000000
 051622 162716 000004
 051626 005037 055226
 051632 032737 020000 001336
 051640 001422
 051642 013737 001336 001140
 051650 042737 020000 001140
 051656 013737 001336 001142
 051664 062716 000004
 051670 112776 000013 000000
 051676 162716 000002
 051702 004736
 051704 000466
 051706
 051706 032737 000010 001352
 051714 001435
 051716 032737 000010 001400
 051724 001031
 051726 013737 001352 001140
 051734 042737 000010 001140
 051742 013737 001352 001142
 051750 062716 000004
 051754 112776 000050 000000
 051762 032737 001000 001346
 051770 001003
 051772 112776 000274 000000
 052000 162716 000002
 052004 004736
 052006 000425

```

58      :MECHANICAL POSITIONING
59
60      :FIRST TEST MXF WHICH WOULD INDICATE COMPOSITE ERROR SET WHEN FUNCTION
61      :CODE AND GO BIT WERE LOADED
62      052010      032737  001000  001346      3$:
63      052010      032737  001000  001346      BIT      #MXF,RMCS2I      :WAS 'MISSED TRANSFER' SET??
64      052016      001425      BEQ      5$      :NO!!
65      052020      013737  001346  001140      MOV      RMCS2I,$GDDAT      :EXPECTED STATUS
66      052026      042737  001000  001140      BIC      #MXF,$GDDAT
67      052034      013737  001346  001142      MOV      RMCS2I,$BDDAT      :RECEIVED STATUS
68      052042      062716  000004      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
69      052046      112776  000275  000000      MOV      #275,a(SP)      :WRITE ERROR NUMBER
70      052054      162716  000002      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
71      052060      004736      JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
72      052062
73
74      :RESTORE SP TO NO ERROR RETURN AND BYPASS FURHTER STATUS CHECKING
75      052062      162716  000010      SUB      #10,(SP)      :MOVE SP TO NO ERROR
76      052066      000137  055200      JMP      46$      :SKIP TO END OF SUB
77
78      5$:
79
80      :REPORT AN ERROR IF 'OPI' ERROR OCCURRED DUE TO 'MOL' = 0, OR IF 'OPI'
81      :AND 'MOL' ARE SET, BUT 'VV' IS RESET, INDICATING AN INTERMITTENT
82      :'MOL'
83      052072      032737  020000  001352      BIT      #OPI,RMER1I      :IS 'OPI' SET??
84      052100      001447      BEQ      8$      :NO!!
85      052102      013737  001352  001140      MOV      RMER1I,$GDDAT      :EXPECTED STATUS
86      052110      042737  020000  001140      BIC      #OPI,$GDDAT
87      052116      013737  001352  001142      MOV      RMER1I,$BDDAT      :RECEIVED STATUS
88      052124      032737  010000  001350      BIT      #MOL,RMDSI      :WAS MEDIUM OFF LINE??
89      052132      001404      BEQ      6$      :YES!!
90      052134      032737  000100  001350      BIT      #VV,RMDSI      :WAS 'MOL' INTERMITTENT??
91      052142      001013      BNE      7$      :NO!!
92      052144      062716  000004      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
93      052150      112776  000276  000000      MOV      #276,a(SP)      :WRITE ERROR NUMBER IN CALL
94      052156      162716  000002      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
95      052162      004736      JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
96      052164      162716  000010      SUB      #10,(SP)      :RESTORE SP TO NO ERROR
97      052170      000413
98      052172
99
100     :REPORT 'OPI' ERROR, WHICH IS DUE TO 'ON CYLINDER' NOT DROPPING OR
101     :'RUN' TIMEOUT (20 MS) OR SEARCH TIMEOUT (50 MS)
102     052172      062716  000004      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
103     052176      112776  000277  000000      MOV      #277,a(SP)      :WRITE ERROR NUMBER IN CALL
104     052204      162716  000002      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
105     052210      004736      JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
106     052212      162716  000010      SUB      #10,(SP)      :RESTORE SP TO NO ERROR
107     052216      000240
108     052220
109
110     :LOOK FOR 'IVC' ERROR DURING COMMAND INITIATION
111     052220      032737  010000  001400      BIT      #IVC,RMER2I      :WAS THERE AN 'IVC' ERROR??
112     052226      001432      BEQ      10$      :NO!!
113     :REPORT 'IVC' ERROR DUE TO 'VV' = 0, OR REPORT ERRONEOUS 'IVC' ERROR
114     052230      013737  001400  001140      MOV      RMER2I,$GDDAT      :EXPECTED STATUS
    
```

```

115 052236 042737 010000 001140      BIC      #IVC,$GDDAT
116 052244 013737 001400 001142      MOV      RMER2I,$BDDAT      ;RECEIVED STATUS
117 052252 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR
118 052256 112776 000300 000000      MOVVB   #300,a(SP)        ;WRITE ERROR NUMBER IN CALL
119 052264 032737 000100 001350      BIT      #VV,RMDSI        ;WAS VOLUME VALID??
120 052272 001403                BEQ      9$                ;NO!!
121 052274 112776 000301 000000      MOVVB   #301,a(SP)        ;CHANGE ERROR NUMBER
122 052302 162716 000002      9$:    SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
123 052306 004736                JSR      PC,a(SP)+        ;REPORT "IVC" ERROR AND RETURN
124 052310 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
125 052314
126
127      ;SEE IF "ILF" OR "RMR" IS SET
128 052314 032737 000007 001352      BIT      #ILR!ILF!RMR,RMER1I
129 052322 001510                BEQ      13$                ;NO ERRORS DETECTED
130      ;REPORT AN ERROR IF "ILR" IS SET
131 052324 032737 000002 001352      BIT      #ILR,RMER1I      ;WAS "ILR" DETECTED??
132 052332 001424                BEQ      11$                ;NO!!
133 052334 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
134 052342 042737 000002 001140      BIC      #ILR,$GDDAT
135 052350 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
136 052356 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
137 052362 112776 000302 000000      MOVVB   #302,a(SP)        ;WRITE ERROR NUMBER IN CALL
138 052370 162716 000002      SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
139 052374 004736                JSR      PC,a(SP)+        ;REPORT ERROR AND RETURN
140 052376 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
141 052402 000240
142 052404
143
144      ;REPORT AN ERROR IF "ILF" IS SET
145 052404 032737 000001 001352      BIT      #ILF,RMER1I      ;WAS "ILF" DETECTED??
146 052412 001424                BEQ      12$                ;NO!!
147 052414 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
148 052422 042737 000001 001140      BIC      #ILF,$GDDAT
149 052430 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
150 052436 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
151 052442 112776 000303 000000      MOVVB   #303,a(SP)        ;WRITE ERROR NUMBER IN CALL
152 052450 162716 000002      SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
153 052454 004736                JSR      PC,a(SP)+        ;REPORT ERORR AND RETURN
154 052456 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
155 052462 000240
156 052464
157
158      ;REPORT AN ERROR IF "RMR" IS SET
159 052464 032737 000004 001352      BIT      #RMR,RMER1I      ;WAS "RMR" DETECTED??
160 052472 001424                BEQ      13$                ;NO!!
161 052474 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
162 052502 042737 000004 001140      BIC      #RMR,$GDDAT
163 052510 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
164 052516 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
165 052522 112776 000304 000000      MOVVB   #304,a(SP)        ;WRITE ERROR NUMBER IN CALL
166 052530 162716 000002      SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
167 052534 004736                JSR      PC,a(SP)+        ;REPORT ERROR AND RETURN
168 052536 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
169 052542 000240
170 052544
171      13$:
      ;DETERMINE WHETHER OR NOT "IAE" SHOULD BE SET AND CHECK FOR ERROR
    
```

DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

```

172 052544 012737 002000 001140      MOV      #IAE,$GDDAT      :SETUP FOR "IAE" = 1
173 052552 052737 040000 055226      BIS      #SKI,49$        :SETUP FOR "SKI" = 1
174 052560 023727 001446 001060      CMP      RMDCO,#560.     :GREATER THAN LAST CYLINDER ?
175 052566 101035                BHI      15$             :YES - CYLINDER IS INVALID
176 052570 042737 040000 055226      BIC      #SKI,49$        :RESET SKI FLAG
177
178 052576 123737 001421 001335      CMPB     RMDAO+1,LSTRK+1 :GREATER THAN LAST TRACK ?
179 052604 101026                BHI      15$             :YES - TRACK IS INVALID
180
181 052606 123727 001420 000035      CMPB     RMDAO,#29.      :IS SECTOR > 29. ?
182 052614 101420                BLOS     14$            :NO
183 052616 032737 010000 001444      BIT      #FMT16,RMOFO    :18 BIT FORMAT ?
184 052624 001416                BEQ      15$            :YES - SECTOR IS INVALID FOR 18 BIT MODE
185 052626 123727 001420 000036      CMPB     RMDAO,#30.      :IS SECTOR > 30. ?
186 052634 101410                BLOS     14$            :NO
187 052636 032737 001000 001444      BIT      #SSEI,RMOFO     :IS SSEI CLEAR ?
188 052644 001406                BEQ      15$            :YES - SECTOR IS INVALID FOR 16 BIT MODE
189 052646 123727 001420 000037      CMPB     RMDAO,#31.      :IS SECTOR > 31. ?
190 052654 101002                BHI      15$            :YES - SECTOR IS INVALID
191 052656 005037 001140          14$:      CLR      $GDDAT          : "IAE" SHOULD = 0
192
193 052662 013737 001352 001142      15$:      MOV      RMER1I,$BDDAT    :GET RECEIVED STATUS
194 052670 042737 175777 001142      BIC      #^CIAE,$BDDAT   :
195 052676 023737 001140 001142      CMP      $GDDAT,$BDDAT   :IS "IAE" STATUS OK??
196 052704 001004                BNE      16$            :NO!!
197 052706 042737 040000 055226      BIC      #SKI,49$        :IAE OK - SKI SHOULD BE 0
198 052714 000412                BR
199 052716 062716 000004          16$:      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
200 052722 112776 000305 000000      MOVB     #305,a(SP)      :WRITE ERROR NUMBER
201 052730 162716 000002          SUB      #2,(SP)         :MOVE SP TO RETURN IF ERROR
202 052734 004736          JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
203 052736 162716 000010          SUB      #10,(SP)       :MOVE SP TO NO ERROR
204 052742          17$:
205
206          :REPORT AN ERROR IF "SKI" IS SET AND "IAE" STATUS WAS OK
207 052742 013737 001400 001142      MOV      RMER2I,$BDDAT   :RECEIVED STATUS
208 052750 042737 137777 001142      BIC      #^CSKI,$BDDAT   :
209 052756 013737 055226 001140      MOV      49$,$GDDAT      :EXPECTED STATUS
210 052764 042737 137777 001140      BIC      #^CSKI,$GDDAT   :
211 052772 032737 040000 001400      BIT      #SKI,RMER2I     :WAS "SKI" SET??
212 053000 001417                BEQ      18$            :NO!!
213 053002 032737 040000 055226      BIT      #SKI,49$        :WAS SKI CAUSED BY IAE = 0??
214 053010 001032                BNE      19$            :YES - DON'T REPORT SKI
215 053012 062716 000004          ADD      #4,(SP)         :MOVE SP TO USERS ERROR CALL
216 053016 112776 000306 000000      MOVB     #306,a(SP)      :WRITE ERROR NUMBER
217 053024 162716 000002          SUB      #2,(SP)         :MOVE SP TO RETURN IF ERROR
218 053030 004736          JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
219 053032 162716 000010          SUB      #10,(SP)       :MOVE SP TO NO ERROR
220 053036 000417                BR
221
222 053040          18$:
223
224          :REPORT AN ERROR IF SKI = 0 AND IAE WAS NOT DETECTED
225 053040 032737 040000 055226      BIT      #SKI,49$        :SHOULD SKI BE SET??
226 053046 001413                BEQ      19$            :NO!!
227 053050 062716 000004          ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
228 053054 112776 000307 000000      MOVB     #307,a(SP)      :WRITE ERROR NUMBER IN CALL

```

```

229 053062 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN IF ERROR
230 053066 004736              JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
231 053070 162716 000010      SUB      #10,(SP)   ;RESTORE SP TO NO ERROR
232 053074 000240              NOP
233 053076              19$:
234
235              ;LOOK FOR 'LSC' OR 'LBC' OR 'DVC' IN ERROR REGISTER #2
236 053076 032737 006200 001400  BIT      #LSC!LBC!DVC,RMER2I
237 053104 001512              BEQ      22$        ;NO ERRORS SET
238
239              ;REPORT ANY DEVICE FAULT, I.E., 'DVC' = 1
240 053106 032737 000200 001400  BIT      #DVC,RMER2I ;IS 'DVC' = 1??
241 053114 001424              BEQ      20$        ;NO!!
242 053116 013737 001400 001140  MOV      RMER2I,$GDDAT ;EXPECTED STATUS
243 053124 042737 000200 001140  BIC      #DVC,$GDDAT
244 053132 013737 001400 001142  MOV      RMER2I,$BDDAT ;RECEIVED STATUS
245 053140 062716 000004              ADD      #4,(SP)    ;MOVE SP TO USERS ERROR
246 053144 112776 000310 000000  MOVB    #310,@(SP)  ;WRITE ERROR NUMBER IN CALL
247 053152 162716 000002              SUB      #2,(SP)    ;MOVE SP TO RETURN IF ERROR
248 053156 004736              JSR      PC,@(SP)+  ;REPORT ERROR AND RETURN
249 053160 162716 000010      SUB      #10,(SP)   ;RESTORE SP TO NO ERROR
250 053164 000240              NOP
251 053166              20$:
252
253              ;REPORT LOSS OF BIT CLOCK, I.E.: 'LBC' = 1, IF 'MOL' = 1
254 053166 032737 002000 001400  BIT      #LBC,RMER2I ;IS LBC SET??
255 053174 001430              BEQ      21$        ;NO!!
256 053176 032737 010000 001350  BIT      #MOL,RMDSI  ;WAS LBC ERROR BY MOL = 0
257 053204 001424              BEQ      21$        ;YES!!
258 053206 013737 001400 001140  MOV      RMER2I,$GDDAT ;EXPECTED STATUS
259 053214 042737 002000 001140  BIC      #LBC,$GDDAT
260 053222 013737 001400 001142  MOV      RMER2I,$BDDAT ;RECEIVED STATUS
261 053230 062716 000004              ADD      #4,(SP)    ;MOVE SP TO USER'S ERROR CALL
262 053234 112776 000311 000000  MOVB    #311,@(SP)  ;WRITE ERROR NUMBER IN CALL
263 053242 162716 000002              SUB      #2,(SP)    ;MOVE SP TO RETURN IF ERROR
264 053246 004736              JSR      PC,@(SP)+  ;REPORT ERROR AND RETURN
265 053250 162716 000010      SUB      #10,(SP)   ;RESTORE SP TO NO ERROR
266 053254 000240              NOP
267 053256              21$:
268
269              ;REPORT LOS OF SYSTEM CLOCK, I.E., 'LSC' = 1
270 053256 032737 004000 001400  BIT      #LSC,RMER2I ;IS 'LSC' = 1??
271 053264 001422              BEQ      22$        ;NO!!
272 053266 013737 001400 001140  MOV      RMER2I,$GDDAT ;EXPECTED STATUS
273 053274 042737 004000 001140  BIC      #LSC,$GDDAT
274 053302 013737 001400 001142  MOV      RMER2I,$BDDAT ;RECEIVED STATUS
275 053310 062716 000004              ADD      #4,(SP)    ;MOVE SP TO USER'S ERROR CALL
276 053314 112776 000312 000000  MOVB    #312,@(SP)  ;WRITE ERROR NUMBER
277 053322 004736              JSR      PC,@(SP)+  ;REPORT ERROR AND RETURN
278 053324 162716 000010      SUB      #10,(SP)   ;RESTORE SP TO NO ERROR
279 053330 000240              NOP
280 053332              22$:
281
282              ;LOOK FOR 'UNS' OR 'DTE' OR 'WLE' IN ERROR REGISTER #1
283 053332 032737 054000 001352  BIT      #UNS!DTE!WLE,RMER1I
284 053340 001527              BEQ      27$        ;NO BITS SET
285              ;REPORT 'UNS' ERROR IF 'DVC' = 0
    
```

```

286 053342 032737 040000 001352      BIT      #UNS,RMER1I      ;IS 'UNS' SET??
287 053350 001427                BEQ      23$             ;NO!!
288 053352 032737 000200 001400      BIT      #DVC,RMER2I      ;WAS 'UNS' CAUSED BY 'DVC'??
289 053360 001023                BNE      23$             ;YES!!
290 053362 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
291 053370 042737 040000 001140      BIC      #UNS,$GDDAT
292 053376 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
293 053404 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
294 053410 112776 000313 000000      MOV      #313,a(SP)      ;WRITE ERROR NUMBER
295 053416 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
296 053422 004736                JSR      PC,a(SP)+        ;REPORT ERROR AND RETURN
297 053424 162716 000010                SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
298 053430
299
300                                     ;REPORT ANY DRIVE TIMING ERROR, I.E., 'DTE' = 1
301 053430 032737 010000 001352      BIT      #DTE,RMER1I      ;IS DTE SET??
302 053436 001423                BEQ      24$             ;NO!!
303 053440 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
304 053446 042737 010000 001140      BIC      #DTE,$GDDAT
305 053454 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
306 053462 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
307 053466 112776 000314 000000      MOV      #314,a(SP)      ;WRITE ERROR NUMBER IN CALL
308 053474 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
309 053500 004736                JSR      PC,a(SP)+        ;REPORT ERROR AND RETURN
310 053502 162716 000010                SUB      #10,(SP)         ;MOVE SP TO NO ERROR
311 053506
312
313                                     ;REPORT AN ERROR IF WRITE LOCK ERROR IS SET. SEE IF DRIVE IS NOT
314                                     ;WRITE PROTECTED, OR IF FUNCTION WAS NOT A WRITE
315 053506 032737 004000 001352      BIT      #WLE,RMER1I      ;WAS 'WLE' SET??
316 053514 001441                BEQ      27$             ;NO!!
317 053516 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
318 053524 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
319 053532 052737 004000 001140      BIS      #WLE,$GDDAT
320 053540 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
321 053544 112776 000315 000000      MOV      #315,a(SP)      ;WRITE ERROR NUMBER IN CALL
322 053552 032737 004000 001350      BIT      #WRL,RMDSI      ;WAS DRIVE WRITE PROTECTED??
323 053560 001404                BEQ      25$             ;NO!!
324 053562 032737 000010 001412      BIT      #BIT3,RMCS10     ;WAS COMMAND A WRITE??
325 053570 001406                BEQ      26$             ;YES!!
326 053572 112776 000316 000000 25$:  MOV      #316,a(SP)      ;CHANGE ERROR NUMBER
327 053600 042737 004000 001140      BIC      #WLE,$GDDAT
328 053606 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
329 053612 004736                JSR      PC,a(SP)+        ;REPORT ERROR AND RETURN
330 053614 162716 000010                SUB      #10,(SP)         ;MOVE SP TO NO ERROR
331
332 053620                                     27$:
333
334                                     ;OMIT DATA ERROR CHECKS IF ANY PREVIOUS ERRORS HAVE BEEN DETECTED
335 053620 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR
336 053624 105716 000000                TST      a(SP)            ;WAS ERROR DETECTED??
337 053630 001404                BEQ      28$             ;NO - DO DATA CHECKS
338 053632 162716 000004                SUB      #4,(SP)          ;RESTORE SP
339 053636 000137 054640                JMP      42$             ;SKIP DATA CHECKS
340 053642 162716 000004                SUB      #4,(SP)          ;RESTORE SP
341
342                                     ;CHECK HEADER ERRORS IF FUNCTION WAS NOT WRITE HEADER AND DATA, AND
    
```

```

343                                     :IF HEADER COMPARE IS NOT INHIBITED
344 053646 013737 001412 055230      MOV      RMCS10,50$      :STRIP AND STORE FUNCTION CODE
345 053654 042737 177700 055230      BIC      #^CFNCMSK,50$
346 053662 022737 000063 055230      CMP      #WH!GO,50$    :WAS FUNCTION WRITE HEADER & DATA??
347 053670 001512                    BEQ      31$           :YES - SKIP HEADER CHECKS
348 053672 032737 002000 001370      BIT      #HCI,RMOFI    :WAS HCI SET??
349 053700 001106                    BNE      31$           :YES - SKIP HEADER CHECKS
350
351                                     :SEE IF ANY HEADER ERRORS ARE SET, I.E., 'FER' OR 'HCRC' OR 'HCE''
352 053702 032737 000620 001352      BIT      #HCRC!FER!HCE,RMER1I
353 053710 001533                    BEQ      33$           :NO ERRORS SET
354
355                                     :REPORT HEADER CRC ERROR IF SET
356 053712 032737 000400 001352      BIT      #HCRC,RMER1I  :WAS HCRC SET??
357 053720 001422                    BEQ      29$           :NO!!
358 053722 013737 001352 001140      MOV      RMER1I,$GDDAT :EXPECTED STATUS
359 053730 042737 000400 001140      BIC      #HCRC,$GDDAT
360 053736 013737 001352 001142      MOV      RMER1I,$BDDAT :RECEIVED STATUS
361 053744 062716 000004                    ADD      #4,(SP)       :MOVE SP TO USERS ERROR
362 053750 112776 000317 000000      MOV      #317,a(SP)   :WRITE ERROR NUMBER
363 053756 162716 000002                    SUB      #2,(SP)       :MOVE SP TO RETURN IF ERROR
364 053762 004736                    JSR      PC,a(SP)+     :REPORT ERROR AND RETURN
365 053764 000501                    BR
366 053766                    29$:
367
368                                     :REPORT FORMAT ERROR IF SET
369 053766 032737 000020 001352      BIT      #FER,RMER1I  :WAS 'FER' SET??
370 053774 001422                    BEQ      30$           :NO!!
371 053776 013737 001352 001140      MOV      RMER1I,$GDDAT :EXPECTED STATUS
372 054004 042737 000020 001140      BIC      #FER,$GDDAT
373 054012 013737 001352 001142      MOV      RMER1I,$BDDAT :RECEIVED STATUS
374 054020 062716 000004                    ADD      #4,(SP)       :MOVE SP TO USERS ERROR
375 054024 112776 000320 000000      MOV      #320,a(SP)   :WRITE ERROR NUMBER
376 054032 162716 000002                    SUB      #2,(SP)       :MOVE SP TO RETURN IF ERROR
377 054036 004736                    JSR      PC,a(SP)+     :REPORT ERROR AND RETURN
378 054040 000453                    BR
379 054042                    30$:
380
381                                     :REPORT HEADER COMPARE ERROR IF SET
382 054042 032737 000200 001352      BIT      #HCE,RMER1I  :WAS 'HCE' SET??
383 054050 001453                    BEQ      33$           :NO!!
384 054052 013737 001352 001140      MOV      RMER1I,$GDDAT :EXPECTED STATUS
385 054060 042737 000200 001140      BIC      #HCE,$GDDAT
386 054066 013737 001352 001142      MOV      RMER1I,$BDDAT :RECEIVED STATUS
387 054074 062716 000004                    ADD      #4,(SP)       :MOVE SP TO USER'S ERROR
388 054100 112776 000321 000000      MOV      #321,a(SP)   :WRITE ERROR NUMBER
389 054106 162716 000002                    SUB      #2,(SP)       :MOVE SP TO RETURN IF ERROR
390 054112 004736                    JSR      PC,a(SP)+     :REPORT ERROR AND RETURN
391 054114 000425                    BR
392
393                                     :THERE SHOULD BE NO HEADER ERRORS BECAUSE
394                                     : .COMMAND WAS WRITE HEADER AND DATA, OR
395                                     : .HEADER COMPARE INHIBIT WAS SET
396 054116 032737 000620 001352      BIT      #HCE!FER!HCRC,RMER1I
397 054124 001425                    BEQ      33$           :NO ERRORS WERE SET
398 054126 013737 001352 001140      MOV      RMER1I,$GDDAT :EXPECTED STATUS
399 054134 042737 000620 001140      BIC      #HCE!FER!HCRC,$GDDAT
399 054142 013737 001352 001142      MOV      RMER1I,$BDDAT :RECEIVED STATUS
    
```

```

400 054150 062716 000004          ADD    #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
401 054154 112776 000322 000000  MOVB   #322,@(SP)      ;WRITE ERROR NUMBER
402 054162 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN IF ERROR
403 054166 004736          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
404 054170 162716 000010 32$:  SUB    #10,(SP)        ;MOVE SP TO NO ERROR
405 054174 000137 054640          JMP    42$             ;OMIT FURTHER DATA CHECKS
406
407 054200          33$:
408
409          ;IF COMMAND WAS A WRITE COMMAND, GO DO WRITE ERROR CHECKS, OTHERWISE
410          ;DO READ ERROR CHECKS
411 054200 032737 000010 055230      BIT    #BIT3,50$       ;WAS THIS A WRITE COMMAND?
412 054206 001002          BNE    34$             ;NO!!
413 054210 000137 054426          JMP    39$             ;GO DO WRITE STATUS CHECK
414 054214
415
416          ;REPORT DATA CHECK IF SET
417 054214 032737 100000 001352      BIT    #DCK,RMER1I     ;DATA CHECK ERROR??
418 054222 001450          BEQ    37$             ;NO!!
419 054224 013737 001352 001140      MOV    RMER1I,$GDDAT   ;EXPECTED STATUS
420 054232 042737 100000 001140      BIC    #DCK,$GDDAT
421 054240 013737 001352 001142      MOV    RMER1I,$BDDAT   ;RECEIVED STATUS
422 054246 062716 000004          ADD    #4,(SP)          ;MOVE SP TO USER'S ERROR
423 054252 112776 000323 000000  MOVB   #323,@(SP)      ;WRITE ERROR NUMBER
424 054260 032737 004000 001370      BIT    #EC1,RMOF1     ;WAS ECC CORRECTION DISABLED??
425 054266 001021          BNE    36$             ;YES!!
426 054270 112776 000324 000000  MOVB   #324,@(SP)      ;CHANGE TO RECOVERABLE ERROR
427 054276 032737 000100 001352      BIT    #ECH,RMER1I     ;IS ERROR RECOVERABLE??
428 054304 001007          BNE    35$             ;NO!!
429          ;DO NOT REPORT RECOVERABLE ERROR IF READ COMMAND
430 054306 032737 000020 055230      BIT    #BIT4,50$       ;WAS THIS A READ COMMAND ??
431 054314 001406          BEQ    36$             ;NO!!
432 054316 162716 000004          SUB    #4,(SP)          ;RESTORE SP
433 054322 000410          BR    37$             ;SKIP ERROR - DATA WILL BE CORRECTED
434 054324 112776 000325 000000 35$:  MOVB   #325,@(SP)      ;CHANGE TO NON RECOVERABLE
435 054332 162716 000002 36$:  SUB    #2,(SP)          ;MOVE SP TO RETURN IF ERROR
436 054336 004736          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
437 054340 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
438
439 054344          37$:
440
441          ;REPORT DATA BUS PARITY ERROR IF SET, I.E., MDPE = 1
442 054344 032737 000400 001346      BIT    #MDPE,RMCS2I    ;PARITY ERROR SET??
443 054352 001423          BEQ    38$             ;NO!!
444 054354 013737 001346 001140      MOV    RMCS2I,$GDDAT   ;EXPECTED STATUS
445 054362 042737 000400 001140      BIC    #MDPE,$GDDAT
446 054370 013737 001346 001142      MOV    RMCS2I,$BDDAT   ;RECEIVED STATUS
447 054376 062716 000004          ADD    #4,(SP)          ;MOVE SP TO USER'S ERROR
448 054402 112776 000326 000000  MOVB   #326,@(SP)      ;WRITE ERROR NUMBER
449 054410 162716 000002          SUB    #2,(SP)          ;MOVE SP TO RETURN IF ERROR
450 054414 004736          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
451 054416 162716 000010          SUB    #10,(SP)        ;MOVE SP TO NO ERROR
452 054422 000137 054640 38$:  JMP    42$             ;SKIP WRITE STATUS CHECK
453
454 054426          39$:
455
456          ;TEST TO SEE THAT OFFSET MODE WAS RESET; REPORT ERROR IF 'OM' = 1
    
```



```

457 054426 032737 000001 001350      BIT      #OM,RMDSI      :IS OFFSET ON??
458 054434 001423                      BEQ      40$         :NO
459 054436 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
460 054444 042737 000001 001140      BIC      #OM,$GDDAT
461 054452 013737 001350 001142      MOV      RMDSI,$BDDAT :RECEIVED STATUS
462 054460 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
463 054464 112776 000327 000000      MOVVB   #327,a(SP)   :WRITE ERROR NUMBER IN CALL
464 054472 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
465 054476 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
466 054500 162716 000010                      SUB      #10,(SP)     :MOVE SP TO NO ERROR
467 054504
468
469                                     ;TEST FOR DATA BUS PARITY ERROR; REPORT ERROR IF 'DPE' = 1
470 054504 032737 000010 001400      BIT      #DPE,RMER2I  :DATA PARITY ERROR??
471 054512 001423                      BEQ      41$         :NO!!
472 054514 013737 001400 001140      MOV      RMER2I,$GDDAT :EXPECTED STATUS
473 054522 042737 000010 001140      BIC      #DPE,$GDDAT
474 054530 013737 001400 001142      MOV      RMER2I,$BDDAT :RECEIVED STATUS
475 054536 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
476 054542 112776 000330 000000      MOVVB   #330,a(SP)   :WRITE ERROR NUMBER
477 054550 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
478 054554 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
479 054556 162716 000010                      SUB      #10,(SP)     :MOVE SP TO NO ERROR
480 054552
481
482                                     ;TEST FOR WRITE CLOCK FAILURE; REPORT ERROR IF 'WCF' = 1
483 054562 032737 000040 001352      BIT      #WCF,RMER1I  :IS 'WCF' SET??
484 054570 001423                      BEQ      42$         :NO!!
485 054572 013737 001352 001140      MOV      RMER1I,$GDDAT :EXPECTED STATUS
486 054600 042737 000040 001140      BIC      #WCF,$GDDAT
487 054606 013737 001352 001142      MOV      RMER1I,$BDDAT :RECEIVED STATUS
488 054614 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
489 054620 112776 000331 000000      MOVVB   #331,a(SP)   :WRITE ERROR NUMBER
490 054626 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
491 054632 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
492 054634 162716 000010                      SUB      #10,(SP)     :MOVE SP TO NO ERROR
493 054640
494
495                                     ;REPORT 'DATA LATE' ERROR IF 'DLT' = 1
496 054640 032737 100000 001346      BIT      #DLT,RMCS2I  :IS 'DLT' SET??
497 054646 001423                      BEQ      43$         :NO!!
498 054650 013737 001346 001140      MOV      RMCS2I,$GDDAT :EXPECTED STATUS
499 054656 042737 100000 001140      BIC      #DLT,$GDDAT
500 054664 013737 001346 001142      MOV      RMCS2I,$BDDAT :RECEIVED STATUS
501 054672 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
502 054676 112776 000332 000000      MOVVB   #332,a(SP)   :WRITE ERROR NUMBER
503 054704 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
504 054710 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
505 054712 162716 000010                      SUB      #10,(SP)     :MOVE SP TO NO ERROR
506 054716
507                                     ;LOOK FOR UNEXPECTED CHANGES IN DRIVE STATUS
508 054716 013746 001350                      MOV      RMDSI,-(SP)  :STACK DRIVE STATUS
509 054722 042716 147677                      BIC      #^C<PIP!MOL!VV>,(SP) :CLEAR DONT CARES
510 054726 022726 010100                      CMP      #MOL!VV,(SP)+ :IS DRIVE STATUS OK??
511 054732 001522                      BEQ      46$         :YES!!
512
513                                     ;REPORT ERROR IF POSITIONING IN PROGRESS AND NO SEEK INCOMPLETE ERROR,
    
```

```

514                                     :I.E. PIP = 1 AND SKI = 0
515 054734 032737 020000 001350      BIT      #PIP,RMDSI      :IS 'PIP' SET??
516 054742 001430                    BEQ      44$          :NO!!
517 054744 032737 040000 001400      BIT      #SKI,RMER2I  :WAS 'SKI' ERROR REPORTED??
518 054752 001024                    BNE      44$          :YES-DONT REPORT PIP
519 054754 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
520 054762 042737 020000 001140      BIC      #PIP,$GDDAT
521 054770 013737 001350 001142      MOV      RMDSI,$BDDAT
522 054776 062716 000004                    ADD      #4,(SP)      :RECEIVED STATUS
523 055002 112776 000333 000000      MOV      #333,a(SP)  :MOVE SP TO USERS ERROR CALL
524 055010 162716 000002                    SUB      #2,(SP)      :WRITE ERROR NUMBER
525 055014 004736                    JSR      PC,a(SP)+    :MOVE SP TO RETURN IF ERROR
526 055016 162716 000010                    SUB      #10,(SP)    :REPORT ERROR AND RETURN
527 055022 000240                    NOP                       :MOVE SP TO NO ERROR
528 055024                                     44$:
529
530                                     :REPORT ERROR IF MEDIUM IS NOT ON LINE AND OPI ERROR WAS NOT
531                                     :REPORTED, I.E., MOL = OPI = 0
532 055024 032737 010000 001350      BIT      #MOL,RMDSI  :IS MEDIUM ON LINE??
533 055032 001027                    BNE      45$          :YES!!
534 055034 032737 020000 001352      BIT      #OPI,RMER1I :WAS OPI ERROR REPORTED??
535 055042 001023                    BNE      45$          :YES!!
536 055044 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
537 055052 052737 010000 001140      BIS      #MOL,$GDDAT
538 055060 013737 001350 001142      MOV      RMDSI,$BDDAT
539 055066 062716 000004                    ADD      #4,(SP)      :RECEIVED STATUS
540 055072 112776 000334 000000      MOV      #334,a(SP)  :MOVE SP TO USER'S ERROR
541 055100 162716 000002                    SUB      #2,(SP)      :WRITE ERROR NUMBER
542 055104 004736                    JSR      PC,a(SP)+    :MOVE SP TO RETURN IF ERROR
543 055106 162716 000010                    SUB      #10,(SP)    :REPORT ERROR AND RETURN
544 055112                                     45$:
545
546                                     :REPORT ERROR IF VOLUME IS NOT VALID AND 'IVC' ERROR WAS NOT
547                                     :REPORTED, I.E., VV = IVC = 0
548 055112 032737 000100 001350      BIT      #VV,RMDSI   :IS VOLUME VALID??
549 055120 001027                    BNE      46$          :YES!!
550 055122 032737 010000 001400      BIT      #IVC,RMER2I :WAS IVC ERROR REPORTED??
551 055130 001033                    BNE      47$          :YES!!
552 055132 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
553 055140 052737 000100 001140      BIS      #VV,$GDDAT
554 055146 013737 001350 001142      MOV      RMDSI,$BDDAT
555 055154 062716 000004                    ADD      #4,(SP)      :RECEIVED STATUS
556 055160 112776 000335 000000      MOV      #335,a(SP)  :MOVE SP TO USERS ERROR CALL
557 055166 162716 000002                    SUB      #2,(SP)      :WRITE ERROR NUMBER
558 055172 004736                    JSR      PC,a(SP)+    :MOVE SP TO RETURN IF ERROR
559 055174 162716 000010                    SUB      #10,(SP)    :REPORT ERROR AND RETURN
560 055200                                     46$:
561
562                                     :AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS FOUND
563 055200 062716 000004                    ADD      #4,(SP)      :MOVE SP TO ERROR CALL
564 055204 105776 000000                    TSTB     a(SP)        :ANY ERROR??
565 055210 001403                    BEQ      47$          :NO!!
566 055212 062716 000004                    ADD      #4,(SP)      :YES - MOVE SP TO ERROR RETURN
567 055216 000402                    BR       48$
568 055220 162716 000004      47$: SUB      #4,(SP)    :MOVE SP TO NO ERROR RETURN
569
570 055224 000207      48$: RTS      PC      :RETURN TO USER
    
```

571
572 055226 000000
573 055230 000000

49S: .WORD
50S: .WORD

:ERROR FLAGS
:TEMPORARY STORAGE

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.SBTTL STATIC DRIVE STATUS CHECK SUBROUTINE

:THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
 :STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID. THE SUBROUTINE
 :CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
 :SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.

:THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
 :IF TRUE:

: .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
 : THAT MOL IS ASSUMED TO HAVE BEEN SET
 : .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
 : .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
 : .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
 : .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT

:THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.

```
:(1) JSR PC,STCDRVSTS
      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
```

STCDRVSTS:

```
:CLEAR USER'S ERROR CALL
  ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
  CLRB @(SP) ;CLEAR ERROR NUMBER
  SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
:SEE IF 'MOL' = 'VV' = 1, AND 'PIP' = 0
  MOV RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
  BIC #^C<PIP!MOL!VV>,(SP)
  CMP #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
  BEQ 3$ ;YES!!
```

```
:REPORT AN ERROR IF MOL = 0 AND 'OPI' = 0
  BIT #MOL,RMDSI ;IS MOL ON ??
  BNE 1$ ;YES!!
  BIT #OPI,RMER1I ;WAS 'OPI' SET??
  BNE 1$ ;YES-DONT 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 #207,@(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 NO ERROR RETURN
  NOP
```

1\$:

```
:REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND 'IVC' = 0
  BIT #VV,RMDSI ;IS 'VV' = 0??
  BNE 2$ ;NO!!
```

```
27 055232
30 055232 062716 000004
31 055236 105076 000000
32 055242 162716 000004
34 055246 013746 001350
35 055252 042716 147677
36 055256 022726 010100
37 055262 001524
40 055264 032737 010000 001350
41 055272 001030
42 055274 032737 020000 001352
43 055302 001024
44 055304 013737 001350 001140
45 055312 052737 010000 001140
46 055320 013737 001350 001142
47 055326 062716 000004
48 055332 112776 000207 000000
49 055340 162716 000002
50 055344 004736
51 055346 162716 000010
52 055352 000240
53 055354
56 055354 032737 000100 001350
57 055362 001030
```

```

58 055364 032737 010000 001400      BIT      #IVC,RMER2I      :WAS "IVC" SET??
59 055372 001024                      BNE      2$              :YES-DONT REPORT "VV" = 0
60 055374 013737 001350 001140      MOV      RMDSI,$GDDAT    :EXPECTED STATUS
61 055402 052737 000100 001350      BIS      #VV,RMDSI
62 055410 013737 001350 001142      MOV      RMDSI,$BDDAT    :RECEIVED STATUS
63 055416 062716 000004                      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
64 055422 112776 000210 000000      MOVB    #210,a(SP)      :WRITE ERROR NUMBER IN CALL
65 055430 162716 000002                      SUB      #2,(SP)         :MOVE SP TO RETURN FOR ERROR
66 055434 004736                      JSR      PC,a(SP)+       :REPORT ERROR VIA USER
67 055436 162716 000010                      SUB      #10,(SP)       :MOVE SP BACK TO NO ERROR
68 055442 000240                      NOP
69 055444                                2$:
70
71                                ;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND "SKI" = 0
72 055444 032737 020000 001350      BIT      #PIP,RMDSI      :IS DRIVE OFF CYLINDER??
73 055452 001430                      BEQ      3$              :NO!!
74 055454 032737 040000 001400      BIT      #SKI,RMER2I     :WAS "SKI" SET??
75 055462 001024                      BNE      3$              :YES-DONT REPORT "PIP" = 1
76 055464 013737 001350 001140      MOV      RMDSI,$GDDAT    :EXPECTED STATUS
77 055472 042737 020000 001140      BIC      #PIP,$GDDAT
78 055500 013737 001350 001142      MOV      RMDSI,$BDDAT    :RECEIVED STATUS
79 055506 062716 000004                      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
80 055512 112776 000211 000000      MOVB    #211,a(SP)      :WRITE ERROR NUMBER IN USER'S CALL
81 055520 162716 000002                      SUB      #2,(SP)         :MOVE SP TO RETURN FOR ERROR
82 055524 004736                      JSR      PC,a(SP)+       :REPORT ERROR VIA USER
83 055526 162716 000010                      SUB      #10,(SP)       :MOVE SP TO NO ERROR RETURN
84 055532 000240                      NOP
85 055534                                3$:
86
87                                ;SEE IF "SKI" = "DVC" = 0
88 055534 013746 001400                      MOV      RMER2I,-(SP)     :PUT ERROR REG 2 ON STACK
89 055540 042726 137577                      BIC      #^C<SKI!DVC>,(SP)+
90 055544 001460                      BEQ      6$              :BRANCH IF NO ERROR
91 055546                                4$:
92
93                                ;REPORT AN ERROR IF THERE IS A DEVICE FAULT
94 055546 032737 000200 001400      BIT      #DVC,RMER2I     :ANY DEVICE FAULT??
95 055554 001424                      BEQ      5$              :NO!!
96 055556 013737 001400 001140      MOV      RMER2I,$GDDAT    :EXPECTED STATUS
97 055564 042737 000200 001140      BIC      #DVC,$GDDAT
98 055572 013737 001400 001142      MOV      RMER2I,$BDDAT    :RECEIVED STATUS
99 055600 062716 000004                      ADD      #4,(SP)         :MOVE SP TO USER'S CALL
100 055604 112776 000212 000000      MOVB    #212,a(SP)      :WRITE NUMBER OF ERROR IN CALL
101 055612 162716 000002                      SUB      #2,(SP)         :MOVE SP TO RETURN FOR ERROR
102 055616 004736                      JSR      PC,a(SP)+       :REPORT ERROR VIA USER
103 055620 162716 000010                      SUB      #10,(SP)       :MOVE SP BACK TO NO ERROR
104 055624 000240                      NOP
105 055626                                5$:
106
107                                ;REPORT AN ERROR IF "SKI" = 1
108 055626 032737 040000 001400      BIT      #SKI,RMER2I     :IS THERE A SEEK INCOMPLETE ERROR
109 055634 001424                      BEQ      6$              :NO!!
110 055636 013737 001400 001140      MOV      RMER2I,$GDDAT    :EXPECTED STATUS
111 055644 042737 040000 001140      BIC      #SKI,$GDDAT
112 055652 013737 001400 001142      MOV      RMER2I,$BDDAT    :RECEIVED STATUS
113 055660 062716 000004                      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
114 055664 112776 000213 000000      MOVB    #213,a(SP)      :WRITE ERROR NUMBER IN USER'S ERROR CALL
    
```

```
115 055672 162716 000002          SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
116 055676 004736                JSR    PC,@(SP)+   ;REPORT ERROR VIA USER
117 055700 162716 000010          SUB    #10,(SP)   ;MOVE SP BACK TO NO ERROR
118 055704 000240                NOP
119 055706                        6$:
120
121                                ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
122 055706 062716 000004          ADD    #4,(SP)    ;MOVE SP TO USER'S ERROR CALL
123 055712 105776 000000          TSTB  @(SP)       ;WAS AN ERROR DETECTED??
124 055716 001403                BEQ    7$         ;NO!!
125 055720 062716 000004          ADD    #4,(SP)   ;YES - MOVE SP TO USER'S ERROR RETURN
126 055724 000402                BR     8$
127 055726 162716 000004          7$: SUB    #4,(SP)   ;NO - MOVE SP TO NO ERROR RETURN
128 055732 000240                8$: NOP
129 055734 000207                RTS    PC         ;RETURN TO USER
```

1

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

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

```
055736  
055736 010046  
055740 010146  
055742 010246  
055744 010346  
055746 010446  
055750 010546  
055752 016646 000022  
055756 016646 000022  
055762 016646 000022  
055766 016646 000022  
055772 000002  
$SAVREG:  
MOV R0,-(SP) ::PUSH R0 ON STACK  
MOV R1,-(SP) ::PUSH R1 ON STACK  
MOV R2,-(SP) ::PUSH R2 ON STACK  
MOV R3,-(SP) ::PUSH R3 ON STACK  
MOV R4,-(SP) ::PUSH R4 ON STACK  
MOV R5,-(SP) ::PUSH R5 ON STACK  
MOV 22(SP),-(SP) ::SAVE PS OF MAIN FLOW  
MOV 22(SP),-(SP) ::SAVE PC OF MAIN FLOW  
MOV 22(SP),-(SP) ::SAVE PS OF CALL  
MOV 22(SP),-(SP) ::SAVE PC OF CALL  
RTI
```

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

.SBTTL 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

056032	010146			\$TYPBN:	MOV	R1,-(SP)	::SAVE R1 ON THE STACK
056034	016601	000006			MOV	6(SP),R1	::GET THE INPUT NUMBER
056040	000261				SEC		::SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
056042	112737	000060	056104	1\$:	MOVB	#'0,\$BIN	::SET CHARACTER TO AN ASCII '0'.
056050	006101				ROL	R1	::GET THIS BIT
056052	001406				BEQ	2\$::DONE?
056054	105537	056104			ADCB	\$BIN	::NO--SET THE CHARACTER EQUAL TO THIS BIT
056060	104401	056104			TYPE	,\$BIN	::GO TYPE THIS BIT
056064	000241				CLC		::CLEAR 'C' SO CAN KEEP TRACK OF BITS
056066	000765				BR	1\$::GO DO THE NEXT BIT
056070	012601			2\$:	MOV	(SP)+,R1	::POP THE STACK INTO R1
056072	016666	000002	000004		MOV	2(SP),4(SP)	::ADJUST THE STACK
056100	012616				MOV	(SP)+,(SP)	
056102	000002				RTI		::RETURN TO USER
056104	000	000		\$BIN:	.BYTE	0,0	::STORAGE FOR ASCII CHAR. AND TERMINATOR

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 *REPLACED WITH SPACES.
 *CALL:

* MOV NUM,-(SP) ;:PUT THE BINARY NUMBER ON THE STACK
 * TYPDS ;:GO TO THE ROUTINE

\$TYPDS:

056106					MOV	R0,-(SP)	:::PUSH R0 ON STACK
056106	010046				MOV	R1,-(SP)	:::PUSH R1 ON STACK
056110	010146				MOV	R2,-(SP)	:::PUSH R2 ON STACK
056112	010246				MOV	R3,-(SP)	:::PUSH R3 ON STACK
056114	010346				MOV	R5,-(SP)	:::PUSH R5 ON STACK
056116	010546				MOV	#20200,-(SP)	:::SET BLANK SWITCH AND SIGN
056120	012746	020200			MOV	20(SP),R5	:::GET THE INPUT NUMBER
056124	016605	000020			BPL	1\$:::BR IF INPUT IS POS.
056130	100004				NEG	R5	:::MAKE THE BINARY NUMBER POS.
056132	005405				MOVB	#'-,1(SP)	:::MAKE THE ASCII NUMBER NEG.
056134	112766	000055	000001		CLR	R0	:::ZERO THE CONSTANTS INDEX
056142	005000			1\$:	MOV	#SDBLK,R3	:::SETUP THE OUTPUT POINTER
056144	012703	056322			MOVB	#',(R3)+	:::SET THE FIRST CHARACTER TO A BLANK
056150	112723	000040			CLR	R2	:::CLEAR THE BCD NUMBER
056154	005002			2\$:	MOV	SDBLK(R0),R1	:::GET THE CONSTANT
056156	016001	056312			SUB	R1,R5	:::FORM THIS BCD DIGIT
056162	160105			3\$:	BLT	4\$:::BR IF DONE
056164	002402				INC	R2	:::INCREASE THE BCD DIGIT BY 1
056166	005202				BR	3\$	
056170	000774				ADD	R1,R5	:::ADD BACK THE CONSTANT
056172	060105			4\$:	TST	R2	:::CHECK IF BCD DIGIT=0
056174	005702				BNE	5\$:::FALL THROUGH IF 0
056176	001002				TSTB	(SP)	:::STILL DOING LEADING 0'S?
056200	105716				BMI	7\$:::BR IF YES
056202	100407				ASLB	(SP)	:::MSD?
056204	106316			5\$:	BCC	6\$:::BR IF NO
056206	103003				MOVB	1(SP),-1(R3)	:::YES--SET THE SIGN
056210	116663	000001	177777		BIS	#'0,R2	:::MAKE THE BCD DIGIT ASCII
056216	052702	000060		6\$:	BIS	#',R2	:::MAKE IT A SPACE IF NOT ALREADY A DIGIT
056222	052702	000040		7\$:	MOVB	R2,(R3)+	:::PUT THIS CHARACTER IN THE OUTPUT BUFFER
056226	110223				TST	(R0)+	:::JUST INCREMENTING
056230	005720				CMP	R0,#10	:::CHECK THE TABLE INDEX
056232	020027	000010			BLT	2\$:::GO DO THE NEXT DIGIT
056236	002746				BGT	8\$:::GO TO EXIT
056240	003002				MOV	R5,R2	:::GET THE LSD
056242	010502				BR	6\$:::GO CHANGE TO ASCII
056244	000764				TSTB	(SP)+	:::WAS THE LSD THE FIRST NON-ZERO?
056246	105726			8\$:	BPL	9\$:::BR IF NO
056250	100003				MOVB	-1(SP),-2(R3)	:::YES--SET THE SIGN FOR TYPING
056252	116663	177777	177776		CLRB	(R3)	:::SET THE TERMINATOR
056260	105013			9\$:	MOV	(SP)+,R5	:::POP STACK INTO R5
056262	012605				MOV	(SP)+,R3	:::POP STACK INTO R3
056264	012603				MOV	(SP)+,R2	:::POP STACK INTO R2
056266	012602				MOV	(SP)+,R1	:::POP STACK INTO R1
056270	012601						

056272	012600			MOV	(SP)+,R0	::POP STACK INTO R0
056274	104401	056322		TYPE	\$DBLK	::NOW TYPE THE NUMBER
056300	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
056306	012616			MOV	(SP)+,(SP)	
056310	000002			RTI		::RETURN TO USER
056312	023420		\$DTBL:	10000.		
056314	001750			1000.		
056316	000144			100.		
056320	000012			10.		
056322			\$DBLK:	.BLKW	4	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
*      TYPOS    ::CALL FOR TYPEOUT
*      .BYTE   N              ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ::M=1 OR 0
*                               ::1=TYPE LEADING ZEROS
*                               ::0=SUPPRESS LEADING ZEROS

```

```

*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC

```

```

*CALL:
*      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
*      TYPON    ::CALL FOR TYPEOUT

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
*      TYPOC    ::CALL FOR TYPEOUT

```

056332	017646	000000		\$TYPOS:	MOV	@(SP),-(SP)	::PICKUP THE MODE
056336	116637	000001	056555		MOVB	1(SP), \$OFILL	::LOAD ZERO FILL SWITCH
056344	112637	056557			MOVB	(SP)+, \$OMODE+1	::NUMBER OF DIGITS TO TYPE
056350	062716	000002			ADD	#2, (SP)	::ADJUST RETURN ADDRESS
056354	000406				BR	\$TYPON	
056356	112737	000001	056555	\$TYPOC:	MOVB	#1, \$OFILL	::SET THE ZERO FILL SWITCH
056364	112737	000006	056557		MOVB	#6, \$OMODE+1	::SET FOR SIX(6) DIGITS
056372	112737	000005	056554	\$TYPON:	MOVB	#5, \$OCNT	::SET THE ITERATION COUNT
056400	010346				MOV	R3, -(SP)	::SAVE R3
056402	010446				MOV	R4, -(SP)	::SAVE R4
056404	010546				MOV	R5, -(SP)	::SAVE R5
056406	113704	056557			MOVB	\$OMODE+1, R4	::GET THE NUMBER OF DIGITS TO TYPE
056412	005404				NEG	R4	
056414	062704	000006			ADD	#6, R4	::SUBTRACT IT FOR MAX. ALLOWED
056420	110437	056556			MOVB	R4, \$OMODE	::SAVE IT FOR USE
056424	113704	056555			MOVB	\$OFILL, R4	::GET THE ZERO FILL SWITCH
056430	016605	000012			MOV	12(SP), R5	::PICKUP THE INPUT NUMBER
056434	005003				CLR	R3	::CLEAR THE OUTPUT WORD
056436	006105			1\$:	ROL	R5	::ROTATE MSB INTO 'C'
056440	000404				BR	3\$::GO DO MSB
056442	006105			2\$:	ROL	R5	::FORM THIS DIGIT
056444	006105				ROL	R5	
056446	006105				ROL	R5	
056450	010503				MOV	R5, R3	
056452	006103			3\$:	ROL	R3	::GET LSB OF THIS DIGIT
056454	105337	056556			DECB	\$OMODE	::TYPE THIS DIGIT?
056460	100016				BPL	7\$::BR IF NO
056462	042703	177770			BIC	#177770, R3	::GET RID OF JUNK
056466	001002				BNE	4\$::TEST FOR 0
056470	005704				TST	R4	::SUPPRESS THIS 0?
056472	001403				BEQ	5\$::BR IF YES
056474	005204			4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S

056476	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
056502	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
056506	110337	056552		MOVB	R3,8\$::SAVE FOR TYPING
056512	104401	056552		TYPE	8\$::GO TYPE THIS DIGIT
056516	105337	056554	7\$:	DECB	\$OCNT	::COUNT BY 1
056522	003347			BGT	2\$::BR IF MORE TO DO
056524	002402			BLT	6\$::BR IF DONE
056526	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
056530	000744			BR	2\$::GO DO THE LAST DIGIT
056532	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
056534	012604			MOV	(SP)+,R4	::RESTORE R4
056536	012603			MOV	(SP)+,R3	::RESTORE R3
056540	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
056546	012616			MOV	(SP)+,(SP)	
056550	000002			RTI		::RETURN
056552	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
056553	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
056554	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
056555	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
056556	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL TYPE ROUTINE

 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
 *NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
 *NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
 *NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 *

*CALL:
 *1) USING A TRAP INSTRUCTION
 * TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
 *OR
 * TYPE
 * MESADR
 *

056560	105737	001173	\$TYPE:	TSTB	\$TFPLG	::IS THERE A TERMINAL?
056564	100002			BPL	1\$::BR IF YES
056566	000000			HALT		::HALT HERE IF NO TERMINAL
056570	000430			BR	3\$::LEAVE
056572	010046		1\$:	MOV	RO,-(SP)	::SAVE RO
056574	017600	000002		MOV	@2(SP),RO	::GET ADDRESS OF ASCIZ STRING
056600	122737	000001	001242	CMPB	#APTENV,\$ENV	::RUNNING IN APT MODE
056606	001011			BNE	62\$::NO,GO CHECK FOR APT CONSOLE
056610	132737	000100	001243	BITB	#APTSPOOL,\$ENVM	::SPOOL MESSAGE TO APT
056616	001405			BEQ	62\$::NO,GO CHECK FOR CONSOLE
056620	010037	056630		MOV	RO,61\$::SETUP MESSAGE ADDRESS FOR APT
056624	004737	063206		JSR	PC,\$ATY3	::SPOOL MESSAGE TO APT
056630	000000		61\$:	.WORD	0	::MESSAGE ADDRESS
056632	132737	000040	001243	62\$:	BITB #APTCSUP,\$ENVM	::APT CONSOLE SUPPRESSED
056640	001003			BNE	60\$::YES,SKIP TYPE OUT
056642	112046		2\$:	MOVB	(RO)+,-(SP)	::PUSH CHARACTER TO BE TYPED ONTO STACK
056644	001005			BNE	4\$::BR IF IT ISN'T THE TERMINATOR
056646	005726			TST	(SP)+	::IF TERMINATOR POP IT OFF THE STACK
056650	012600		60\$:	MOV	(SP)+,RO	::RESTORE RO
056652	062716	000002		3\$:	ADD #2,(SP)	::ADJUST RETURN PC
056656	000002			RTI		::RETURN
056660	122716	000011		4\$:	CMPB #HT,(SP)	::BRANCH IF <HT>
056664	001430			BEQ	8\$	
056666	122716	000200		CMPB	#CRLF,(SP)	::BRANCH IF NOT <CRLF>
056672	001006			BNE	5\$	
056674	005726			TST	(SP)+	::POP <CR><LF> EQUIV
056676	104401			TYPE		::TYPE A CR AND LF
056700	001217			\$CRLF		
056702	105037	057110		CLRB	\$CHARCNT	::CLEAR CHARACTER COUNT
056706	000755			BR	2\$::GET NEXT CHARACTER
056710	004737	056772		5\$:	JSR PC,\$TYPEC	::GO TYPE THIS CHARACTER
056714	123726	001172		6\$:	CMPB \$FILLC,(SP)+	::IS IT TIME FOR FILLER CHARS.?
056720	001350			BNE	2\$::IF NO GO GET NEXT CHAR.
056722	013746	001170		MOV	\$NULL,-(SP)	::GET # OF FILLER CHARS. NEEDED
						::AND THE NULL CHAR.
056726	105366	000001	7\$:	DECB	1(SP)	::DOES A NULL NEED TO BE TYPED?
056732	002770			BLT	6\$::BR IF NO--GO POP THE NULL OFF OF STACK
056734	004737	056772		JSR	PC,\$TYPEC	::GO TYPE A NULL
056740	105337	057110		DECB	\$CHARCNT	::DO NOT COUNT AS A COUNT
056744	000770			BR	7\$::LOOP

:HORIZONTAL TAB PROCESSOR

056746	112716	000040		8\$:	MOVB	#' (SP)	::REPLACE TAB WITH SPACE
056752	004737	056772		9\$:	JSR	PC,\$TYPEC	::TYPE A SPACE
056756	132737	000007	057110		BITB	#7,\$SCHARCNT	::BRANCH IF NOT AT
056764	001372				BNE	9\$::TAB STOP
056766	005726				TST	(SP)+	::POP SPACE OFF STACK
056770	000724				BR	2\$::GET NEXT CHARACTER
056772				\$TYPEC:			
056772	105777	122162			TSTB	@\$TKS	::CHAR IN KYBD BUFFER?
056776	100022				BPL	10\$::BR IF NOT
057000	017746	122156			MOV	@\$TKB,-(SP)	::GET CHAR
057004	042716	177600			BIC	#177600,(SP)	::STRIP EXTRANEIOUS BITS
057010	122716	000023			CMPB	#\$XOFF,(SP)	::WAS CHAR XOFF
057014	001012				BNE	102\$::BR IF NOT
057016				101\$:			
057016	105777	122136			TSTB	@\$TKS	::WAIT FOR CHAR
057022	100375				BPL	101\$	
057024	117716	122132			MOVB	@\$TKB,(SP)	::GET CHAR
057030	042716	177600			BIC	#177600,(SP)	::STRIP IT
057034	122716	000021			CMPB	#\$XON,(SP)	::WAS IT XON?
057040	001366				BNE	101\$::BR IF NOT
057042				102\$:			
057042	005726				TST	(SP)+	::FIX STACK
057044				10\$:			
057044	105777	122114			TSTB	@\$TPS	::WAIT UNTIL PRINTER IS READY
057050	100375				BPL	10\$	
057052	116677	000002	122106		MOVB	2(SP),@\$TPB	::LOAD CHAR TO BE TYPED INTO DATA REG.
057060	122766	000015	000002		CMPB	#CR,2(SP)	::IS CHARACTER A CARRIAGE RETURN?
057066	001003				BNE	1\$::BRANCH IF NO
057070	105037	057110			CLRB	\$SCHARCNT	::YES--CLEAR CHARACTER COUNT
057074	000406				BR	\$TYPEX	::EXIT
057076	122766	000012	000002	1\$:	CMPB	#LF,2(SP)	::IS CHARACTER A LINE FEED?
057104	001402				BEQ	\$TYPEX	::BRANCH IF YES
057106	105227				INCB	(PC)+	::COUNT THE CHARACTER
057110	000000			\$SCHARCNT:	.WORD	0	::CHARACTER COUNT STORAGE
057112	000207			\$TYPEX:	RTS	PC	


```

057320 000002 RTI
057322 012637 000004 2001$: MOV (SP)+,ERRVEC ::RESTORE CONTENTS OF ERROR VECTOR

057326 022737 177777 060300 2002$: CMP #-1,CPSAVE ::SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
057334 001430 BEQ 2003$ ::BRANCH IF SO
057336 032737 000001 060300 BIT #BIT00,CPSAVE ::SEE IF THE POWER MONITOR BIT IS ON
057344 001424 BEQ 2003$ ::BRANCH TO CONTINUE ROUTINE IF CLEAR
057346 042737 000001 177766 BIC #BIT00,177766 ::CLEAR THE BIT FOUND TO BE SET
057354 013746 001154 MOV SWR,-(SP) ::SAVE SWR ADDRESS
057360 017646 000000 MOV @ (SP),-(SP) ::SAVE SWR VALUE
057364 012737 000176 001154 MOV #176,SWR ::GET SOFTWARE SWR ADDRESS
057372 011677 121556 MOV (SP),@SWR ::GET CURRENT SWR VALUE
057376 042777 001000 121550 BIC #BIT09,@SWR ::DON'T ALLOW LOOP ON ERROR ON THIS ERROR
057404 104177 EMT 177 ::CALL SPECIAL POWER FAIL BIT ERROR CALL
057406 012676 000000 MOV (SP)+,@(SP) ::RESTORE SWR TO ORIGINAL VALUE
057412 012637 001154 MOV (SP)+,SWR ::RESTORE SWR ADDRESS
057416 2003$:
057416 123737 001131 001117 CMPB $ERMAX,$ERFLG ::MAX. ERRORS FOR THIS TEST OCCURRED?
057424 101015 BHI 3$ ::BR IF NO
057426 032777 001000 121520 BIT #BIT09,@SWR ::LOOP ON ERROR?
057434 001404 BEQ 4$ ::BR IF NO
057436 013737 001124 001122 7$: MOV $LPERR,$LPADR ::SET LOOP ADDRESS TO LAST SCOPE
057444 000446 BR $OVER
057446 105037 001117 4$: CLRB $ERFLG ::ZERO THE ERROR FLAG
057452 005037 001206 CLR $TIMES ::CLEAR THE NUMBER OF ITERATIONS TO MAKE
057456 000415 BR 1$ ::ESCAPE TO THE NEXT TEST
057460 032777 004000 121466 3$: BIT #BIT11,@SWR ::INHIBIT ITERATIONS?
057466 001011 BNE 1$ ::BR IF YES
057470 005737 001230 TST $PASS ::IF FIRST PASS OF PROGRAM
057474 001406 BEQ 1$ ::INHIBIT ITERATIONS
057476 005237 001120 INC $ICNT ::INCREMENT ITERATION COUNT
057502 023737 001206 001120 CMP $TIMES,$ICNT ::CHECK THE NUMBER OF ITERATIONS MADE
057510 002024 BGE $OVER ::BR IF MORE ITERATION REQUIRED
057512 012737 000001 001120 1$: MOV #1,$ICNT ::REINITIALIZE THE ITERATION COUNTER
057520 013737 057576 001206 MOV $MXCNT,$TIMES ::SET NUMBER OF ITERATIONS TO DO
057526 105237 001116 $SVLAD: INCB $STNM ::COUNT TEST NUMBERS
057532 113737 001116 001226 MOVB $STNM,$TESTN ::SET TEST NUMBER IN APT MAILBOX
057540 011637 001122 MOV (SP),$LPADR ::SAVE SCOPE LOOP ADDRESS
057544 011637 001124 MOV (SP),$LPERR ::SAVE ERROR LOOP ADDRESS
057550 005037 001210 CLR $ESCAPE ::CLEAR THE ESCAPE FROM ERROR ADDRESS
057554 112737 000001 001131 MOVB #1,$ERMAX ::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
057562 013777 001116 121366 $OVER: MOV $STNM,@DISPLAY ::DISPLAY TEST NUMBER
057570 013716 001122 MOV $LPADR,(SP) ::FUDGE RETURN ADDRESS
057574 000002 RTI ::FIXES PS
057576 000005 SMXCNT: 5. ::MAX. NUMBER OF ITERATIONS
057600 $SWOBTBL:
.REPT $TN-1
057600 000031 .WORD TST1+2 ::STARTING ADDRESS OF TEST 1
057602 010100 .WORD TST2+2 ::STARTING ADDRESS OF TEST 2
057604 011012 .WORD TST3+2 ::STARTING ADDRESS OF TEST 3
057606 011646 .WORD TST4+2 ::STARTING ADDRESS OF TEST 4
057610 012436 .WORD TST5+2 ::STARTING ADDRESS OF TEST 5
057612 013376 .WORD TST6+2 ::STARTING ADDRESS OF TEST 6
057614 014216 .WORD TST7+2 ::STARTING ADDRESS OF TEST 7
057616 015006 .WORD TST10+2 ::STARTING ADDRESS OF TEST 10
057620 015744 .WORD TST11+2 ::STARTING ADDRESS OF TEST 11
057622 016562 .WORD TST12+2 ::STARTING ADDRESS OF TEST 12

```


057624	017222		.WORD	TST13+2	::STARTING ADDRESS OF TEST 13
057626	017676		.WORD	TST14+2	::STARTING ADDRESS OF TEST 14
057630	020520		.WORD	TST15+2	::STARTING ADDRESS OF TEST 15
057632	021262		.WORD	TST16+2	::STARTING ADDRESS OF TEST 16
057634	022024		.WORD	TST17+2	::STARTING ADDRESS OF TEST 17
057636	022376		.WORD	TST20+2	::STARTING ADDRESS OF TEST 20
057640	022750		.WORD	TST21+2	::STARTING ADDRESS OF TEST 21
057642	023312		.WORD	TST22+2	::STARTING ADDRESS OF TEST 22
057644	023634		.WORD	TST23+2	::STARTING ADDRESS OF TEST 23
057646	024152		.WORD	TST24+2	::STARTING ADDRESS OF TEST 24
057650	025064		.WORD	TST25+2	::STARTING ADDRESS OF TEST 25
057652	025706		.WORD	TST26+2	::STARTING ADDRESS OF TEST 26
057654	026610		.WORD	TST27+2	::STARTING ADDRESS OF TEST 27
057656	027676		.WORD	TST30+2	::STARTING ADDRESS OF TEST 30
057660	030522		.WORD	TST31+2	::STARTING ADDRESS OF TEST 31

2					
3					
4					
5	057662		STOP:		
	057662	012746		MOV #PR3,-(SP)	::PUT NEW PS ON STACK
	057666	012746		MOV #64\$,-(SP)	::PUT NEW PC ON STACK
	057672	000002		RTI	::POP NEW PC AND PS
	057674		64\$:		
6					
7					
8					
9	057674	012746		MOV #PR5,-(SP)	::PUT NEW PS ON STACK
	057700	012746		MOV #65\$,-(SP)	::PUT NEW PC ON STACK
	057704	000002		RTI	::POP NEW PC AND PS
	057706		65\$:		
10	057706	000207		RTS PC	:RETURN

1

.SBTTL ERROR HANDLER ROUTINE

```

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

```

057710 105037 060302      SERROR: CLRB      IBSAVE      ;;CLEAR THE ITEM BYTE SAVE LOCATION
057714 104410              CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
057716 105237 001117      7$:      INCB      SERFLG      ;;SET THE ERROR FLAG
057722 001775              BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
057724 013777 001116 121224  MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
057732 032777 002000 121214  BIT      #BIT10,@SWR      ;;BELL ON ERROR?
057740 001402              BEQ      1$      ;;NO - SKIP
057742 104401 001212              TYPE      ,SBELL      ;;RING BELL
057746 005237 001126      1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
057752 011637 001132              MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
057756 162737 000002 001132  SUB      #2,$ERRPC
057764 117737 121142 001130  MOVB     @SERPC,$ITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
057772 032777 001000 121154  BIT      #BIT09,@SWR      ;;SEE IF LOOP ON ERROR IS SET
060000 001060              BNE      1004$      ;;BRANCH AROUND ROUTINE IF SO
060002 122737 000177 001130  CMPB     #177,$ITEMB      ;;SEE IF THIS IS THE POWER FAIL CALL
060010 001454              BEQ      1004$      ;;BRANCH AROUND ROUTINE IF IT IS
060012 105737 060302              TSTB     IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
060016 001047              BNE      1003$      ;;BRANCH IF SO
060020 022737 177777 060300  CMP      #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
060026 001445              BEQ      1004$      ;;BRANCH IF SO
060030 013746 000004              MOV      ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
060034 012737 060052 000004  MOV      #1000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
060042 013737 177766 060300  MOV      177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
060050 000406              BR      1001$
060052 012737 177777 060300 1000$:  MOV      #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
060060 012716 060066              MOV      #1001$, (SP)    ;;SETUP RETURN ADDRESS
060064 000002              RTI
060066 012637 000004      1001$:  MOV      (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

060072 022737 177777 060300 1002$:  CMP      #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
060100 001420              BEQ      1004$      ;;BRANCH IF SO
060102 032737 000001 060300  BIT      #BIT00,CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
060110 001414              BEQ      1004$      ;;BRANCH IF OK
060112 042737 000001 177766  BIC      #BIT00,177766   ;;CLEAR THE BIT FOUND SET
060120 113737 001130 060302  MOVB     $ITEMB,IBSAVE   ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
060126 112737 000177 001130  MOVB     #177,$ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
060134 000402              BR      1004$      ;;BRANCH OVER IBSAVE CLEARING

060136 105037 060302      1003$:  CLRB      IBSAVE      ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
060142 032777 020000 121004 1004$:  BIT      #BIT13,@SWR    ;;SKIP TYPEOUT IF SET
060150 001004              BNE      20$      ;;SKIP TYPEOUTS
060152 004737 060304              JSR      PC,ERRTP      ;;GO TO USER ERROR ROUTINE
    
```

ERROR HANDLER ROUTINE

```

060156 104401 001217          TYPE      ,SCLRF
060162 122737 000001 001242 20$:      CMPB      #APTENV,$ENV      ::RUNNING IN APT MODE
060170 001007          BNE        2$              ::NO,SKIP APT ERROR REPORT
060172 113737 001130 060204  MOVB      $ITEMB,21$      ::SET ITEM NUMBER AS ERROR NUMBER
060200 004737 063216          JSR        PC,$ATY4      ::REPORT FATAL ERROR TO APT
060204 000          21$:      .BYTE     0
060205 000          .BYTE     0
060206 000777          BR         22$              ::APT ERROR LOOP
060210 105737 060302 22$:      TSTB      IBSAVE      ::SEE IF IBSAVE IS LOADED
060214 001005          BNE        3$              ::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
060216 005777 120732          TST        @SWR          ::HALT ON ERROR
060222 100002          BPL        3$              ::SKIP IF CONTINUE
060224 000000          HALT
060226 104410          CKSWR          ::HALT ON ERROR!
060230          3$:          BIT         #BIT09,@SWR      ::TEST FOR CHANGE IN SOFT-SWR
060230 032777 001000 120716  BIT         4$              ::LOOP ON ERROR SWITCH SET?
060236 001402          BEQ        4$              ::BR IF NO
060240 013716 001124          MOV        $LPERR,(SP)   ::FUDGE RETURN FOR LOOPING
060244 005737 001210 4$:          TST        $ESCAPE     ::CHECK FOR AN ESCAPE ADDRESS
060250 001402          BEQ        5$              ::BR IF NONE
060252 013716 001210          MOV        $ESCAPE,(SP) ::FUDGE RETURN ADDRESS FOR ESCAPE
060256          5$:          CMP         #SENDAD,@#42      ::ACT-11 AUTO-ACCEPT?
060256 022737 031402 000042  BNE        6$              ::BRANCH IF NO
060264 001001          HALT              ::YES
060266 000000
060270          6$:          TSTB      IBSAVE      ::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
060270 105737 060302          BNE        7$              ::BRANCH BACK TO CALL ORIGINAL ERROR
060274 001210          RTI              ::RETURN
060276 000002
060300 000000          CPSAVE: .WORD     0      ::LOCATION TO SAVE CPU ERROR REG CONTENTS
060302 000000          IBSAVE: .WORD     0      ::LOCATION TO SAVE ITEM BYTE

```

.SBTTL ERROR TYPEOUT ROUTINE

*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
 *REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
 *
 * UNIT NUMBER, DRIVE TYPE, 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.

2
3
4
5
6
7
8
9
10
11
12
13 060304 104414
14 060306 032777 020000 120640
15 060314 001402
16 060316 000137 061114
17
18
19
20 060322 104401 001217
21 060326 104401 061130
22 060332 013746 001234

060336 104403
060340 003
060341 000

23
24
25 060342 013700 001276
26 060346 016000 000026
27 060352 042700 177740
28 060356 012737 064522 060400
29 060364 022700 000026
30 060370 001004
31 060372 104401 061165
32 060376 104401
33 060400 000000
34
35
36 060402 005037 061120
37 060406 013737 001226 061120
38 060414 104401 061135
39 060420 013746 061120

060424 104403
060426 003
060427 000
40 060430 005037 061122
41 060434 113737 001130 061122
42 060442 001406
43 060444 104401 061145
44 060450 013746 061122

060454 104403
060456 003
060457 000
45 060460 104401 061154

ERRTYP: SAVREG
 BIT #SW13,@SWR ;INHIBIT TYPEOUTS??
 BEQ 1\$;NO!!
 JMP 27\$;YES!!

 ;TYPE UNIT NUMBER, DRIVE TYPE, TEST NUMBER, ERROR NUMBER, AND
 ;PROGRAM COUNTER
 1\$: TYPE .SCLRF ;TYPE "DRV#"
 TYPE .ERTY00 ;:SAVE \$UNIT FOR TYPEOUT
 MOV \$UNIT,-(SP) ;:TYPE DRIVE NUMBER
 ;:GO TYPE--OCTAL ASCII
 TYPOS ;:TYPE 3 DIGIT(S)
 .BYTE 3 ;:SUPPRESS LEADING ZEROS
 .BYTE 0

 ;TYPE 'DRIVE TYPE', RM80 FOR UNIT UNDER TEST
 MOV \$BASE,R0 ;GET RM BASE ADDRESS
 MOV RMDT(R0),R0 ;GET DRIVE TYPE REGISTER
 BIC #177740,R0 ;SAVE DRIVE TYPE BITS AND
 MOV #SRM80,3\$;GET ASCII DRIVE TYPE
 CMP #26,R0 ;IS DEVICE AN RM80 ?
 BNE 4\$;NO !!
 2\$: TYPE " - " ;TYPE DRIVE TYPE
 TYPE .ERTY05 ;DRIVE TYPE MESSAGE IS STORED HERE
 3\$: .WORD 0

 ;TYPE TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER
 4\$: CLR TSTNMB ;LOAD TEST NUMBER FOR
 MOV \$TESTN,TSTNMB
 TYPE .ERTY01 ;TYPE "TST#"
 MOV TSTNMB,-(SP) ;:SAVE TSTNMB FOR TYPEOUT
 ;:TYPE TEST NUMBER
 TYPOS ;:GO TYPE--OCTAL ASCII
 .BYTE 3 ;:TYPE 3 DIGIT(S)
 .BYTE 0 ;:SUPPRESS LEADING ZEROS
 CLR ERRNMB ;LOAD ERROR NUMBER FOR
 MOV \$ITEMB,ERRNMB ;TYPEOUT
 BEQ 5\$;SKIP IF NO ERROR CALLED
 TYPE .ERTY02 ;TYPE "ERR#"
 MOV ERRNMB,-(SP) ;:SAVE ERRNMB FOR TYPEOUT
 ;:TYPE ERROR NUMBER
 TYPOS ;:GO TYPE--OCTAL ASCII
 .BYTE 3 ;:TYPE 3 DIGIT(S)
 .BYTE 0 ;:SUPPRESS LEADING ZEROS
 5\$: TYPE .ERTY03 ;TYPE "PC="

```

46 060464 013746 001132      MOV      $ERRPC,-(SP)      ;;SAVE SERRPC FOR TYPEOUT
                                ;;TYPE PROGRAM COUNTER
                                ;;GO TYPE--OCTAL ASCII
                                ;;TYPE 6 DIGIT(S)
                                ;;TYPE LEADING ZEROS
    060470 104403      TYPOS
    060472      006      .BYTE      6
    060473      001      .BYTE      1
47
48      ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
49 060474 005737 061122 6$:      TST      ERRNMB      ;WAS AN ERROR CALLED?
50 060500 001002      BNE      7$              ;BR IF YES
51 060502 000137 061114      JMP      27$            ;NO--EXIT
52
53 060506 104401 001217 7$:      TYPE      ,SCRLF      ;YES-TYPE CRLF
54 060512 105037 061126      CLRB     BOTFLG        ;CLEAR BOT FLAG
55 060516 105037 061127      CLRB     CHRCNT        ;CLEAR CHARACTER COUNTER
56 060522 013700 061122      MOV      ERRNMB,R0     ;R0 POINTS TO FIRST OF
57 060526 122700 000177      CMPB    #177,R0        ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
58 060532 001003      BNE      8$              ;BRANCH IF NOT
59 060534 012700 061172      MOV      #PFECH,R0     ;MOVE POWER FAIL ERROR CALL TABLE TO R0
60 060540 000405      BR      9$
61 060542 006300 8$:      ASL      R0              ;FOUR ENTRIES IN ERROR
62 060544 006300      ASL      R0              ;TABLE
63 060546 006300      ASL      R0
64 060550 062700 001574 9$:      ADD      #$ERRTB-8.,R0 ;R1 POINTS TO ERROR MESSAGE
65 060554 011001      MOV      (R0),R1      ;TABLE
66
67 060556 001507      BEQ      19$            ;BRANCH IF NO ERROR MESSAGE
68
69      ;TYPE THE ERROR MESSAGE
70 060560 012102 10$:     MOV      (R1)+,R2      ;R2=ADDRESS OF MESSAGE STRING
71 060562 001505      BEQ      19$            ;BRANCH IF END OF MESSAGE
72 060564 010237 060732      MOV      R2,18$        ;LOAD ADDRESS OF STRING
73 060570 005037 061124      CLR     BOTADR        ;CLEAR BOT ADDRESS
74 060574 112203 11$:     MOVB    (R2)+,R3      ;END OF STRING??
75 060576 001454      BEQ      17$            ;YES!!
76 060600 122703 000015      CMPB    #CR,R3        ;CARRIAGE RETURN??
77 060604 001003      BNE      12$            ;NO!!
78 060606 105037 061127      CLRB     CHRCNT        ;YES-CLEAR CHAR COUNT
79 060612 000770      BR      11$            ;GET NEXT CHARACTER
80 060614 122703 000012 12$:     CMPB    #LF,R3        ;LINE FEED??
81 060620 001765      BEQ      11$            ;YES-GET NEXT CHARACTER
82 060622 122703 000011      CMPB    #HT,R3        ;HORIZONTAL TAB??
83 060626 001007      BNE      14$            ;NO!!
84 060630 105237 061127 13$:     INCB    CHRCNT        ;ADJUST CHARACTER COUNT
85 060634 132737 000007 061127      BITB    #7,CHRCNT
86 060642 001372      BNE      13$
87 060644 000407      BR      15$
88 060646 105237 061127 14$:     INCB    CHRCNT        ;INCREMENT CHARACTER COUNT
89 060652 122703 000040      CMPB    #' ,R3        ;SPACE??
90 060656 001002      BNE      15$            ;NO!!
91 060660 010237 061124      MOV      R2,BOTADR     ;SAVE ADDRESS OF SPACE
92 060664 122737 000100 061127 15$:     CMPB    #64.,CHRCNT   ;END OF LINE??
93 060672 103340      BHS     11$            ;NO!!
94 060674 013704 061124      MOV      BOTADR,R4     ;GET ADDRESS OF LAST SPACE
95 060700 001007      BNE      16$            ;BRANCH IF SPACE DETECTED
96 060702 104401 001217      TYPE    ,SCRLF        ;TYPE CRLF
97 060706 105037 061127      CLRB     CHRCNT        ;CLEAR CHARACTER COUNT
98 060712 013702 060732      MOV      18$,R2       ;SET UP R2 FOR TESTING
    
```

```

99 060716 000726
100 060720 105044
101 060722 112737 177777 061126
102 060730 104401
103 060732 000000
104 060734 105737 061126
105 060740 001707
106 060742 104401 001217
107 060746 105037 061126
108 060752 105037 061127
109 060756 013702 061124
110 060762 010237 060732
111 060766 112742 000040
112 060772 105722
113 060774 000677
114
115
116 060776 016001 000002
117 061002 001444
118 061004 104401 001217
119 061010 016002 000004
120 061014 016003 000006
121 061020 012137 061030
122 061024 001433
123
124 061026 104401
125 061030 000000
126 061032 104401 001217
127 061036 005702
128 061040 001767
129 061042 012204
130 061044 012305
131 061046 105725
132 061050 100407
133 061052 001403
134 061054 013446
135 061056 104405
136 061060 000405
137 061062 013446
138 061064 104402
139 061066 000402
140 061070 013446
141 061072 104406
142 061074 005714
143 061076 001403
144 061100 104401 061162
145 061104 000760
146 061106 104401 001217
147 061112 000742
148 061114 104415
149 061116 000207
150
151 061120 000000
152 061122 000000
153 061124 000000
154 061126 000
155 061127 000

16$: BR 11$
      CLRB -(R4)
      MOVB #-1,BOTFLG
17$: TYPE
18$: .WORD
      TSTB BOTFLG
      BEQ 10$
      TYPE ,SCLF
      CLRB BOTFLG
      CLRB CHRCNT
      MOV BOTADR,R2
      MOV R2,18$
      MOVB #'-(R2)
      TSTB (R2)+
      BR 11$
      :REPLACE SPACE
      :SET BOT FLAG
      :TYPE ERROR MESSAGE STRING
      :STRING ADDRESS GOES HERE
      :WAS STRING TRUNCATED??
      :NO!!
      :YES-TYPE CRLF
      :CLEAR BOT FLAG
      :CLEAR CHARACTER COUNT
      :SETUP R2 FOR TESTING
      :SETUP 18$ FOR TYPING
      :RESTORE SPACE
      :RESTORE R2
      :TYPE REST OF STRING

:TYPE ERROR HEADER AND ERROR DATA
19$: MOV 2(R0),R1
      BEQ 27$
      TYPE ,SCLF
      MOV 4(R0),R2
      MOV 6(R0),R3
20$: MOV (R1)+,21$
      BEQ 27$
      :R1 POINTS TO ERROR HEADER TABLE
      :BRANCH IF NO HEADER
      :(ASSUME NO DATA)
      :R2 POINTS TO DATA ADDRESS TABLE
      :R3 POINTS TO FORMAT TABLE
      :PUT HEADER ADDRESS FOR TYPE
      :BRANCH IF END OF HEADERS
      :(ASSUME END OF DATA)

21$: TYPE
      .WORD 0
      TYPE ,SCLF
      TST R2
      BEQ 20$
      MOV (R2)+,R4
      MOV (R3)+,R5
22$: TSTB (R5)+
      BMI 24$
      BEQ 23$
      MOV @ (R4)+,-(SP)
      TYPDS
      BR 25$
23$: MOV @ (R4)+,-(SP)
      TYPOC
      BR 25$
24$: MOV @ (R4)+,-(SP)
      TYPBN
25$: TST (R4)
      BEQ 26$
      TYPE ,ERTY04
      BR 22$
26$: TYPE ,SCLF
      BR 20$
27$: RESREG
      RTS PC
      :HEADER ADDRESS GOES HERE
      :DATA WITH HEADER??
      :NO!!
      :R4 POINTS TO DATA ADDRESS
      :R5 POINTS TO FORMAT
      :WHAT KIND OF DATA??
      :BINARY
      :OCTAL
      :DECIMAL
      :MORE DATA??
      :NO!!
      :YES-TYPE 2 SPACES
      :AND CONTINUE
      :TYPE ONE BLANK LINE
      :BEFORE NEXT HEADER
      :TEST NUMBER
      :ERROR NUMBER
      :BEGINNING OF TEXT ADDRESS
      :BOT FLAG
      :CHARACTER COUNT
  
```

156						
157	061130	104	122	126	ERTY00:	.ASCIZ @DRV#@
158	061135	054	040	124	ERTY01:	.ASCIZ @, TEST#@
159	061145	054	040	105	ERTY02:	.ASCIZ @, ERR#@
160	061154	054	040	120	ERTY03:	.ASCIZ @, PC=@
161	061162	040	040	000	ERTY04:	.ASCIZ @ @
162	061165	040	055	040	ERTY05:	.ASCIZ @ - @
163						.EVEN
164	061172	061202	061270	061306	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4 ;WORDS DEFINING TABLES BELOW
165	061202	061206	000000		PFECH1:	+.4,0
166	061206	120	117	127		.ASCIZ ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
167						.EVEN
168	061270	061274	000000		PFECH2:	+.4,0
169	061274	103	120	125		.ASCIZ ?CPUERREG?
170						.EVEN
171	061306	061310			PFECH3:	+.2
172	061310	060300	000000			.WORD CPSAVE,0
173	061314	061316			PFECH4:	+.2
174	061316	000	000			.BYTE 0,0

1

.SBTTL TTY INPUT ROUTINE

061320 000000
061322 000000
061324 000000
061326 061327

```
.ENABL LSB
$TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0          ;;INPUT POINTER
$TKQOUT: .WORD 0         ;;OUTPUT POINTER
$TKQSRV: .BLKB 1        ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN
```

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT

```
*CALL:
*      JSR      PC,$TKINT
*      RETURN
```

061330 005037 061320
061334 012737 061326 061322
061342 013737 061322 061324
061350 012737 061400 000060
061356 012737 000200 000062
061364 005777 117572
061370 012777 000100 117562
061376 000207

```
$TKINT: CLR      $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
        MOV      #$TKQSRV,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
        MOV      $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV      #$TKSRV,@$TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
        MOV      #200,@$TKVEC+2 ;;'BR' LEVEL 4
        TST      @$TKB          ;;CLEAR DONE FLAG
        MOV      #100,@$TKS     ;;ENABLE TTY KEYBOARD INTERRUPT
        RTS      PC            ;;RETURN TO CALLER
```

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A "CONTROL-C" (^C) \$TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (SHUT2)

061400 117746 117556
061404 042716 177600
061410 021627 000021
061414 001002
061416 005726
061420 000002
061422
061422 021627 000003
061426 001007
061430 104401 062526
061434 004737 061330
061440 005726
061442 000137 062570
061446 021627 000007
061452 001004
061454 022737 000176 001154
061462 001500

061464
061464 022737 000001 061320
061472 001004
061474 104401 001212

```
$TKSRV: MOVB     @$TKB,-(SP)    ;;PICKUP THE CHARACTER
        BIC      #^C177,(SP)   ;;STRIP THE JUNK
        CMP      (SP),#$XON    ;;IS IT A RANDOM XON?
        BNE     30$           ;;BRANCH IF NO
        TST      (SP)+        ;;CLEAN RANDOM XON OFF STACK
        RTI                    ;;RETURN

30$:    CMP      (SP),#3        ;;IS IT A CONTROL C?
        BNE     1$           ;;BRANCH IF NO
        TYPE     ,SCNTLC      ;;TYPE A CONTROL-C (^C)
        JSR     PC,$TKINT     ;;INIT THE KEYBOARD
        TST      (SP)+        ;;CLEAN UP STACK
        JMP     SHUT2        ;;CONTROL C RESTART

1$:    CMP      (SP),#7        ;;IS IT A CONTROL G?
        BNE     2$           ;;BRANCH IF NO
        CMP      #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
        BEQ     6$           ;;GO TO SWR CHANGE

2$:    CMP      #1,$TKCNT      ;;IS THE QUEUE FULL?
        BNE     3$           ;;BRANCH IF NO
        TYPE     ,SBELL       ;;RING THE TTY BELL

6$:    BEQ     6$
```



```

061500 005726          TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
061502 000451          BR       5$          ;;EXIT
061504 021627 000023 3$:    CMP      (SP),#23      ;;IS IT A CONTROL-S?
061510 001021          BNE      32$          ;;BRANCH IF NO
061512 005077 117442    CLR      @STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
061516 005726          TST      (SP)+      ;;CLEAN CHAR OFF STACK
061520 105777 117434 31$:  TSTB     @STKS        ;;WAIT FOR A CHAR
061524 100375          BPL      31$          ;;LOOP UNTIL ITS THERE
061526 117746 117430    MOVB     @STKB,-(SP)  ;;GET THE CHARACTER
061532 042716 177600    BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
061536 022627 000021    CMP      (SP)+,#21   ;;IS IT A CONTROL-Q?
061542 001366          BNE      31$          ;;BRANCH IF NO
061544 012777 000100 117406 MOV      #100,@STKS  ;;REENABLE TTY KEYBOARD INTERRUPTS
061552 000002          RTI                    ;;RETURN
061554 005237 061320 32$:  INC      $TKCNT       ;;COUNT THIS CHARACTER
061560 021627 000140    CMP      (SP),#140   ;;IS IT UPPER CASE?
061564 002405          BLT      4$          ;;BRANCH IF YES
061566 021627 000175    CMP      (SP),#175   ;;IS IT A SPECIAL CHAR?
061572 003002          BGT      4$          ;;BRANCH IF YES
061574 042716 000040    BIC      #40,(SP)    ;;MAKE IT UPPER CASE
061600 112677 177516 4$:  MOVB     (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
061604 005237 061322    INC      $TKQIN      ;;UPDATE THE POINTER
061610 023727 061322 061327 CMP      $TKQIN,#$TKQEND ;;GO OFF THE END?
061616 001003          BNE      5$          ;;BRANCH IF NO
061620 012737 061326 061322 MOV      #$TKQSRST,$TKQIN ;;RESET THE POINTER
061626 000002          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.

```

```

061630 022737 000176 001154 $CKSWR: CMP      #SWREG,SWR  ;;IS THE SOFT-SWR SELECTED
061636 001124          BNE      15$          ;;EXIT IF NOT
061640 105777 117314    TSTB     @STKS        ;;IS A CHAR WAITING?
061644 100121          BPL      15$          ;;IF NOT, EXIT
061646 117746 117310    MOVB     @STKB,-(SP)  ;;YES
061652 042716 177600    BIC      #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
061656 021627 000007    CMP      (SP),#7     ;;IS IT A CONTROL-G?
061662 001300          BNE      2$          ;;IF NOT, PUT IT IN THE TTY QUEUE
                          ;;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.

```

```

061664 123727 001150 000001 6$:  CMPB     $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
061672 001674          BEQ      2$          ;;BRANCH IF YES
061674 005726          TST      (SP)+      ;;CLEAR CONTROL-G OFF STACK
061676 004737 061330    JSR      PC,$TKINT   ;;FLUSH THE TTY INPUT QUEUE
061702 005077 117252    CLR      @STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
061706 112737 000001 001151    MOVB     #1,$INTAG   ;;SET INTERRUPT MODE INDICATOR

061714 104401 062540    TYPE     ,SCNTLG     ;;ECHO THE CONTROL-G (^G)
061720 104401 062545    TYPE     ,SMSWR      ;;TYPE CURRENT CONTENTS
061724 013746 000176    MOV      SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
061730 104402          TYPOC             ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

```

```

061732 104401 062556
061736 005046
061740 005046
061742 105777 117212
061746 100375
19$: CLR -(SP)
7$: CLR -(SP)
TSTB @STKS
BPL 7$
MOV @STKB, -(SP)
BIC #^C177, (SP)
CMP (SP), #3
BNE 9$
TYPE ,SCNTLC
ADD #6, SP
CMPB $INTAG, #1
BNE 8$
MOV #100, @STKS
JMP SHUT2
19$: CMP (SP), #25
BNE 10$
TYPE ,SCNTLU
20$: ADD #6, SP
BR 19$
10$: CMP (SP), #15
BNE 16$
TST 4(SP)
BEQ 11$
MOV 2(SP), @SWR
11$: ADD #6, SP
14$: TYPE ,SCRLF
CMPB $INTAG, #1
BNE 15$
MOV #100, @STKS
15$: RTI
16$: JSR PC, $TYPEC
CMP (SP), #60
BLT 18$
CMP (SP), #67
BGT 18$
BIC #60, (SP)+
TST 2(SP)
BEQ 17$
ASL (SP)
ASL (SP)
ASL (SP)
17$: INC 2(SP)
BIS -2(SP), (SP)
BR 7$
18$: TYPE ,SQUES
BR 20$
.DSABL LSB

```

```

::PROMPT FOR NEW SWR
::CLEAR COUNTER
::THE NEW SWR
::CHAR THERE?
::IF NOT TRY AGAIN
::PICK UP CHAR
::MAKE IT 7-BIT ASCII
::IS IT A CONTROL-C?
::BRANCH IF NOT
::YES, ECHO CONTROL-C (^C)
::CLEAN UP STACK
::REENABLE TTY KEYBOARD INTERRUPTS?
::BRANCH IF NO
::ALLOW TTY KEYBOARD INTERRUPTS
::CONTROL-C RESTART
::IS IT A CONTROL-U?
::BRANCH IF NOT
::YES, ECHO CONTROL-U (^U)
::IGNORE PREVIOUS INPUT
::LET'S TRY IT AGAIN
::IS IT A <CR>?
::BRANCH IF NO
::YES, IS IT THE FIRST CHAR?
::BRANCH IF YES
::SAVE NEW SWR
::CLEAR UP STACK
::ECHO <CR> AND <LF>
::RE-ENABLE TTY KBD INTERRUPTS?
::BRANCH IF NOT
::RE-ENABLE TTY KBD INTERRUPTS
::RETURN
::ECHO CHAR
::CHAR < 0?
::BRANCH IF YES
::CHAR > 7?
::BRANCH IF YES
::STRIP-OFF ASCII
::IS THIS THE FIRST CHAR
::BRANCH IF YES
::NO, SHIFT PRESENT
::CHAR OVER TO MAKE
::ROOM FOR NEW ONE.
::KEEP COUNT OF CHAR
::SET IN NEW CHAR
::GET THE NEXT ONE
::TYPE ?<CR><LF>
::SIMULATE CONTROL-U

```

::*****

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *      RDCHR          ::GET A CHARACTER FROM THE QUEUE
: *      RETURN HERE   ::CHARACTER IS ON THE STACK
: *                   ::WITH PARITY BIT STRIPPED OFF
:
062172 011646          SRDCHR: MOV      (SP),-(SP)      ::PUSH DOWN THE PC AND
062174 016666 000004 000002  MOV      4(SP),2(SP)    ::THE PS
062202 005066 000004          CLR      4(SP)          ::GET READY FOR A CHARACTER
062206 005046          CLR      -(SP)          ::PUT NEW PS ON STACK
062210 012746 062216          MOV      #64$,-(SP)      ::PUT NEW PC ON STACK
062214 000002          RTI                    ::POP NEW PC AND PS
062216
062216 005737 061320 64$:  TST      $TKCNT          ::WAIT ON A CHARACTER
062222 001775 1$:      BEQ      1$
062224 005337 061320          DEC      $TKCNT          ::DECREMENT THE COUNTER
062230 117766 177070 000004  MOVB   @$TKQOUT,4(SP)    ::GET ONE CHARACTER
062236 005237 061324          INC      $TKQOUT        ::UPDATE THE POINTER
062242 023727 061324 061327  CMP     $TKQOUT,#$TKQEND ::DID IT GO OFF OF THE END?
062250 001003          BNE     2$              ::BRANCH IF NO
062252 012737 061326 061324  MOV     #$TKQ$RT,$TKQOUT ::RESET THE POINTER
062260 000002          RTI                    ::RETURN
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *      RDLIN          ::INPUT A STRING FROM THE TTY
: *      RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                   ::TERMINATOR WILL BE A BYTE OF ALL 0'S
:
062262 010346          $RDLIN: MOV     R3,-(SP)      ::SAVE R3
062264 005046          CLR     -(SP)          ::CLEAR THE RUBOUT KEY
062266 012703 062516 1$:  MOV     #$TTYIN,R3      ::GET ADDRESS
062272 022703 062526 2$:  CMP     #$TTYIN+8.,R3    ::BUFFER FULL?
062276 101456          BLOS   4$              ::BR IF YES
062300 104411          RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
062302 112613          MOVB   (SP)+,(R3)      ::GET CHARACTER
062304 122713 000177 10$:  CMPB   #177,(R3)       ::IS IT A RUBOUT
062310 001022          BNE   5$              ::BR IF NO
062312 005716          TST   (SP)            ::IS THIS THE FIRST RUBOUT?
062314 001007          BNE   6$              ::BR IF NO
062316 112737 000134 062514  MOVB   #'\",9$         ::TYPE A BACK SLASH
062324 104401 062514          TYPE  ,9$
062330 012716 177777          MOV   #-1,(SP)        ::SET THE RUBOUT KEY
062334 005303 6$:      DEC     R3              ::BACKUP BY ONE
062336 020327 062516          CMP   R3,$$TTYIN      ::STACK EMPTY?
062342 103434          BLO   4$              ::BR IF YES
062344 111337 062514          MOVB   (R3),9$        ::SETUP TO TYPEOUT THE DELETED CHAR.
062350 104401 062514          TYPE  ,9$
062354 000746          BR    2$              ::GO TYPE
062356 005716 5$:      TST   (SP)            ::GO READ ANOTHER CHAR.
062360 001406          BEQ   7$              ::RUBOUT KEY SET?
062362 112737 000134 062514  MOVB   #'\",9$         ::BR IF NO
062370 104401 062514          TYPE  ,9$
062374 005016          CLR   (SP)            ::TYPE A BACK SLASH
062376 122713 000025 7$:  CMPB   #25,(R3)       ::CLEAR THE RUBOUT KEY
062402 001003          BNE   8$              ::IS CHARACTER A CTRL U?
: *                   ::BR IF NO

```

062404	104401	062533			TYPE	SCNTLU	::TYPE A CONTROL 'U'	
062410	000726				BR	1\$::GO START OVER	
062412	122713	000022	8\$:		CMPB	#22,(R3)	::IS CHARACTER A '^R'?	
062416	001011				BNE	3\$::BRANCH IF NO	
062420	105013				CLRB	(R3)	::CLEAR THE CHARACTER	
062422	104401	001217			TYPE	,\$SCRLF	::TYPE A 'CR' & 'LF'	
062426	104401	062516			TYPE	,\$STTYIN	::TYPE THE INPUT STRING	
062432	000717				BR	2\$::GO PICKUP ANOTHER CHACTER	
062434	104401	001216	4\$:		TYPE	,\$QUES	::TYPE A '?'	
062440	000712				BR	1\$::CLEAR THE BUFFER AND LOOP	
062442	111337	062514	3\$:		MOVB	(R3),9\$::ECHO THE CHARACTER	
062446	104401	062514			TYPE	,\$9\$		
062452	122723	000015			CMPB	#15,(R3)+	::CHECK FOR RETURN	
062456	001305				BNE	2\$::LOOP IF NOT RETURN	
062460	105063	177777			CLRB	-1(R3)	::CLEAR RETURN (THE 15)	
062464	104401	001220			TYPE	,\$LF	::TYPE A LINE FEED	
062470	005726				TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK	
062472	012603				MOV	(SP)+,R3	::RESTORE R3	
062474	011646				MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE	
062476	016666	000004	000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT	
062504	012766	062516	000004		MOV	,\$STTYIN,4(SP)		
062512	000002				RTI		::RETURN	
062514	000			9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE	
062515	000				.BYTE	0	::TERMINATOR	
062516					.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT	
062526	136	103	015	\$TTYIN:	.ASCIZ	/'^C/<15><12>	::CONTROL 'C'	
062533	136	125	015	SCNTLC:	.ASCIZ	/'^U/<15><12>	::CONTROL 'U'	
062540	136	107	015	SCNTLG:	.ASCIZ	/'^G/<15><12>	::CONTROL 'G'	
062545	015	012	123	\$MSWR:	.ASCIZ	<15><12>/SWR = /		
062556	040	040	116	\$MNEW:	.ASCIZ	/ NEW = /		
					.EVEN			
2								
3	062570	012737	177777	001326	SHUT2:	MOV	#-1,CTFLG	::SET THE CONTROL-C FLAG
4	062576	105737	001116		TSTB	\$STNM	::DOING ANY TESTS ?	
5	062602	001002			BNE	1\$::BR IF YES	
6	062604	000137	031422		JMP	SHUT	::NO--RESPOND TO ^C	
7	062610	000002		1\$:	RTI		::EXIT FROM INTERRUPT	

1

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

 ::THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 ::CHANGE IT TO BINARY.
 ::CALL:
 ::* RDOCT ::READ AN OCTAL NUMBER
 ::* RETURN HERE ::LOW ORDER BITS ARE ON TOP OF THE STACK
 ::* ::HIGH ORDER BITS ARE IN \$HIOCT

```

062612 011646 000004 000002 $RDOCT: MOV (SP),-(SP) ::PROVIDE SPACE FOR THE
062614 016666 MOV 4(SP),2(SP) ::INPUT NUMBER
062622 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
062624 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
062626 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
062630 104412 1$: RDL IN ::READ AN ASCII LINE
062632 012600 MOV (SP)+,R0 ::GET ADDRESS OF 1ST CHARACTER
062634 005001 CLR R1 ::CLEAR DATA WORD
062636 005002 CLR R2
062640 112046 2$: MOV B (R0)+,-(SP) ::PICKUP THIS CHARACTER
062642 001412 BEQ 3$ ::IF ZERO GET OUT
062644 006301 ASL R1 ::*2
062646 006102 ROL R2
062650 006301 ASL R1 ::*4
062652 006102 ROL R2
062654 006301 ASL R1 ::*8
062656 006102 ROL R2
062660 042716 177770 BIC #^C7,(SP) ::STRIP THE ASCII JUNK
062664 062601 ADD (SP)+,R1 ::ADD IN THIS DIGIT
062666 000764 BR 2$ ::LOOP
062670 005726 3$: TST (SP)+ ::CLEAN TERMINATOR FROM STACK
062672 010166 MOV R1,12(SP) ::SAVE THE RESULT
062676 010237 062712 MOV R2,$HIOCT
062702 012602 MOV (SP)+,R2 ::POP STACK INTO R2
062704 012601 MOV (SP)+,R1 ::POP STACK INTO R1
062706 012600 MOV (SP)+,R0 ::POP STACK INTO R0
062710 000002 RTI ::RETURN
062712 000000 $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.

062714 016646 000002
 062720 042716 000020
 062724 012746 062732
 062730 000002
 062732 010046
 062734 016600 000002
 062740 005740
 062742 111000
 062744 006300
 062746 016000 062766
 062752 000200

```

$TRAP:  MOV  2(SP),-(SP)  ::ASSUME THE STATUS OF
        BIC  #20,(SP)    :: THE CALLER--DO NOT ALLOW
        MOV  #1$,-(SP)   :: T-BIT TRAPS
        RTI                          ::SET THE NEW STATUS
1$:     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

062754 011646
 062756 016666 000004 000002
 062764 000002

```

$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		

062766 062754	\$TRPAD: .WORD \$TRAP2	TRAP+1(104401)	TTY TYPEOUT ROUTINE
062770 056560	\$TYPE ::CALL=TYPE	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
062772 056356	\$TYPOC ::CALL=TYPOC	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
062774 056332	\$TYPOS ::CALL=TYPOS	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
062776 056372	\$TYPON ::CALL=TYPON	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
063000 056106	\$TYPDS ::CALL=TYPDS	TRAP+6(104406)	TYPE BINARY (ASCII) NUMBER
063002 056032	\$TYPBN ::CALL=TYPBN		
063004 061720	\$GTSWR ::CALL=GTSWR	TRAP+7(104407)	GET SOFT-SWR SETTING
063006 061630	\$CKSWR ::CALL=CKSWR	TRAP+10(104410)	TEST FOR CHANGE IN SOFT-SWR
063010 062172	\$RDCHR ::CALL=RDCHR	TRAP+11(104411)	TTY TYPEIN CHARACTER ROUTINE
063012 062262	\$RDLIN ::CALL=RDLIN	TRAP+12(104412)	TTY TYPEIN STRING ROUTINE
063014 062612	\$RDOCT ::CALL=RDOCT	TRAP+13(104413)	READ AN OCTAL NUMBER FROM TTY
063016 055736	\$SAVREG ::CALL=SAVREG	TRAP+14(104414)	SAVE R0-R5 ROUTINE
063020 055774	\$RESREG ::CALL=RESREG	TRAP+15(104415)	RESTORE R0-R5 ROUTINE

.SBTTL POWER DOWN AND UP ROUTINES

 :POWER DOWN ROUTINE

063022	012737	063162	000024	\$PWRDN: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST UP
063030	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
063036	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
063040	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
063042	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
063044	010346			MOV	R3,-(SP)	::PUSH R3 ON STACK
063046	010446			MOV	R4,-(SP)	::PUSH R4 ON STACK
063050	010546			MOV	R5,-(SP)	::PUSH R5 ON STACK
063052	017746	116076		MOV	@SWR,-(SP)	::PUSH @SWR ON STACK
063056	010637	063166		MOV	SP,\$SAVR6	::SAVE SP
063062	012737	063074	000024	MOV	#\$PWRUP,@#PWRVEC	::SET UP VECTOR
063070	000000			HALT		
063072	000776			BR	.-2	::HANG UP

 :POWER UP ROUTINE

063074	012737	063162	000024	\$PWRUP: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST DOWN
063102	013706	063166		MOV	\$SAVR6,SP	::GET SP
063106	005037	063166		CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
063112	005237	063166		1\$: INC	\$SAVR6	::WAIT FOR THE INC
063116	001375			BNE	1\$::OF WORD
063120	012677	116030		MOV	(SP)+,@SWR	::POP STACK INTO @SWR
063124	012605			MOV	(SP)+,R5	::POP STACK INTO R5
063126	012604			MOV	(SP)+,R4	::POP STACK INTO R4
063130	012603			MOV	(SP)+,R3	::POP STACK INTO R3
063132	012602			MOV	(SP)+,R2	::POP STACK INTO R2
063134	012601			MOV	(SP)+,R1	::POP STACK INTO R1
063136	012600			MOV	(SP)+,R0	::POP STACK INTO R0
063140	012737	063022	000024	MOV	#\$PWRDN,@#PWRVEC	::SET UP THE POWER DOWN VECTOR
063146	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
063154	104401			TYPE		::REPORT THE POWER FAILURE
063156	063170			\$PWRMG: .WORD	\$POWER	::POWER FAIL MESSAGE POINTER
063160	000002			RTI		
063162	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
063164	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
063166	000000			\$SAVR6: 0		::PUT THE SP HERE
063170	015	012	120	\$POWER: .ASCIZ	<15><12>'POWER'	
					.EVEN	

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
063200 112737 000001 063444 $ATY1: MOV #1,$FFLG ::TO REPORT FATAL ERROR
063206 112737 000001 063442 $ATY3: MOV #1,$MFLG ::TO TYPE A MESSAGE
063214 000403 BR $ATYC
063216 112737 000001 063444 $ATY4: MOV #1,$FFLG ::TO ONLY REPORT FATAL ERROR
063224 $ATYC:
063224 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
063226 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
063230 105737 063442 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
063234 001450 BEQ 5$ ::IF NOT: BR
063236 122737 000001 001242 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
063244 001031 BNE 3$ ::IF NOT: BR
063246 132737 000100 001243 BITB #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
063254 001425 BEQ 3$ ::IF NOT: BR
063256 017600 000004 MOV @4(SP),R0 ::GET MESSAGE ADDR.
063262 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
063270 005737 001222 1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
063274 001375 BNE 1$ ::IF NOT: WAIT
063276 010037 001236 MOV R0,$MSGAD ::PUT ADDR IN MAILBOX
063302 105720 2$: TSTB (R0)+ ::FIND END OF MESSAGE
063304 001376 BNE 2$
063306 163700 001236 SUB $MSGAD,R0 ::SUB START OF MESSAGE
063312 006200 ASR R0 ::GET MESSAGE LGTH IN WORDS
063314 010037 001240 MOV R0,$MSGGLT ::PUT LENGTH IN MAILBOX
063320 012737 000004 001222 MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
063326 000413 BR 5$
063330 017637 000004 063354 3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
063336 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
063344 013746 177776 MOV 177776,-(SP) ::PUSH 177776 ON STACK
063350 004737 056560 JSR PC,$TYPE ::CALL TYPE MACRO
063354 000000 4$: .WORD 0
063356 5$:
063356 105737 063444 10$: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
063362 001416 BEQ 12$ ::IF NOT: BR
063364 005737 001242 TST $ENV ::RUNNING UNDER APT?
063370 001413 BEQ 12$ ::IF NOT: BR
063372 005737 001222 11$: TST $MSGTYPE ::FINISHED LAST MESSAGE?
063376 001375 BNE 11$ ::IF NOT: WAIT
063400 017637 000004 001224 MOV @4(SP),$FATAL ::GET ERROR #
063406 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
063414 005237 001222 INC $MSGTYPE ::TELL APT TO TAKE ERROR
063420 105037 063444 12$: CLRB $FFLG ::CLEAR FATAL FLAG
063424 105037 063443 CLRB $LFLG ::CLEAR LOG FLAG
063430 105037 063442 CLRB $MFLG ::CLEAR MESSAGE FLAG
063434 012601 MOV (SP)+,R1 ::POP STACK INTO R1
063436 012600 MOV (SP)+,R0 ::POP STACK INTO R0
063440 000207 RTS PC ::RETURN
063442 000 $MFLG: .BYTE 0 ::MESSG. FLAG
063443 000 $LFLG: .BYTE 0 ::LOG FLAG
063444 000 $FFLG: .BYTE 0 ::FATAL FLAG
.EVEN
000200 APTSIZE = 200
000001 APTENV = 001
000100 APTPOOL = 100
000040 APTCSUP = 040
  
```


.SBTTL CONSOLE MESSAGES

2					
3	063446	200	103	101	SCTMSG: .ASCIZ <CRLF>@CANNOT RECOVER THE BAD SECTOR FILES ON THIS DEVICE@
4	063532	075	000		EQUALS: .ASCIZ @=@
5	063534	101	114	114	ALL: .ASCIZ @ALL@<CRLF>
6	063541	040	077	040	QUES: .ASCIZ @ ? @
7	063545	054	040	000	COMMA: .ASCIZ @, @
8	063550	200	124	117	MSHELP: .ASCII <CRLF>@TO ENSURE THAT NO BAD HEADERS ARE LEFT ON THE DISK@
9	063636	200	104	125	.ASCII <CRLF>@DURING ERROR FREE OPERATION, THIS PROGRAM SHOULD BE@
10	063724	200	102	105	.ASCII <CRLF>@BE HALTED BY TYPING A (^C) CONTROL C. AS A RESULT,@
11	064012	200	124	110	.ASCII <CRLF>@THE PROGRAM WILL BE HALTED WHEN THE DRIVE UNDER TEST@
12	064100	200	110	101	.ASCII <CRLF>@HAS COMPLETED TESTING.@
13	064127	200			.ASCII <CRLF>
14	064130	200	124	131	.ASCIZ <CRLF>@TYPE HELP TEXT (L) N ? @
15	064161	200	122	115	CNSLO1: .ASCIZ <CRLF>@RMCS1=@
16	064171	040	114	111	CNSLO2: .ASCIZ @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
17	064233	122	115	126	CNSLO3: .ASCIZ @RMVEC=@
18	064242	040	114	111	CNSLO4: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
19	064276	200	124	131	CNSLO7: .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
20	064363	200	101	116	.ASCIZ <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
21	064440	200			.ASCII <CRLF>
22	064441	040	077	111	CNSLO9: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
23	064462	200	104	122	MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
24	064476	104	122	111	MSGDRV: .ASCIZ /DRIVE/
25	064504	200	125	116	SYSTAT: .ASCIZ <CRLF>/UNIT STATUS: /
26	064522	122	115	070	\$RM80: .ASCIZ /RM80/
27	064527	040	116	117	NOTRM: .ASCIZ / NOT AN RM80/
28	064544	040	114	117	LODEV: .ASCIZ / LOAD DEVICE/
29	064561	040	116	117	NOTPRS: .ASCIZ / NOT PRESENT/
30	064576	040	116	117	NOTAVL: .ASCIZ / NOT AVAILABLE/
31	064615	040	117	106	UNTOFF: .ASCIZ / OFFLINE/
32	064626	040	117	116	UNTON: .ASCIZ / ONLINE/
33	064636	200	104	122	DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED/
34	064665	116	117	116	NONE: .ASCIZ /NONE/
35	064672	116	000		N: .ASCIZ /N/
36	064674	131	000		Y: .ASCIZ /Y/
37	064676	040			BLNKS4: .ASCII / /
38	064677	040			BLNKS3: .ASCII / /
39	064700	040			BLNKS2: .ASCII / /
40	064701	040	000		BLNKS1: .ASCIZ / /
41					.EVEN

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

.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.

FNCDTB: :FUNCTION CODE TABLE

.WORD OPI :NOP

064704

064704 020000

58	064706	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	064710	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	064712	130000	.WORD	ATA!OPI!IVC	:RECALIBRATE
61	064714	020000	.WORD	OPI	:DRIVE CLEAR
62	064716	030000	.WORD	OPI!IVC	:RELEASE
63	064720	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	064722	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	064724	020000	.WORD	OPI	:READ IN PRESET
66	064726	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	064730	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	064732	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	064734	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	064736	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	064740	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	064742	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	064744	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	064746	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	064750	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	064752	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	064754	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	064756	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	064760	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	064762	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	064764	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	064766	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	064770	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	064772	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	064774	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	064776	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	065000	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	065002	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)

1		
2		
3	065004	001
4	065005	002
5	065006	004
6	065007	010
7	065010	020
8	065011	040
9	065012	100
10	065013	200

.SBTTL ATTENTION (ATA) TABLE

ATNTBL:	.BYTE	1.
	.BYTE	2.
	.BYTE	4.
	.BYTE	8.
	.BYTE	16.
	.BYTE	32.
	.BYTE	64.
	.BYTE	128.

1
 2
 3 065014
 4 065014
 5 065014 000000
 6 065016 000001
 7 065020 000003
 8 065022 000007
 9 065024 000017
 10 065026 000037
 11 065030 000077
 12 065032 000177
 13 065034 000377
 14 065036 000777
 15 065040 001777
 16 065042 003777
 17 065044 007777
 18 065046 017777
 19 065050 037777
 20 065052 077777
 21 065054 177777
 22 065056 177777
 23 065060 077777
 24 065062 037777
 25 065064 017777
 26 065066 007777
 27 065070 003777
 28 065072 001777
 29 065074 000777
 30 065076 000377
 31 065100 000177
 32 065102 000077
 33 065104 000037
 34 065106 000017
 35 065110 000007
 36 065112 000003
 37 065114 000001
 38 065116 000000
 39 065120 000000
 40 065122 000001
 41 065124 000002
 42 065126 000004
 43 065130 000010
 44 065132 000020
 45 065134 000040
 46 065136 000100
 47 065140 000200
 48 065142 000400
 49 065144 001000
 50 065146 002000
 51 065150 004000
 52 065152 010000
 53 065154 020000
 54 065156 040000
 55 065160 100000
 56 065162 100000
 57 065164 040000

.SBTTL DATA PATTERN TABLE

RGDTPT:
 MIXED:

.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.
 ONES: .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.
 ZEROS: .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.
 .WORD 16384.

58	065166	020000	.WORD	8192.
59	065170	010000	.WORD	4096.
60	065172	004000	.WORD	2048.
61	065174	002000	.WORD	1024.
62	065176	001000	.WORD	512.
63	065200	000400	.WORD	256.
64	065202	000200	.WORD	128.
65	065204	000100	.WORD	64.
66	065206	000040	.WORD	32.
67	065210	000020	.WORD	16.
68	065212	000010	.WORD	8.
69	065214	000004	.WORD	4.
70	065216	000002	.WORD	2.
71	065220	000001	.WORD	1.
72	065222	000000	.WORD	0.
73	065224	177777	.WORD	65535.
74	065226	177776	.WORD	65534.
75	065230	177774	.WORD	65532.
76	065232	177770	.WORD	65528.
77	065234	177760	.WORD	65520.
78	065236	177740	.WORD	65504.
79	065240	177700	.WORD	65472.
80	065242	177600	.WORD	65408.
81	065244	177400	.WORD	65280.
82	065246	177000	.WORD	65024.
83	065250	176000	.WORD	64512.
84	065252	174000	.WORD	63488.
85	065254	170000	.WORD	61440.
86	065256	160000	.WORD	57344.
87	065260	140000	.WORD	49152.
88	065262	100000	.WORD	32768.
89	065264	000000	.WORD	0.
90	065266	000000	.WORD	0.
91	065270	100000	.WORD	32768.
92	065272	140000	.WORD	49152.
93	065274	160000	.WORD	57344.
94	065276	170000	.WORD	61440.
95	065300	174000	.WORD	63488.
96	065302	176000	.WORD	64512.
97	065304	177000	.WORD	65024.
98	065306	177400	.WORD	65280.
99	065310	177600	.WORD	65408.
100	065312	177700	.WORD	65472.
101	065314	177740	.WORD	65504.
102	065316	177760	.WORD	65520.
103	065320	177770	.WORD	65528.
104	065322	177774	.WORD	65532.
105	065324	177776	.WORD	65534.
106	065326	177777	.WORD	65535.
107	065330	125252	.WORD	43690.
108	065332	152525	.WORD	43690./2
109	065334	125252	.WORD	43690.
110	065336	177777	.WORD	65535.
111	065340	177776	.WORD	65534.
112	065342	177775	.WORD	65533.
113	065344	177773	.WORD	65531.
114	065346	177767	.WORD	65527.

EARLY:

115	065350	177757	.WORD	65519.
116	065352	177737	.WORD	65503.
117	065354	177677	.WORD	65471.
118	065356	177577	.WORD	65407.
119	065360	177377	.WORD	65279.
120	065362	176777	.WORD	65023.
121	065364	175777	.WORD	64511.
122	065366	173777	.WORD	63487.
123	065370	167777	.WORD	61439.
124	065372	157777	.WORD	57343.
125	065374	137777	.WORD	49151.
126	065376	077777	.WORD	32767.
127	065400	077777	.WORD	32767.
128	065402	137777	.WORD	49151.
129	065404	157777	.WORD	57343.
130	065406	167777	.WORD	61439.
131	065410	173777	.WORD	63487.
132	065412	175777	.WORD	64511.
133	065414	176777	.WORD	65023.
134	065416	177377	.WORD	65279.
135	065420	177577	.WORD	65407.
136	065422	177677	.WORD	65471.
137	065424	177737	.WORD	65503.
138	065426	177757	.WORD	65519.
139	065430	177767	.WORD	65527.
140	065432	177773	.WORD	65531.
141	065434	177775	.WORD	65533.
142	065436	177776	.WORD	65534.
143	065440	177777	.WORD	65535.
144	065442			

ENRGDT:

				.SBTTL	ERROR MESSAGE TABLE		
1							
2							
3	065442	072102	000000	EMT1:	.WORD	EMS1,0	
4	065446	072151	072166	000000	EMT2:	.WORD	EMS2,EMS3,0
5	065454	072151	072231	000000	EMT3:	.WORD	EMS2,EMS4,0
6	065462	072274	072324	000000	EMT4:	.WORD	EMS5,EMS6,0
7	065470	072274	072436	000000	EMT5:	.WORD	EMS5,EMS10,0
8	065476	077070	074317	000000	EMT6:	.WORD	EMS167,EMS64,0
9	065504	075065	077115	000000	EMT7:	.WORD	EMS110,EMS170,0
10	065512	072371	000000	EMT10:	.WORD	EMS7,0	
11	065516	072436	000000	EMT11:	.WORD	EMS10,0	
12	065522	072500	072511	000000	EMT12:	.WORD	EMS11,EMS12,0
13	065530	072552	072563	072574	EMT13:	.WORD	EMS13,EMS14,EMS15,EMS16,0
14	065542	072646	074317	000000	EMT14:	.WORD	EMS17,EMS64,0
15	065550	072500	072731	000000	EMT15:	.WORD	EMS11,EMS21,0
16	065556	072500	072754	073103	EMT16:	.WORD	EMS11,EMS22,EMS27,0
17	065566	072500	072770	073114	EMT17:	.WORD	EMS11,EMS23,EMS30,0
18	065576	072500	073016	073114	EMT20:	.WORD	EMS11,EMS24,EMS30,0
19	065606	072500	073045	073103	EMT21:	.WORD	EMS11,EMS25,EMS27,0
20	065616	072500	073062	073103	EMT22:	.WORD	EMS11,EMS26,EMS27,0
21	065626	072500	073124	073114	EMT23:	.WORD	EMS11,EMS31,EMS30,0
22	065636	072500	073153	073114	EMT24:	.WORD	EMS11,EMS32,EMS30,0
23	065646	072500	073202	073114	EMT25:	.WORD	EMS11,EMS33,EMS30,0
24	065656	072500	073230	073114	EMT26:	.WORD	EMS11,EMS34,EMS30,0
25	065666	072500	073301	073114	EMT27:	.WORD	EMS11,EMS35,EMS30,0
26	065676	072500	073330	073114	EMT30:	.WORD	EMS11,EMS36,EMS30,0
27	065706	072500	073357	073114	EMT31:	.WORD	EMS11,EMS37,EMS30,0
28	065716	072500	073405	073114	EMT32:	.WORD	EMS11,EMS40,EMS30,0
29	065726	072500	073434	073114	EMT33:	.WORD	EMS11,EMS41,EMS30,0
30	065736	072500	073462	073114	EMT34:	.WORD	EMS11,EMS42,EMS30,0
31	065746	072500	073511	073114	EMT35:	.WORD	EMS11,EMS43,EMS30,0
32	065756	072500	073540	073114	EMT36:	.WORD	EMS11,EMS44,EMS30,0
33	065766	072500	073613	073114	EMT37:	.WORD	EMS11,EMS45,EMS30,0
34	065776	074403	072706	000000	EMT40:	.WORD	EMS66,EMS20,0
35	066004	074571	076271	074577	EMT41:	.WORD	EMS75,EMS141,EMS76,0
36	066014	076362	076372	074525	EMT42:	.WORD	EMS144,EMS145,EMS72,EMS76,0
37	066026	073705	074023	074577	EMT43:	.WORD	EMS47,EMS53,EMS76,0
38	066036	074630	074023	074577	EMT44:	.WORD	EMS77,EMS53,EMS76,0
39	066046	074656	074023	074577	EMT45:	.WORD	EMS100,EMS53,EMS76,0
40	066056	074704	074023	074577	EMT46:	.WORD	EMS101,EMS53,EMS76,0
41	066066	074335	074317	000000	EMT47:	.WORD	EMS65,EMS64,0
42	066074	072552	072574	074271	EMT50:	.WORD	EMS13,EMS15,EMS63,0
43	066104	072500	073656	073114	EMT51:	.WORD	EMS11,EMS46,EMS30,EMS67,0
44	066116	073705	074023	074453	EMT52:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
45	066134	073705	074023	074453	EMT53:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
46	066152	073734	074023	074453	EMT54:	.WORD	EMS50,EMS53,EMS67,0
47	066162	076317	076334	074023	EMT55:	.WORD	EMS142,EMS143,EMS53,EMS67,0
48	066174	073763	074525	074453	EMT56:	.WORD	EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
49	066212	077070	074535	074453	EMT57:	.WORD	EMS167,EMS73,EMS67,0
50	066222	074106	074453	075265	EMT60:	.WORD	EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
51	066240	074512	074106	074453	EMT61:	.WORD	EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
52	066260	076250	074453	075265	EMT62:	.WORD	EMS140,EMS67,EMS115,EMS47,EMS70,0
53	066274	100270	100313	072706	EMT63:	.WORD	EMS225,EMS226,EMS20,0
54	066304	076455	076520	074500	EMT64:	.WORD	EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
55	066324	074011	077515	077676	EMT65:	.WORD	EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
56	066344	077027	074761	074525	EMT66:	.WORD	EMS165,EMS103,EMS72,EMS124,0
57	066356	077027	074761	074525	EMT67:	.WORD	EMS165,EMS103,EMS72,EMS171,0

58	066370	073656	072706	076722	EMT70:	.WORD	EMS46,EMS20,EMS163,0
59	066400	074512	074704	076722	EMT71:	.WORD	EMS71,EMS101,EMS163,0
60	066410	073705	074525	076722	EMT72:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS140,EMS141,0
61	066426	073705	074525	076722	EMT73:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS141,EMS72,0
62	066444	074106	074023	076722	EMT74:	.WORD	EMS56,EMS53,EMS163,0
63	066454	074512	074106	076722	EMT75:	.WORD	EMS71,EMS56,EMS163,EMS115,EMS150,EMS152,EMS72,0
64	066474	073734	074023	076722	EMT76:	.WORD	EMS50,EMS53,EMS163,0
65	066504	076317	076334	074023	EMT77:	.WORD	EMS142,EMS143,EMS53,EMS163,0
66	066516	074571	076271	076722	EMT100:	.WORD	EMS75,EMS141,EMS163,EMS115,EMS47,EMS70,0
67	066534	074571	076455	076722	EMT101:	.WORD	EMS75,EMS150,EMS163,EMS115,EMS56,EMS73,0
68	066552	077070	074535	076722	EMT102:	.WORD	EMS167,EMS73,EMS163,0
69	066562	076440	076474	074552	EMT103:	.WORD	EMS147,EMS151,EMS74,EMS163,0
70	066574	073763	074552	076722	EMT104:	.WORD	EMS51,EMS74,EMS163,EMS115,EMS50,EMS70,0
71	066612	077027	074656	076722	EMT105:	.WORD	EMS165,EMS100,EMS163,0
72	066622	077027	074630	076722	EMT106:	.WORD	EMS165,EMS77,EMS163,0
73	066632	076362	076372	074023	EMT107:	.WORD	EMS144,EMS145,EMS53,EMS163,EMS115,EMS143,EMS70,0
74	066652	075065	075125	075270	EMT110:	.WORD	EMS110,EMS112,EMS116,EMS111,0
75	066664	075211	072231	000000	EMT111:	.WORD	EMS113,EMS4,0
76	066672	075211	072166	000000	EMT112:	.WORD	EMS113,EMS3,0
77	066700	075065	075265	075270	EMT113:	.WORD	EMS110,EMS115,EMS116,EMS117,EMS114,0
78	066714	000000			EMT114:	.WORD	
79	066716	075342	076002	076026	EMT115:	.WORD	EMS121,EMS132,EMS133,0
80	066726	075405	076002	076026	EMT116:	.WORD	EMS122,EMS132,EMS133,0
81	066736	075442	076002	076026	EMT117:	.WORD	EMS123,EMS132,EMS133,0
82	066746	075505	076002	076026	EMT120:	.WORD	EMS124,EMS132,EMS133,0
83	066756	075537	076002	076026	EMT121:	.WORD	EMS125,EMS132,EMS133,0
84	066766	075602	076002	076026	EMT122:	.WORD	EMS126,EMS132,EMS133,0
85	066776	076213	076002	076026	EMT123:	.WORD	EMS137,EMS132,EMS133,0
86	067006	075704	076002	076026	EMT124:	.WORD	EMS130,EMS132,EMS133,0
87	067016	075742	076002	076026	EMT125:	.WORD	EMS131,EMS132,EMS133,0
88	067026	075342	076002	076051	EMT126:	.WORD	EMS121,EMS132,EMS134,EMS123,0
89	067040	075405	076002	076051	EMT127:	.WORD	EMS122,EMS132,EMS134,EMS123,0
90	067052	075442	076002	076051	EMT130:	.WORD	EMS123,EMS132,EMS134,EMS123,0
91	067064	075505	076002	076051	EMT131:	.WORD	EMS124,EMS132,EMS134,EMS123,0
92	067076	075537	076002	076051	EMT132:	.WORD	EMS125,EMS132,EMS134,EMS123,0
93	067110	075602	076002	076051	EMT133:	.WORD	EMS126,EMS132,EMS134,EMS123,0
94	067122	076213	076002	076051	EMT134:	.WORD	EMS137,EMS132,EMS134,EMS123,0
95	067134	075704	076002	076051	EMT135:	.WORD	EMS130,EMS132,EMS134,EMS123,0
96	067146	075742	076002	076051	EMT136:	.WORD	EMS131,EMS132,EMS134,EMS123,0
97	067160	075342	076002	076113	EMT137:	.WORD	EMS121,EMS132,EMS135,0
98	067170	075442	076002	076113	EMT140:	.WORD	EMS123,EMS132,EMS135,0
99	067200	075342	076002	076166	EMT141:	.WORD	EMS121,EMS132,EMS136,0
100	067210	076213	076002	076166	EMT142:	.WORD	EMS137,EMS132,EMS136,0
101	067220	075505	076002	076166	EMT143:	.WORD	EMS124,EMS132,EMS136,0
102	067230	075537	076002	076166	EMT144:	.WORD	EMS125,EMS132,EMS136,0
103	067240	075602	076002	076166	EMT145:	.WORD	EMS126,EMS132,EMS136,0
104	067250	075742	076002	076166	EMT146:	.WORD	EMS131,EMS132,EMS136,0
105	067260	077144	076002	076166	EMT147:	.WORD	EMS171,EMS132,EMS136,0
106	067270	075704	076002	076166	EMT150:	.WORD	EMS130,EMS132,EMS136,0
107	067300	076250	075265	076271	EMT151:	.WORD	EMS140,EMS115,EMS141,EMS70,0
108	067312	076317	075265	076334	EMT152:	.WORD	EMS142,EMS115,EMS143,EMS72,0
109	067324	076362	076372	076421	EMT153:	.WORD	EMS144,EMS145,EMS146,EMS115,EMS143,EMS72,0
110	067342	076362	076372	072706	EMT154:	.WORD	EMS144,EMS145,EMS20,EMS115,EMS143,EMS70,0
111	067360	076455	076520	076566	EMT155:	.WORD	EMS150,EMS152,EMS154,EMS153,0
112	067372	076440	076474	076566	EMT156:	.WORD	EMS147,EMS151,EMS154,EMS155,0
113	067404	076440	076474	076622	EMT157:	.WORD	EMS147,EMS151,EMS156,EMS157,0
114	067416	076672	076622	076655	EMT160:	.WORD	EMS161,EMS156,EMS160,0

115	067426	073045	073103	076622	EMT161: .WORD	EMS25,EMS27,EMS156,EMS160,0
116	067440	073062	073103	076622	EMT162: .WORD	EMS26,EMS27,EMS156,EMS160,0
117	067452	073705	076722	076250	EMT163: .WORD	EMS47,EMS163,EMS140,0
118	067462	073705	072706	000000	EMT164: .WORD	EMS47,EMS20,0
119	067470	074106	072706	000000	EMT165: .WORD	EMS56,EMS20,0
120	067476	073656	072706	000000	EMT166: .WORD	EMS46,EMS20,0
121	067504	100216	072706	000000	EMT167: .WORD	EMS224,EMS20,0
122	067512	077454	076271	073114	EMT170: .WORD	EMS203,EMS141,EMS30,EMS202,0
123	067524	077070	076566	076640	EMT171: .WORD	EMS167,EMS154,EMS157,0
124	067534	077070	076566	076602	EMT172: .WORD	EMS167,EMS154,EMS155,0
125	067544	077144	076002	076026	EMT173: .WORD	EMS171,EMS132,EMS133,0
126	067554	077144	076002	076051	EMT174: .WORD	EMS171,EMS132,EMS134,EMS123,0
127	067566	075211	077317	000000	EMT175: .WORD	EMS113,EMS177,0
128	067574	077341	077356	000000	EMT176: .WORD	EMS200,EMS201,0
129	067602	000000			EMT177: .WORD	
130	067604	077454	073763	073114	EMT200: .WORD	EMS203,EMS51,EMS30,EMS202,0
131	067616	077454	077467	073114	EMT201: .WORD	EMS203,EMS204,EMS30,EMS202,0
132	067630	077454	073734	073114	EMT202: .WORD	EMS203,EMS50,EMS30,EMS202,0
133	067642	077454	076334	073114	EMT203: .WORD	EMS203,EMS143,EMS30,EMS202,0
134	067654	076455	074552	077424	EMT204: .WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
135	067672	073734	074761	074525	EMT205: .WORD	EMS50,EMS103,EMS72,0
136	067702	074770	074535	077424	EMT206: .WORD	EMS104,EMS73,EMS202,0
137	067712	074571	076271	075265	EMT207: .WORD	EMS75,EMS141,EMS115,EMS140,0
138	067724	074571	076455	000000	EMT210: .WORD	EMS75,EMS150,0
139	067732	073763	074525	075265	EMT211: .WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
140	067746	076317	074023	075265	EMT212: .WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
141	067762	073734	074761	074023	EMT213: .WORD	EMS50,EMS103,EMS53,0
142	067772	074011	077515	077043	EMT214: .WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
143	070012	074011	075320	074106	EMT215: .WORD	EMS52,EMS117,EMS56,EMS163,0
144	070024	074106	073114	074317	EMT216: .WORD	EMS56,EMS30,EMS64,0
145	070034	073656	073114	074317	EMT217: .WORD	EMS46,EMS30,EMS64,0
146	070044	073124	073114	074317	EMT220: .WORD	EMS31,EMS30,EMS64,0
147	070054	073705	073114	074317	EMT221: .WORD	EMS47,EMS30,EMS64,0
148	070064	074011	075320	074630	EMT222: .WORD	EMS52,EMS117,EMS77,0
149	070074	074011	075320	074271	EMT223: .WORD	EMS52,EMS117,EMS63,0
150	070104	074011	075320	100270	EMT224: .WORD	EMS52,EMS117,EMS225,EMS115,EMS226,0
151	070120	100270	100342	100353	EMT225: .WORD	EMS225,EMS227,EMS230,EMS115,EMS156,EMS231,EMS57,0
152	070140	072500	100353	073114	EMT226: .WORD	EMS11,EMS230,EMS30,0
153	070150	100270	100313	075265	EMT227: .WORD	EMS225,EMS226,EMS115,EMS230,EMS72,0
154	070164	000000			EMT230: .WORD	
155	070166	000000			EMT231: .WORD	
156	070170	000000			EMT232: .WORD	
157	070172	000000			EMT233: .WORD	
158	070174	000000			EMT234: .WORD	
159	070176	000000			EMT235: .WORD	
160	070200	000000			EMT236: .WORD	
161	070202	000000			EMT237: .WORD	
162	070204	000000			EMT240: .WORD	
163	070206	000000			EMT241: .WORD	
164	070210	000000			EMT242: .WORD	
165	070212	000000			EMT243: .WORD	
166	070214	000000			EMT244: .WORD	
167	070216	000000			EMT245: .WORD	
168	070220	077070	076002	077554	EMT246: .WORD	EMS167,EMS132,EMS207,0
169	070230	077070	076002	077601	EMT247: .WORD	EMS167,EMS132,EMS210,EMS125,0
170	070242	077070	077612	077601	EMT250: .WORD	EMS167,EMS211,EMS210,EMS207,EMS206,0
171	070256	077630	077653	000000	EMT251: .WORD	EMS212,EMS213,0

172	070264	077027	077630	000000	EMT252: .WORD	EMS165,EMS212,0
173	070272	076440	076474	076622	EMT253: .WORD	EMS147,EMS151,EMS156,EMS210,EMS26,EMS27,0
174	070310	077454	074704	073114	EMT254: .WORD	EMS203,EMS101,EMS30,0
175	070320	077454	077070	073114	EMT255: .WORD	EMS203,EMS167,EMS30,0
176	070330	077454	074656	073114	EMT256: .WORD	EMS203,EMS100,EMS30,0
177	070340	072500	073656	073114	EMT257: .WORD	EMS11,EMS46,EMS30,EMS102,0
178	070352	074011	075320	074106	EMT260: .WORD	EMS52,EMS117,EMS56,EMS102,0
179	070364	074011	077515	100020	EMT261: .WORD	EMS52,EMS205,EMS220,EMS206,EMS115,EMS51,EMS72,0
180	070404	074770	074535	077424	EMT262: .WORD	EMS104,EMS73,EMS202,0
181	070414	073734	074023	074732	EMT263: .WORD	EMS50,EMS53,EMS102,0
182	070424	074106	074023	074732	EMT264: .WORD	EMS56,EMS53,EMS102,EMS115,EMS150,EMS152,EMS70,0
183	070444	074512	074106	074732	EMT265: .WORD	EMS71,EMS56,EMS102,EMS115,EMS150,EMS152,EMS72,0
184	070464	076317	076334	074023	EMT266: .WORD	EMS142,EMS143,EMS53,EMS102,0
185	070476	073734	076421	075265	EMT267: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,0
186	070514	073705	074023	074732	EMT270: .WORD	EMS47,EMS53,EMS102,EMS115,EMS140,0
187	070530	073705	074023	074732	EMT271: .WORD	EMS47,EMS53,EMS102,EMS115,EMS141,EMS72,0
188	070546	074571	076271	074732	EMT272: .WORD	EMS75,EMS141,EMS102,EMS115,EMS47,EMS73,0
189	070564	073763	074552	074732	EMT273: .WORD	EMS51,EMS74,EMS102,EMS115,EMS50,EMS70,0
190	070602	074271	074023	074163	EMT274: .WORD	EMS63,EMS53,EMS57,EMS115,EMS41,EMS146,0
191	070620	075270	074023	073434	EMT275: .WORD	EMS116,EMS53,EMS41,EMS57,0
192	070632	073705	074023	074163	EMT276: .WORD	EMS47,EMS53,EMS57,EMS115,EMS140,0
193	070646	073705	074023	074163	EMT277: .WORD	EMS47,EMS53,EMS57,EMS115,EMS141,EMS72,0
194	070664	074106	074023	074163	EMT300: .WORD	EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS70,0
195	070704	074512	074106	074023	EMT301: .WORD	EMS71,EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS72,0
196	070726	077027	074656	074761	EMT302: .WORD	EMS165,EMS100,EMS103,EMS57,0
197	070740	077027	074704	074761	EMT303: .WORD	EMS165,EMS101,EMS103,EMS57,0
198	070752	077027	074630	074761	EMT304: .WORD	EMS165,EMS77,EMS103,EMS57,0
199	070764	073656	073114	074317	EMT305: .WORD	EMS46,EMS30,EMS64,EMS57,0
200	070776	073734	074023	074163	EMT306: .WORD	EMS50,EMS53,EMS57,0
201	071006	073734	076421	075265	EMT307: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,EMS57,0
202	071026	076317	076334	074023	EMT310: .WORD	EMS142,EMS143,EMS53,EMS57,0
203	071040	074770	074761	074023	EMT311: .WORD	EMS104,EMS103,EMS53,EMS57,0
204	071052	075017	074761	074023	EMT312: .WORD	EMS105,EMS103,EMS53,EMS57,0
205	071064	076362	076372	074761	EMT313: .WORD	EMS144,EMS145,EMS103,EMS57,EMS115,EMS143,EMS70,0
206	071104	073462	074761	074023	EMT314: .WORD	EMS42,EMS103,EMS53,EMS57,0
207	071116	073124	074761	074023	EMT315: .WORD	EMS31,EMS103,EMS53,EMS57,0
208	071130	074512	073124	074761	EMT316: .WORD	EMS71,EMS31,EMS103,EMS57,0
209	071142	073511	074761	074163	EMT317: .WORD	EMS43,EMS103,EMS57,0
210	071152	073613	074761	074163	EMT320: .WORD	EMS45,EMS103,EMS57,0
211	071162	073540	074761	074163	EMT321: .WORD	EMS44,EMS103,EMS57,0
212	071172	075046	072706	000000	EMT322: .WORD	EMS106,EMS20,0
213	071200	073330	074761	074163	EMT323: .WORD	EMS36,EMS103,EMS57,0
214	071210	077217	073330	074761	EMT324: .WORD	EMS173,EMS36,EMS103,EMS57,0
215	071222	077177	073330	074761	EMT325: .WORD	EMS172,EMS36,EMS103,EMS57,0
216	071234	072552	077234	072574	EMT326: .WORD	EMS13,EMS174,EMS15,EMS35,EMS53,EMS175,0
217	071252	076440	076474	074552	EMT327: .WORD	EMS147,EMS151,EMS74,EMS175,0
218	071264	074403	074023	077242	EMT330: .WORD	EMS66,EMS53,EMS175,0
219	071274	073202	074761	074023	EMT331: .WORD	EMS33,EMS103,EMS53,EMS175,0
220	071306	073405	074761	074023	EMT332: .WORD	EMS40,EMS103,EMS53,EMS57,0
221	071320	073763	074552	074163	EMT333: .WORD	EMS51,EMS74,EMS57,EMS115,EMS50,EMS70,0
222	071336	074571	076271	074163	EMT334: .WORD	EMS75,EMS141,EMS57,EMS115,EMS47,EMS73,0
223	071354	074571	076455	076520	EMT335: .WORD	EMS75,EMS150,EMS152,EMS57,EMS115,EMS56,EMS73,0
224	071374	074211	074224	074253	EMT336: .WORD	EMS60,EMS61,EMS62,0
225	071404	075270	075320	073230	EMT337: .WORD	EMS116,EMS117,EMS34,0
226	071414	073230	074023	074035	EMT340: .WORD	EMS34,EMS53,EMS54,EMS111,0
227	071426	074057	073230	000000	EMT341: .WORD	EMS55,EMS34,0
228	071434	074011	075320	074106	EMT342: .WORD	EMS52,EMS117,EMS56,EMS57,0

1	071600	100445	101251	101326	EHT1:	.WORD	EH1,STSH1,STSH2,STSH4,0
2	071612	101251	101326	101453	EHT2:	.WORD	STSH1,STSH2,STSH4,0
3							
4	071622	100464	000000		EHT110:	.WORD	EH110,0
5	071626	100473	000000		EHT111:	.WORD	EH111,0
6							
7	071632	100512	000000		EHT114:	.WORD	EH114,0
8	071636	100541	101251	101326	EHT223:	.WORD	EH223,STSH1,STSH2,STSH4,0
9	071650	100567	101251	101326	EHT256:	.WORD	EH256,STSH1,STSH2,STSH4,0
10							
11	071662	100643	101251	101326	EHT336:	.WORD	EH336,STSH1,STSH2,STSH4,0
12	071674	100702	101251	101326	EHT337:	.WORD	EH337,STSH1,STSH2,STSH4,0
13	071706	101037	101251	101326	EHT344:	.WORD	EH344,STSH1,STSH2,STSH4,0
14							
15	071720	101175	000000		EHT353:	.WORD	EH353,0

1	071724	101512	101606	101624	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
2	071734	101606	101624	101656	EDT2:	.WORD	STSD1,STSD2,STSD4
3							
4	071742	101520			EDT110:	.WORD	ED110
5	071744	101524			EDT111:	.WORD	ED111
6							
7	071746	101532			EDT114:	.WORD	ED114
8	071750	101542	101606	101624	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
9							
10	071760	101552	101606	101624	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
11	071770	101564	101606	101624	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
12	072000	101564	101606	101624	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
13							
14	072012	101576			EDT353:	.WORD	ED353

1	072014	101671	101707	101707	EFT1:	.WORD	EF111,STSF,STSF,STSF
2	072024	101707	101707	101707	EFT2:	.WORD	STSF,STSF,STSF
3							
4	072032	101670			EFT110:	.WORD	EF110
5	072034	101671			EFT111:	.WORD	EF111
6							
7	072036	101673			EFT114:	.WORD	EF114
8	072040	101673	101707	101707	EFT223:	.WORD	EF114,STSF,STSF,STSF
9							
10	072050	101676	101707	101707	EFT336:	.WORD	EF336,STSF,STSF,STSF
11	072060	101676	101707	101707	EFT337:	.WORD	EF336,STSF,STSF,STSF
12	072070	101676	101707	101707	EFT344:	.WORD	EF336,STSF,STSF,STSF
13							
14	072100	101673			EFT353:	.WORD	EF114

				.SBTTL	ERROR MESSAGE STRINGS
1					
2					
3	072102	127	122	117	EMS1: .ASCIZ @WRONG UNIT SELECTED (RMCS2, BITS 0-2) a
4	072151	104	105	126	EMS2: .ASCIZ @DEVICE WENT a
5	072166	125	116	101	EMS3: .ASCIZ @UNAVAILABLE 'DVA' (RMCS1, BIT 11) a
6	072231	116	117	116	EMS4: .ASCIZ @NONEXISTENT 'NED' (RMCS2, BIT 12) a
7	072274	103	117	115	EMS5: .ASCIZ @COMMAND NOT COMPLETED, a
8	072324	103	117	116	EMS6: .ASCIZ @CONTROLLER NOT READY (RMCS1, BIT 7) a
9	072371	104	122	111	EMS7: .ASCIZ @DRIVE NOT READY 'DRY' (RMDS, BIT 7) a
10	072436	107	117	040	EMS10: .ASCIZ @GO NOT RESET 'GO' (RMCS1, BIT 0) a
11	072500	111	116	126	EMS11: .ASCIZ @INVALID a
12	072511	106	125	116	EMS12: .ASCIZ @FUNCTION CODE (RMCS1, BITS 1-5) a
13	072552	115	101	123	EMS13: .ASCIZ @MASSBUS a
14	072563	103	117	116	EMS14: .ASCIZ @CONTROL a
15	072574	102	125	123	EMS15: .ASCIZ @BUS PARITY ERROR a
16	072616	042	115	103	EMS16: .ASCIZ @'MCPE' (RMCS1, BIT 13) a
17	072646	124	122	101	EMS17: .ASCIZ @TRANSFER ERROR (RMCS1, BIT 14) a
18	072706	123	110	117	EMS20: .ASCIZ @SHOULD NOT BE SET a
19	072731	127	117	122	EMS21: .ASCIZ @WORD COUNT (RMWC) a
20	072754	102	125	123	EMS22: .ASCIZ @BUS (RMB) a
21	072770	042	114	102	EMS23: .ASCIZ @'LBT' (RMDS, BIT 10) a
22	073016	042	101	117	EMS24: .ASCIZ @'AOE' (RMER1, BIT 09) a
23	073045	104	111	123	EMS25: .ASCIZ @DISK (RMD) a
24	073062	103	131	114	EMS26: .ASCIZ @CYLINDER (RMDC) a
25	073103	101	104	104	EMS27: .ASCIZ @ADDRESS a
26	073114	123	124	101	EMS30: .ASCIZ @STATUS a
27	073124	042	127	114	EMS31: .ASCIZ @'WLE' (RMER1, BIT 11) a
28	073153	042	125	120	EMS32: .ASCIZ @'UPE' (RMCS2, BIT 13) a
29	073202	042	127	103	EMS33: .ASCIZ @'WCF' (RMER1, BIT 5) a
30	073230	127	122	111	EMS34: .ASCIZ @WRITE CHECK ERROR-'WCE' (RMCS2, BIT 14) a
31	073301	042	115	104	EMS35: .ASCIZ @'MDPE' (RMCS2, BIT 8) a
32	073330	042	104	103	EMS36: .ASCIZ @'DCK' (RMER1, BIT 15) a
33	073357	042	105	103	EMS37: .ASCIZ @'ECH' (RMER1, BIT 6) a
34	073405	042	104	114	EMS40: .ASCIZ @'DLT' (RMCS2, BIT 15) a
35	073434	042	115	130	EMS41: .ASCIZ @'MXF' (RMCS2, BIT 9) a
36	073462	042	104	124	EMS42: .ASCIZ @'DTE' (RMER1, BIT 12) a
37	073511	042	110	103	EMS43: .ASCIZ @'HCRC' (RMER1, BIT 8) a
38	073540	110	105	101	EMS44: .ASCIZ @HEADER COMPARE ERROR 'HCE' (RMER1, BIT 7) a
39	073613	106	117	122	EMS45: .ASCIZ @FORMAT ERROR 'FER' (RMER1, BIT 4) a
40	073656	042	111	101	EMS46: .ASCIZ @'IAE' (RMER1, BIT 10) a
41	073705	042	117	120	EMS47: .ASCIZ @'DPI' (RMER1, BIT 13) a
42	073734	042	123	113	EMS50: .ASCIZ @'SKI' (RMER2, BIT 14) a
43	073763	042	120	111	EMS51: .ASCIZ @'PIP' (RMDS, BIT 13) a
44	074011	124	110	105	EMS52: .ASCIZ @THE RM80 a
45	074023	104	105	124	EMS53: .ASCIZ @DETECTED a
46	074035	101	124	040	EMS54: .ASCIZ @AT AN UNEXPECTED a
47	074057	111	116	103	EMS55: .ASCIZ @INCORRECT DATA DURING a
48	074106	111	116	126	EMS56: .ASCIZ @INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
49	074163	104	125	122	EMS57: .ASCIZ @DURING DATA TRANSFER a
50	074211	104	101	124	EMS60: .ASCIZ @DATA READ a
51	074224	104	117	105	EMS61: .ASCIZ @DOES NOT COMPARE WITH a
52	074253	104	101	124	EMS62: .ASCIZ @DATA WRITTEN a
53	074271	042	120	101	EMS63: .ASCIZ @'PAR' (RMER1, BIT 3) a
54	074317	111	123	040	EMS64: .ASCIZ @IS INCORRECT a
55	074335	103	117	115	EMS65: .ASCIZ @COMPOSITE ERROR 'ERR' (RMDS, BIT 14) a
56	074403	104	101	124	EMS66: .ASCIZ @DATA PARITY ERROR 'DPE' (RMER2, BIT 3) a
57	074453	104	125	122	EMS67: .ASCIZ @DURING SEEK COMMAND a

58	074500	111	123	040	EMS70:	.ASCIZ	@IS RESET @
59	074512	105	122	122	EMS71:	.ASCIZ	@ERRONEOUS @
60	074525	111	123	040	EMS72:	.ASCIZ	@IS SET @
61	074535	104	111	104	EMS73:	.ASCIZ	@DID NOT SET @
62	074552	104	111	104	EMS74:	.ASCIZ	@DID NOT RESET @
63	074571	114	117	123	EMS75:	.ASCIZ	@LOST @
64	074577	104	125	122	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
65	074630	042	122	115	EMS77:	.ASCIZ	@'RMR' (RMER1, BIT 2) @
66	074656	042	111	114	EMS100:	.ASCIZ	@'ILR' (RMER1, BIT 1) @
67	074704	042	111	114	EMS101:	.ASCIZ	@'ILF' (RMER1, BIT 0) @
68	074732	104	125	122	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
69	074761	105	122	122	EMS103:	.ASCIZ	@ERROR @
70	074770	042	114	102	EMS104:	.ASCIZ	@'LBC' (RMER2, BIT 10) @
71	075017	042	114	123	EMS105:	.ASCIZ	@'LSC' (RMER2, BIT 11) @
72	075046	110	105	101	EMS106:	.ASCIZ	@HEADER ERRORS @
73	075065	102	125	123	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
74	075114	101	104	104	EMS111:	.ASCIZ	@ADDRESS @
75	075125	127	110	105	EMS112:	.ASCIZ	@WHEN READING/WRITING RH REGISTERS @
76	075167	101	124	040		.ASCIZ	@AT THE FOLLOWING @
77	075211	124	110	105	EMS113:	.ASCIZ	@THE SELECTED DEVICE IS @
78	075241	116	117	116	EMS114:	.ASCIZ	@NONEXISTENT DEVICE @
79	075265	015	012	000	EMS115:	.ASCIZ	<CR><LF>
80	075270	124	110	105	EMS116:	.ASCIZ	@THE MASSBUS CONTROLLER @
81	075320	106	101	111	EMS117:	.ASCIZ	@FAILED TO DETECT @
82	075342				EMS120:		:NOT USED
83	075342	103	117	116	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
84	075405	102	125	123	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMBA, @
85	075442	103	117	116	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
86	075505	105	122	122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMER1, @
87	075537	101	124	124	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMAS, @
88	075602	115	101	111	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMMR #1, @
89	075645	105	103	103	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
90	075704	105	103	103	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2, @
91	075742	115	101	111	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMMR2, @
92	076002	116	117	124	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
93	076026	125	116	111	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
94	076051	103	117	116	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
95	076113	122	110	040	EMS135:	.ASCIZ	@RH CONTROLLER ERROR CLEAR (RMCS1, BIT 14) @
96	076166	104	122	111	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
97	076213	104	122	111	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMD5 @
98	076250	115	105	104	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
99	076271	042	115	117	EMS141:	.ASCIZ	@'MOL' (RMD5, BIT 12) @
100	076317	104	122	111	EMS142:	.ASCIZ	@DRIVE FAULT @
101	076334	042	104	126	EMS143:	.ASCIZ	@'DVC' (RMER2, BIT 7) @
102	076362	125	116	123	EMS144:	.ASCIZ	@UNSAFE @
103	076372	042	125	116	EMS145:	.ASCIZ	@'UNS' (RMER1, BIT 14) @
104	076421	123	110	117	EMS146:	.ASCIZ	@SHOULD BE SET @
105	076440	117	106	106	EMS147:	.ASCIZ	@OFFSET MODE @
106	076455	040	126	117	EMS150:	.ASCIZ	@ VOLUME VALID @
107	076474	042	117	115	EMS151:	.ASCIZ	@'OM' (RMD5, BIT 0) @
108	076520	042	126	126	EMS152:	.ASCIZ	@'VV' (RMD5, BIT 6) @
109	076544	120	101	103	EMS153:	.ASCIZ	@PACK ACK COMMAND @
110	076566	116	117	124	EMS154:	.ASCIZ	@NOT SET BY @
111	076602	117	106	106	EMS155:	.ASCIZ	@OFFSET COMMAND @
112	076622	116	117	124	EMS156:	.ASCIZ	@NOT RESET BY @
113	076640	122	124	103	EMS157:	.ASCIZ	@RTC COMMAND @
114	076655	122	111	120	EMS160:	.ASCIZ	@RIP COMMAND @

115 076672	117	106	106	EMS161: .ASCIZ	@OFFSET REGISTER (RMOF) @
116 076722				EMS162:	:<UNUSED>
117 076722	104	125	122	EMS163: .ASCIZ	@DURING RECALIBRATE @
118 076746	111	123	040	EMS164: .ASCII	@IS INTERMITTENT OR DRIVE DIDNT DROP ON @
119 077015	103	131	114	.ASCIZ	@CYLINDER @
120 077027	125	116	105	EMS165: .ASCIZ	@UNEXPECTED @
121 077043	122	105	103	EMS166: .ASCIZ	@RECALIBRATE COMMAND @
122 077070	042	101	124	EMS167: .ASCIZ	@'ATA' (RMDS, BIT15) @
123 077115	127	110	105	EMS170: .ASCIZ	@WHEN READING REGISTER @
124 077144	105	122	122	EMS171: .ASCIZ	@ERROR REGISTER #2, RMER2, @
125 077177	116	117	116	EMS172: .ASCIZ	@NONRECOVERABLE @
126 077217	122	105	103	EMS173: .ASCIZ	@RECOVERABLE @
127 077234	104	101	124	EMS174: .ASCIZ	@DATA @
128 077242	104	125	122	EMS175: .ASCIZ	@DURING WRITE COMMAND @
129 077270	042	117	120	EMS176: .ASCIZ	@'OPE' (RMER2, BIT 13) @
130 077317	111	116	040	EMS177: .ASCIZ	@IN WRITE PROTECT @
131 077341	103	101	116	EMS200: .ASCIZ	@CAN NOT SET @
132 077356	104	111	101	EMS201: .ASCIZ	@DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 0) @
133 077424	104	125	122	EMS202: .ASCIZ	@DURING DIAGNOSTIC MODE @
134 077454	111	116	103	EMS203: .ASCIZ	@INCORRECT @
135 077467	042	127	122	EMS204: .ASCIZ	@'WRL' (RMDS, BIT 11) @
136 077515	105	130	105	EMS205: .ASCIZ	@EXECUTED @
137 077527	127	111	124	EMS206: .ASCIZ	@WITH COMP ERROR SET @
138 077554	042	107	117	EMS207: .ASCIZ	@'GO' (RMCS1, BIT 0) @
139 077601	127	122	111	EMS210: .ASCIZ	@WRITING @
140 077612	127	101	123	EMS211: .ASCIZ	@WAS RESET BY @
141 077630	120	122	117	EMS212: .ASCIZ	@PROGRAM INTERRUPT @
142 077653	127	101	123	EMS213: .ASCIZ	@WAS NOT GENERATED @
143 077676	123	105	105	EMS214: .ASCIZ	@SEEK COMMAND @
144 077714	120	122	117	EMS215: .ASCIZ	@PROGRAM TIMEOUT @
145 077735	104	125	122	EMS216: .ASCIZ	@DURING LOOK AHEAD TEST @
146 077765	114	117	117	EMS217: .ASCIZ	@LOOK AHEAD REGISTER,RMLA, @
147 100020	123	105	101	EMS220: .ASCIZ	@SEARCH COMMAND @
148 100040	102	101	104	EMS221: .ASCIZ	@BAD SECTOR ERROR 'BSE' (RMER2, BIT 15) @
149 100110	101	040	104	EMS222: .ASCIZ	@A DATA TRANSFER COMMAND @
150 100141	110	105	101	EMS223: .ASCIZ	@HEADER COMPARE INHIBIT 'HCI' (RMOF, BIT 10) @
151 100216	116	117	116	EMS224: .ASCIZ	@NONEXISTENT MEMORY 'NEM' (RMCS2, BIT 11) @
152 100270	123	113	111	EMS225: .ASCIZ	@SKIP SECTOR ERROR @
153 100313	042	123	123	EMS226: .ASCIZ	@'SSE' (RMER2, BIT 05) @
154 100342	111	116	110	EMS227: .ASCIZ	@INHIBIT @
155 100353	042	123	123	EMS230: .ASCIZ	@'SSEI' (RMOF, BIT 09) @
156 100402	105	116	104	EMS231: .ASCIZ	@END OF TRACK @
157 100420	101	106	124	EMS232: .ASCIZ	@AFTER DATA TRANSFER @

1	100445	105	130	120	EH1:	.ASCIZ	@EXPCTD	RECEVD@				
2	100464	102	125	123	EH110:	.ASCIZ	@BUSADRA@					
3	100473	040	122	115	EH111:	.ASCIZ	@ RMCS2	RMCS1@				
4												
5	100512	122	105	103	EH114:	.ASCIZ	@RECEVD	SNGPRT	DULPRT@			
6	100541	105	130	120	EH223:	.ASCIZ	@EXPCTD	RECEVD	DATA@			
7	100567	105	130	120	EH256:	.ASCII	@EXPCTD	RECEVD	RGSTRA<CRLF>			
8	100615	123	124	101		.ASCIZ	@STATUS	STATUS	INDEX@			
9												
10	100643	107	104	101	EH336:	.ASCIZ	@GDADRS	GDDATA	BDADRS	BDDATA@		
11	100702	122	115	103	EH337:	.ASCII	@RMCS2	STATUS	FAILING	DATA@<CRLF>		
12	100741	137	137	137		.ASCII	@			@<CRLF>		
13	101000	105	130	120		.ASCIZ	@EXPCTD	RECEVD	--BIT--	ADRESS@		
14												
15	101037	122	115	105	EH344:	.ASCII	@RMER1	STATUS	HEADER	FAILING@<CRLF>		
16	101077	137	137	137		.ASCII	@		WORD	BIT@<CRLF>		
17	101135	105	130	120		.ASCIZ	@EXPCTD	RECEVD	NUMBER	POSITON@		
18	101175	105	130	120	EH353:	.ASCII	@EXPCTD	RECEVD@<CRLF>				
19	101214	074	103	122		.ASCIZ	@<CRLF>	RMLA	RMLA	RMOF @		
20												
21	101251	040	122	115	STSH1:	.ASCII	@ RMCS1	RMCS2	RMDS	RMER1	RMER2@	
22	101316	040	040	040		.ASCIZ	@	RMAS@				
23	101326	040	122	115	STSH2:	.ASCII	@ RMWC	RMBA	RMDA	RMOF	RMDC@	
24	101373	040	040	040		.ASCIZ	@	RMEC1	RMEC2@			
25	101415	040	122	115	STSH3:	.ASCIZ	@ RMDA	RMDC	RMOF	RMLA@		
26	101453	040	122	115	STSH4:	.ASCIZ	@ RMR1	RMR2	RMDT	RMSN@		
27						.EVEN						

1	101512	001140	001142	000000	ED1:	.WORD	SGDDAT,SBDDAT,0
2	101520	001276	000000		ED110:	.WORD	SBASE,0
3	101524	001174	001176	000000	ED111:	.WORD	STMP0,STMP1,0
4							
5	101532	001364	001176	001200	ED114:	.WORD	RMDTI,STMP1,STMP2,0
6	101542	001140	001142	001174	ED223:	.WORD	SGDDAT,SBDDAT,STMP0,0
7							
8	101552	001134	001140	001136	ED336:	.WORD	SGDADR,SGDDAT,SBADR,SBDDAT,0
9							
10	101564	001140	001142	001174	ED337:	.WORD	SGDDAT,SBDDAT,STMP0,STMP1,0
11	101576	001140	001142	001444	ED353:	.WORD	SGDDAT,SBDDAT,RMOFO,0
12							
13	101606	001336	001346	001350	STSD1:	.WORD	RMCS1I,RMCS2I,RMDSI,RMER1I,RMER2I,RMASI,0
14	101624	001340	001342	001344	STSD2:	.WORD	RMWCI,RMBAI,RMDAI,RMOFI,RMDCI,RMEC1I
15	101640	001404	000000			.WORD	RMEC2I,0
16	101644	001344	001372	001370	STSD3:	.WORD	RMDAI,RMDCI,RMOFI,RMLAI,0
17	101656	001362	001376	001364	STSD4:	.WORD	RMMR1I,RMMR2I,RMDTI,RMSNI,0

1	101670	000			EF110:	.BYTE	0
2	101671	000	000		EF111:	.BYTE	0.0
3	101673	000	000	000	EF114:	.BYTE	0.0.0
4	101676	000	000	000	EF336:	.BYTE	0.0.0.0
5	101702	000	000	000	EF337:	.BYTE	0.0.0.0.0
6							
7	101707	000	000	000	STSF:	.BYTE	0.0.0.0.0.0.0
8					.EVEN		

```

1      ;STORAGE FOR GENERAL DATA TRANSFERRS
2 101716 BUFFER:
3 101716 BUFONE: .BLKW 258.
4 102722 BUFTWO: .BLKW 258.
5
6      ;STORAGE FOR MANUFACTURES BAD SECTOR FILE(DEC144)
7 103726 000000 000000 MFGFIL: .WORD 0,0          ;2 HEADER WORDS
8 103732 .BLKW 256.          ;256. WORDS OF DATA
9 104732 177777 .WORD -1          ;TERMINATOR IF FILE IS FULL
10
11     ;STORAGE FOR USERS BAD SECTOR FILE(DEC144)
12 104734 000000 000000 USRFIL: .WORD 0,0          ;2 HEADER WORDS
13 104740 .BLKW 256.          ;256. WORDS OF DATA
14 105740 177777 .WORD -1          ;TERMINATOR IF FILE IS FULL
15
16     ;STORAGE FOR SKIP SECTOR FILE
17 105742 000000 000000 SSFIL: .WORD 0,0          ;2 HEADER WORDS
18 105746 .BLKW 28.           ;2 CYLINDERS OF SKIP SECTOR ENTRIES
19 106036 177777 .WORD -1          ;TERMINATOR
20
21     .-BUFFER
22
23     HELP:
24 101716 .ASCII <CRLF>
25 101717 .ASCII <CRLF>
26 101720 114 111 123 .ASCII @LIST OF TESTS@<CRLF>
27 101736 055 055 055 .ASCII @-----@<CRLF>
28 101754 124 061 011 .ASCII @T1 CONTROLLER ACCESS TEST@<CRLF>
29 102006 124 062 011 .ASCII @T2 FORMAT ZEROS@<CRLF>
30 102026 124 063 011 .ASCII @T3 ZERO FILL TEST@<CRLF>
31 102050 124 064 011 .ASCII @T4 FORMAT CHECK ZEROS@<CRLF>
32 102076 124 065 011 .ASCII @T5 FORMAT CHECK ZEROS W/ WCE ERROR@<CRLF>
33 102141 124 066 011 .ASCII @T6 FORMAT ONES@<CRLF>
34 102160 124 067 011 .ASCII @T7 FORMAT CHECK ONES@<CRLF>
35 102205 124 061 060 .ASCII @T10 FORMAT CHECK ONES W/ WCE ERRORS@<CRLF>
36 102251 124 061 061 .ASCII @T11 FORMAT MULTIPLE SECTORS@<CRLF>
37 102305 124 061 062 .ASCII @T12 READ HEADER & DATA W/ HEAD SWITCHING@<CRLF>
38 102356 124 061 063 .ASCII @T13 READ HEADER & DATA W/ MID TRANSFER SEEK@<CRLF>
39 102432 124 061 064 .ASCII @T14 FORMAT W/ IMPLIED SEEK@<CRLF>
40 102465 124 061 065 .ASCII @T15 FORMAT EACH SECTOR ADDRESS@<CRLF>
41 102524 124 061 066 .ASCII @T16 FORMAT EACH TRACK ADDRESS@<CRLF>
42 102562 124 061 067 .ASCII @T17 READ HEADER & DATA IN LAST SECTOR@<CRLF>
43 102630 124 062 060 .ASCII @T20 READ HEADER & DATA W/ AOE ERROR@<CRLF>
44 102674 124 062 061 .ASCII @T21 READ INVALID SECTOR ADDRESS@<CRLF>
45 102734 124 062 062 .ASCII @T22 READ INVALID TRACK ADDRESS@<CRLF>
46 102773 124 062 063 .ASCII @T23 READ INVALID CYLINDER ADDRESS@<CRLF>
47 103035 124 062 064 .ASCII @T24 FORMAT AT OFFSET@<CRLF>
48 103062 124 062 065 .ASCII @T25 IVC FORMAT TEST@<CRLF>
49 103106 124 062 066 .ASCII @T26 FORMAT ERROR TEST@<CRLF>
50 103134 124 062 067 .ASCII @T27 FORMAT HCE, FIRST HEADER WORD@<CRLF>
51 103176 124 063 060 .ASCII @T30 FORMAT HCE, SECOND HEADER WORD@<CRLF>
52 103241 124 063 061 .ASCII @T31 FORMAT FE CYLINDERS@<CRLF>
53 103271 200 .ASCII <CRLF>
54 103272 117 120 105 .ASCII @OPERATIONAL SWITCH SETTINGS@<CRLF>
55 103326 055 055 055 .ASCII @-----@<CRLF>
56 103362 123 127 111 .ASCII @SWITCH USE@<CRLF>
57 103377 055 055 055 .ASCII @-----@<CRLF>
    
```

58	103434	040	040	061	.ASCII	a	15
59	103460	040	040	061	.ASCII	a	14
60	103503	040	040	061	.ASCII	a	13
61	103540	040	040	061	.ASCII	a	12
62	103547	040	040	061	.ASCII	a	11
63	103600	040	040	061	.ASCII	a	10
64	103624	040	040	040	.ASCII	a	9
65	103650	040	040	040	.ASCII	a	8
66	103707	040	040	040	.ASCII	a	7
67	103723	040	040	040	.ASCII	a	6
68	103736	040	040	040	.ASCII	a	5
69	103751	040	040	040	.ASCII	a	4
70	103764	040	040	040	.ASCII	a	3
71	103776	040	040	040	.ASCII	a	2
72	104010	040	040	040	.ASCII	a	1
73	104022	040	040	040	.ASCIZ	a	0
74							
75	000200			.END		200	

HALT ON ERRORa<CRLF>
LOOP ON TESTa<CRLF>
INHIBIT ERROR TYPEOUTSa<CRLF>
a<CRLF>
INHIBIT ITERATIONSa<CRLF>
BELL ON ERRORa<CRLF>
LOOP ON ERRORa<CRLF>
LOOP ON TEST IN SWR<7:0>a<CRLF>
TN128a<CRLF>
TN64a<CRLF>
TN32a<CRLF>
TN16a<CRLF>
TN8a<CRLF>
TN4a<CRLF>
TN2a<CRLF>
TN1a<CRLF>

SYMBOL TABLE

ABASE = 176700	ATNMSK= 000377	BUFONE 101716	EBL = 020000	EH344 101037
ACDW1 = 000000	ATNTBL 065004	BUFTWO 102722	ECH = 000100	EH353 101175
ACDW2 = 000000	AUNIT = 000000	CC = 004000	ECI = 004000	EMS1 072102
ACKSTS 046452	AUSWR = 000000	CH = 002000	ECRC = 001000	EMS10 072436
ACPUOP= 000000	AVECT1= 120254	CHGADR 001330	EDT1 071724	EMS100 074656
ADDW0 = 000000	AVECT2= 000000	CHRCNT 061127	EDT110 071742	EMS101 074704
ADDW1 = 000000	A16 = 000400	CKSWR = 104410	EDT111 071744	EMS102 074732
ADDW10= 000000	A17 = 001000	CLKADR 001522	EDT114 071746	EMS103 074761
ADDW11= 000000	BACK = 000001	CLKVCT 001524	EDT2 071734	EMS104 074770
ADDW12= 000000	BADSCT 032406	CLR = 000040	EDT223 071750	EMS105 075017
ADDW13= 000000	BADTMO 005344	CLRSTS 045572	EDT336 071760	EMS106 075046
ADDW14= 000000	BAI = 000010	CMNSTA 007332	EDT337 071770	EMS11 072500
ADDW15= 000000	BB00 = 000001	CMPBUF 035106	EDT344 072000	EMS110 075065
ADDW2 = 000000	BB01 = 000002	CMPERR 043550	EDT353 072012	EMS111 075114
ADDW3 = 000000	BB02 = 000004	CNSLO1 064161	ED1 101512	EMS112 075125
ADDW4 = 000000	BB03 = 000010	CNSLO2 064171	ED110 101520	EMS113 075211
ADDW5 = 000000	BB04 = 000020	CNSLO3 064233	ED111 101524	EMS114 075241
ADDW6 = 000000	BB05 = 000040	CNSLO4 064242	ED114 101532	EMS115 075265
ADDW7 = 000000	BB06 = 000100	CNSLO7 064276	ED223 101542	EMS116 075270
ADDW8 = 000000	BB07 = 000200	CNSLO8 064440	ED336 101552	EMS117 075320
ADDW9 = 000000	BB08 = 000400	CNSLO9 064441	ED337 101564	EMS12 072511
ADEVCT= 000000	BB09 = 001000	CNTCLR 045454	ED353 101576	EMS120 075342
ADEVM = 000000	BIT0 = 000001	COMMA 063545	EECC = 000020	EMS121 075342
ADR = 000001	BIT00 = 000001	CONT = 000100	EFT1 072014	EMS122 075405
AENV = 000000	BIT01 = 000002	CPSAVE 060300	EFT110 072032	EMS123 075442
AENVM = 000000	BIT02 = 000004	CR = 000015	EFT111 072034	EMS124 075505
AFATAL= 000000	BIT03 = 000010	CRLF = 000200	EFT114 072036	EMS125 075537
ALL 063534	BIT04 = 000020	CTLFG 001326	EFT2 072024	EMS126 075602
AMADR1= 000000	BIT05 = 000040	CYLSK= 001777	EFT223 072040	EMS127 075645
AMADR2= 000000	BIT06 = 000100	DBCK = 100000	EFT336 072050	EMS13 072552
AMADR3= 000000	BIT07 = 000200	DBEN = 040000	EFT337 072060	EMS130 075704
AMADR4= 000000	BIT08 = 000400	DBL = 002000	EFT344 072070	EMS131 075742
AMAMS1= 000000	BIT09 = 001000	DCK = 100000	EFT353 072100	EMS132 076002
AMAMS2= 000000	BIT1 = 000002	DDISP = 177570	EF110 101670	EMS133 076026
AMAMS3= 000000	BIT10 = 002000	DEBL = 020000	EF111 101671	EMS134 076051
AMAMS4= 000000	BIT11 = 004000	DEVSEL 043762	EF114 101673	EMS135 076113
AMSGAD= 000000	BIT12 = 010000	DISPLA 001156	EF336 101676	EMS136 076166
AMSGLG= 000000	BIT13 = 020000	DISPRE 000174	EF337 101702	EMS137 076213
AMSGTY= 000000	BIT14 = 040000	DLT = 100000	EHT1 071600	EMS14 072563
AMTYP1= 000000	BIT15 = 100000	DMD = 000001	EHT110 071622	EMS140 076250
AMTYP2= 000000	BIT2 = 000004	DPE = 000010	EHT111 071626	EMS141 076271
AMTYP3= 000000	BIT3 = 000010	DPEHI = 040000	EHT114 071632	EMS142 076317
AMTYP4= 000000	BIT4 = 000020	DPELO = 020000	EHT2 071612	EMS143 076334
AOE = 001000	BIT5 = 000040	DPR = 000400	EHT223 071636	EMS144 076362
APASS = 000000	BIT6 = 000100	DRIVES 064636	EHT256 071650	EMS145 076372
APE = 100000	BIT7 = 000200	DRQ = 004000	EHT336 071662	EMS146 076421
APRIOR= 000000	BIT8 = 000400	DRVCLR= 000010	EHT337 071674	EMS147 076440
APTCSU= 000040	BIT9 = 001000	DRVSTS 051010	EHT344 071706	EMS15 072574
APTENV= 000001	BLNKS1 064701	DRY = 000200	EHT353 071720	EMS150 076455
APTSIZ= 000200	BLNKS2 064700	DSWR = 177570	EH1 100445	EMS151 076474
APTSPO= 000100	BLNKS3 064677	DTASTS 051612	EH110 100464	EMS152 076520
AKGS = 000004	BLNKS4 064676	DTE = 010000	EH111 100473	EMS153 076544
ASNDA 001520	BOTADR 061124	DTO = 010000	EH114 100512	EMS154 076566
ASNDC 001516	BOTFLG 061126	DULPRT= 024026	EH223 100541	EMS155 076602
ASWREG= 000000	BPTVEC= 000014	DVA = 004000	EH256 100567	EMS156 076622
ATA = 100000	BSE = 100000	DVC = 000200	EH336 100643	EMS157 076640
ATESTN= 000000	BUFFER 101716	EARLY 065330	EH337 100702	EMS16 072616

SYMBOL TABLE

EMS160	076655	EMS33	073202	EMT114	066714	EMT177	067602	EMT260	070352
EMS161	076672	EMS34	073230	EMT115	066716	EMT2	065446	EMT261	070364
EMS162	076722	EMS35	073301	EMT116	066726	EMT20	065576	EMT262	070404
EMS163	076722	EMS36	073330	EMT117	066736	EMT200	067604	EMT263	070414
EMS164	076746	EMS37	073357	EMT12	065522	EMT201	067616	EMT264	070424
EMS165	077027	EMS4	072231	EMT120	066746	EMT202	067630	EMT265	070444
EMS166	077043	EMS40	073405	EMT121	066756	EMT203	067642	EMT266	070464
EMS167	077070	EMS41	073434	EMT122	066766	EMT204	067654	EMT267	070476
EMS17	072646	EMS42	073462	EMT123	066776	EMT205	067672	EMT27	065666
EMS170	077115	EMS43	073511	EMT124	067006	EMT206	067702	EMT270	070514
EMS171	077144	EMS44	073540	EMT125	067016	EMT207	067712	EMT271	070530
EMS172	077177	EMS45	073613	EMT126	067026	EMT21	065606	EMT272	070546
EMS173	077217	EMS46	073656	EMT127	067040	EMT210	067724	EMT273	070564
EMS174	077234	EMS47	073705	EMT13	065530	EMT211	067732	EMT274	070602
EMS175	077242	EMS5	072274	EMT130	067052	EMT212	067746	EMT275	070620
EMS176	077270	EMS50	073734	EMT131	067064	EMT213	067762	EMT276	070632
EMS177	077317	EMS51	073763	EMT132	067076	EMT214	067772	EMT277	070646
EMS2	072151	EMS52	074011	EMT133	067110	EMT215	070012	EMT3	065454
EMS20	072706	EMS53	074023	EMT134	067122	EMT216	070024	EMT30	065676
EMS200	077341	EMS54	074035	EMT135	067134	EMT217	070034	EMT300	070664
EMS201	077356	EMS55	074057	EMT136	067146	EMT22	065616	EMT301	070704
EMS202	077424	EMS56	074106	EMT137	067160	EMT220	070044	EMT302	070726
EMS203	077454	EMS57	074163	EMT14	065542	EMT221	070054	EMT303	070740
EMS204	077467	EMS6	072324	EMT140	067170	EMT222	070064	EMT304	070752
EMS205	077515	EMS60	074211	EMT141	067200	EMT223	070074	EMT305	070764
EMS206	077527	EMS61	074224	EMT142	067210	EMT224	070104	EMT306	070776
EMS207	077554	EMS62	074253	EMT143	067220	EMT225	070120	EMT307	071006
EMS21	072731	EMS63	074271	EMT144	067230	EMT226	070140	EMT31	065706
EMS210	077601	EMS64	074317	EMT145	067240	EMT227	070150	EMT310	071026
EMS211	077612	EMS65	074335	EMT146	067250	EMT23	065626	EMT311	071040
EMS212	077630	EMS66	074403	EMT147	067260	EMT230	070164	EMT312	071052
EMS213	077653	EMS67	074453	EMT15	065550	EMT231	070166	EMT313	071064
EMS214	077676	EMS7	072371	EMT150	067270	EMT232	070170	EMT314	071104
EMS215	077714	EMS70	074500	EMT151	067300	EMT233	070172	EMT315	071116
EMS216	077735	EMS71	074512	EMT152	067312	EMT234	070174	EMT316	071130
EMS217	077765	EMS72	074525	EMT153	067324	EMT235	070176	EMT317	071142
EMS22	072754	EMS73	074535	EMT154	067342	EMT236	070200	EMT32	065716
EMS220	100020	EMS74	074552	EMT155	067360	EMT237	070202	EMT320	071152
EMS221	100040	EMS75	074571	EMT156	067372	EMT24	065636	EMT321	071162
EMS222	100110	EMS76	074577	EMT157	067404	EMT240	070204	EMT322	071172
EMS223	100141	EMS77	074630	EMT16	065556	EMT241	070206	EMT323	071200
EMS224	100216	EMTVEC=	000030	EMT160	067416	EMT242	070210	EMT324	071210
EMS225	100270	EMT1	065442	EMT161	067426	EMT243	070212	EMT325	071222
EMS226	100313	EMT10	065512	EMT162	067440	EMT244	070214	EMT326	071234
EMS227	100342	EMT100	066516	EMT163	067452	EMT245	070216	EMT327	071252
EMS23	072770	EMT101	066534	EMT164	067462	EMT246	070220	EMT33	065726
EMS230	100353	EMT102	066552	EMT165	067470	EMT247	070230	EMT330	071264
EMS231	100402	EMT103	066562	EMT166	067476	EMT25	065646	EMT331	071274
EMS232	100420	EMT104	066574	EMT167	067504	EMT250	070242	EMT332	071306
EMS24	073016	EMT105	066612	EMT17	065566	EMT251	070256	EMT333	071320
EMS25	073045	EMT106	066622	EMT170	067512	EMT252	070264	EMT334	071336
EMS26	073062	EMT107	066632	EMT171	067524	EMT253	070272	EMT335	071354
EMS27	073103	EMT11	065516	EMT172	067534	EMT254	070310	EMT336	071374
EMS3	072166	EMT110	066652	EMT173	067544	EMT255	070320	EMT337	071404
EMS30	073114	EMT111	066664	EMT174	067554	EMT256	070330	EMT34	065736
EMS31	073124	EMT112	066672	EMT175	067566	EMT257	070340	EMT340	071414
EMS32	073153	EMT113	066700	EMT176	067574	EMT26	065656	EMT341	071426

EMT342 071434
EMT343 071446
EMT344 071460
EMT345 071472
EMT346 071502
EMT347 071516
EMT35 065746
EMT350 071530
EMT351 071546
EMT352 071556
EMT353 071562
EMT354 071572
EMT36 065756
EMT37 065766
EMT4 065462
EMT40 065776
EMT41 066004
EMT42 066014
EMT43 066026
EMT44 066036
EMT45 066046
EMT46 066056
EMT47 066066
EMT5 065470
EMT50 066074
EMT51 066104
EMT52 066116
EMT53 066134
EMT54 066152
EMT55 066162
EMT56 066174
EMT57 066212
EMT6 065476
EMT60 066222
EMT61 066240
EMT62 066260
EMT63 066274
EMT64 066304
EMT65 066324
EMT66 066344
EMT67 066356
EMT7 065504
EMT70 066370
EMT71 066400
EMT72 066410
EMT73 066426
EMT74 066444
EMT75 066454
EMT76 066474
EMT77 066504
ENRGDT 065442
EQUALS 063532
ERR = 040000
ERRNMB 061122
ERROR = 104000
ERRTYP 060304
ERRVEC= 000004

ERTY00 061130
ERTY01 061135
ERTY02 061145
ERTY03 061154
ERTY04 061162
ERTY05 061165
ESRC = 004000
FER = 000020
FIND = 000001
FMT16 = 010000
FNCDTB 064704
FNCMSK= 000077
FO = 000002
F1 = 000004
F2 = 000010
F3 = 000020
F4 = 000040
GENBUF 034640
GET 035534
GETBUF 001336
GETINX 001526
GETSSF 034514
GETSTS 035450
GO = 000001
GTSWR = 104407
HCE = 000200
HCI = 002000
HCRC = 000400
HELP 101716
HT = 000011
IAE = 002000
IBSAVE 060302
IDXMSK= 000077
IE = 000100
ILF = 000001
ILF02 = 000002
ILF24 = 000024
ILF26 = 000026
ILF30 = 000030
ILF32 = 000032
ILF34 = 000034
ILF36 = 000036
ILF40 = 000040
ILF42 = 000042
ILF44 = 000044
ILF46 = 000046
ILF54 = 000054
ILF56 = 000056
ILF64 = 000064
ILF66 = 000066
ILF74 = 000074
ILF76 = 000076
ILR = 000002
ILRG50= 000050
ILRG52= 000052
ILRG54= 000054
ILRG56= 000056

ILRG60= 000060
ILRG62= 000062
ILRG64= 000064
ILRG66= 000066
ILRG70= 000070
ILRG72= 000072
ILRG74= 000074
ILRG76= 000076
IOTVEC= 000020
IPCK0 = 000001
IPCK1 = 000002
IPCK2 = 000004
IPCK3 = 000010
IR = 000100
IVC = 010000
LBC = 002000
LBT = 002000
LF = 000012
LODEV 064544
LS = 000004
LSC = 004000
LST = 000002
LSTRK 001334
MCLK = 004000
MCPE = 020000
MDF = 000100
MDPE = 000400
MEDENB 001512
MFGFIL 103726
MI = 000004
MIXED 065014
MOC = 000400
MOH = 020000
MOL = 010000
MRD = 002000
MS = 000040
MSC = 000002
MSDRVS 064462
MSE = 100000
MSER = 000200
MSGDRV 064476
MSHELP 063550
MUR = 001000
MWD = 000010
MWP = 000010
MXF = 001000
N 064672
NDTMSK= 115760
NED = 010000
NEM = 004000
NONE 064665
NOP = 000000
NOTAVL 064576
NOTPRS 064561
NOTRM 064527
NSA = 100000
OCC = 100000

OFD = 000200
OFFSET= 000014
OM = 000001
ONES 065054
OPE = 020000
OPI = 020000
OR = 000200
PACACK= 000022
PAKACK= 000022
PAR = 000010
PAT = 000020
PDA = 000400
PFECB 061172
PFECB1 061202
PFECB2 061270
PFECB3 061306
PFECB4 061314
PGE = 002000
PGM = 001000
PHA = 000200
PIP = 020000
PIRQ = 177772
PIRQVE= 000240
PLFS = 002000
PRIERR 036552
PRO = 000000
PR1 = 000040
PR2 = 000100
PR3 = 000140
PR4 = 000200
PR5 = 000240
PR6 = 000300
PR7 = 000340
PS = 177776
PSEL = 002000
PSW = 177776
PUT 036004
PUTBUF 001412
PUTINX 001555
PWRVEC= 000024
QUES 063541
RCLSTS 047246
RD = 000070
RDCHR = 104411
RDLIN = 104412
RDOCT = 104413
RDY = 000200
READY 007524
RECAL = 000006
RESREG= 104415
RESVEC= 000010
REX = 010000
RG = 040000
RGDTPT 065014
RH = 000072
RIP = 000020
RELEASE= 000012

RMAS = 000016
RMASI 001354
RMASO 001430
RMBA = 000004
RMBAE = 000050
RMBAEI 001406
RMBAEO 001462
RMBAI 001342
RMBAO 001416
RMCS1 = 000000
RMCS1I 001336
RMCS10 001412
RMCS2 = 000010
RMCS2I 001346
RMCS20 001422
RMCS3 = 000052
RMCS3I 001410
RMCS30 001464
RMDA = 000006
RMDAI 001344
RMDAO 001420
RMDB = 000022
RMDBI 001360
RMDBO 001434
RMDC = 000034
RMDCI 001372
RMDCO 001446
RMDS = 000012
RMSDI 001350
RMSO 001422
RMDT = 000026
RMDTI 001364
RMDTO 001440
RMEC1 = 000044
RMEC1I 001402
RMEC10 001456
RMEC2 = 000046
RMEC2I 001404
RMEC20 001460
RMER1 = 000014
RMER1I 001352
RMER10 001426
RMER2 = 000042
RMER2I 001400
RMER20 001454
RMHR = 000036
RMHRI 001374
RMHRO 001450
RMLA = 000020
RMLAI 001356
RMLAO 001432
RMMR1 = 000024
RMMR1I 001362
RMMR10 001436
RMMR2 = 000040
RMMR2I 001376
RMMR20 001452

SYMBOL TABLE

RMOF	=	000032	STSD4	101656	TST12	016560	SATY3	063206	SGTSWR	061720
RMOF I		001370	STSF	101707	TST13	017220	SATY4	063216	SGT42P	031366
RMOFO		001444	STSH1	101251	TST14	017674	SAUTOB	001150	SHD	= 000000
RMR	=	000004	STSH2	101326	TST15	020516	SBASE	001276	SHIBTS	001100
RMSN	=	000030	STSH3	101415	TST16	021260	SBDADR	001136	SHIOCT	062712
RMSNI		001366	STSH4	101453	TST17	022022	SBDDAT	001142	SICNT	001120
RMSNO		001442	SWR	001154	TST2	010076	SBELL	001212	SILLUP	063162
RMWC	=	000002	SWREG	000176	TST20	022374	SBIN	056104	SINTAG	001151
RMWCI		001340	SW0	= 000001	TST21	022746	SCDW1	001302	SITEMB	001130
RMWCO		001414	SW00	= 000001	TST22	023310	SCDW2	001304	SLF	001220
RQA	=	100000	SW01	= 000002	TST23	023632	SCHARC	057110	SLFLG	063443
RQB	=	040000	SW02	= 000004	TST24	024150	SCKSWR	061630	SLPADR	001122
RTC	=	000016	SW03	= 000010	TST25	025062	SCMTAG	001114	SLPERR	001124
R6	=	X000006	SW04	= 000020	TST26	025704	SCM3	= 000000	SMADR1	001254
R7	=	X000007	SW05	= 000040	TST27	026606	SCM4	= 000005	SMADR2	001260
SADMSK	=	000377	SW06	= 000100	TST3	011010	SCNTLC	062526	SMADR3	001264
SAVREG	=	104414	SW07	= 000200	TST30	027674	SCNTLG	062540	SMADR4	001270
SA1	=	000001	SW08	= 000400	TST31	030520	SCNTLU	062533	SMAIL	001222
SA16	=	000020	SW09	= 001000	TST4	011644	SCPUOP	001250	SMAMS1	001252
SA2	=	000002	SW1	= 000002	TST5	012434	SCRLF	001217	SMAMS2	001256
SA4	=	000004	SW10	= 002000	TST6	013374	SDBLK	056322	SMAMS3	001262
SAB	=	000010	SW11	= 004000	TST7	014214	SDDW0	001306	SMAMS4	001266
SC	=	100000	SW12	= 010000	TYPBN	= 104406	SDDW1	001310	SMBADR	001102
SCOPE	=	000004	SW13	= 020000	TYPDS	= 104405	SDDW2	001312	SMFLG	063442
SCTMSG		063446	SW14	= 040000	TYPE	= 104401	SDDW3	001314	SMNEW	062556
SCTMSK	=	003700	SW15	= 100000	TYPOC	= 104402	SDDW4	001316	SMSGAD	001236
SCO	=	000100	SW2	= 000004	TYPON	= 104404	SDDW5	001320	SMSGLG	001240
SC1	=	000200	SW3	= 000010	TYPOS	= 104403	SDDW6	001322	SMSGTY	001222
SC2	=	000400	SW4	= 000020	UNS	= 040000	SDDW7	001324	SMSWR	062545
SC3	=	001000	SW5	= 000040	UNTMSK	= 000007	SDEVCT	001232	SMTYP1	001253
SC4	=	002000	SW6	= 000100	UNTOFF	064615	SDEVN	001300	SMTYP2	001257
SEARCH	=	000030	SW7	= 000200	UNTON	064626	SDOAGN	031412	SMTYP3	001263
SECERR		037404	SW8	= 000400	UPE	= 020000	SDTBL	056312	SMTYP4	001267
SEEK	=	000004	SW9	= 001000	USE	= 040000	SENDAD	031402	SMXCNT	057576
SEKSTS		044174	SYSTAT	064504	USRFIL	104734	SENDCT	031240	SNULL	001170
SHUT		031422	TADMSK	= 177400	U0	= 000001	SENULL	031416	SNWTST	= 000001
SHUT2		062570	TAG	= 020000	U1	= 000002	SENV	001242	SOCNT	056554
SIZCLK		036234	TAGADR	= 001114	U2	= 000004	SENVN	001243	SOMODE	056556
SKI	=	040000	TAP	= 040000	VV	= 000100	SEOP	031204	SOVER	057562
SNGPRT	=	020026	TA1	= 000400	WC	= 000040	SEOPCT	031232	SPASS	001230
SSE	=	000040	TA2	= 001000	WCD	= 000050	SEOSP	031146	SPASTM	001106
SSEI	=	001000	TA4	= 002000	WCE	= 040000	SERFLG	001117	SPOWER	063170
SSF	=	020000	TAB	= 004000	WCEHI	= 010000	SERMAX	001131	SPWRDN	063022
SSFENB		001514	TBITVE	= 000014	WCELO	= 004000	SERROR	057710	SPWRMG	063156
SSFIL		105742	TIMOUT	036356	WCF	= 000040	SERRPC	001132	SPWRUP	063074
STACK	=	001100	TKVEC	= 000060	WCH	= 000052	SERRTB	001604	SQUES	001216
STANDA		006654	TPVEC	= 000064	WD	= 000060	SERTTL	001126	SRDCHR	062172
START		005434	TRAPVE	= 000034	WH	= 000062	SESCAP	001210	SRDLIN	062262
START1		005424	TRE	= 040000	WLE	= 004000	SETABL	001242	SRDOCT	062612
START2		005440	TRTVEC	= 000014	WRL	= 004000	SETEND	001326	SRDSZ	= 000010
STCDRV		055232	TST	= 010000	XSIZ	006364	SFATAL	001224	SRESRE	055774
STIMER		036520	TSTNMB	061120	XXDP	001332	SFFLG	063444	SRM80	064522
STKLMT	=	177774	TSTPRP	031456	Y	064674	SFILLC	001172	SRTNAD	031414
STOP		057662	TSTQUE	001466	ZEROS	065116	SFILLS	001171	SSAVRE	055736
STSD1		101606	TST1	007676	\$APTHD	001100	SGDADR	001134	SSAVR6	063166
STSD2		101624	TST10	015004	\$ATYC	063224	SGDDAT	001140	SSCOPE	057114
STSD3		101644	TST11	015742	\$ATY1	063200	\$GET42	031372	\$SETUP	= 000137

SYMBOL TABLE

SSTUP = 177777
 SSVLAD 057526
 SSVPC = 000210
 SSWR = 167400
 SSWREG 001244
 SSWRMK= 000000
 SSWOST 057600
 STESTN 001226
 STIMES 001206
 STKB 001162
 STKCNT 061320

STKINT 061330
 STKQEN= 061327
 STKQIN 061322
 STKQOU 061324
 STKQSR 061326
 STKS 001160
 STKSRV 061400
 STMP0 001174
 STMP1 001176
 STMP2 001200
 STMP3 001202

STMP4 001204
 STN = 000032
 STPB 001166
 STPFLG 001173
 STPS 001164
 STRAP 062714
 STRAP2 062754
 STRP = 000016
 STRPAD 062766
 STSTM 001104
 STSTNM 001116

STTYIN 062516
 STYPBN 056032
 STYPDS 056106
 STYPE 056560
 STYPEC 056772
 STYPEX 057112
 STYPOC 056356
 STYPON 056372
 STYPOS 056332
 SUNIT 001234
 SUNITM 001110

SUSWR 001246
 SVECT1 001272
 SVECT2 001274
 SXOFF = 000023
 SXON = 000021
 SXTSTR 057136
 \$\$GET4= 000000
 \$\$\$W08= 000032
 \$OFILL 056555
 .SX = 001100

. ABS. 106040 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 61952 WORDS (242 PAGES)
 DYNAMIC MEMORY AVAILABLE FOR 70 PAGES
 CZRNEA.BIC,CZRNEA/C=CZRNEA.DOC,CZRNEA,[20,0]SYSMAC/M

AENV	6-0	6-0												
AENVM	6-0	6-0												
AFATAL	6-0	6-0												
ALL	11-69	49-5#												
AMADR1	6-0	6-0												
AMADR2	6-0	6-0												
AMADR3	6-0	6-0												
AMADR4	6-0	6-0												
AMAMS1	6-0	6-0												
AMAMS2	6-0	6-0												
AMAMS3	6-0	6-0												
AMAMS4	6-0	6-0												
AMSGAD	6-0	6-0												
AMSGLG	6-0	6-0												
AMSGTY	6-0	6-0												
AMTYP1	6-0	6-0												
AMTYP2	6-0	6-0												
AMTYP3	6-0	6-0												
AMTYP4	6-0	6-0												
AOE	4-579#	4-590	25-307	25-322	25-337	25-352	25-356	25-679	25-683	50-77	50-78	50-81	50-82	50-85
	50-86													
APASS	6-0	6-0												
APE	4-760#													
APRIOR	6-0													
APTCSU	40-1	48-1#												
APTENV	40-1	42-1	48-1	48-1#										
APTSIZ	10-23	48-1#												
APTSPO	40-1	48-1	48-1#											
ARGS	13-44#	13-52	13-52#	13-64#	13-75#	13-84#	13-88#	13-99#	13-105#	13-109#	13-111#	13-113#	13-120#	13-126#
	13-130#	13-132#	13-134#	13-138#	13-156#	13-164	13-164#	13-176#	13-185#	13-189#	13-201#	13-207#	13-211#	13-213#
	13-215#	13-223#	13-229#	13-233#	13-235#	13-237#	13-241#	13-254#	13-262	13-262#	13-274#	13-283#	13-287#	13-298#
	13-304#	13-308#	13-310#	13-312#	13-318#	13-324#	13-328#	13-330#	13-332#	13-347#	13-355	13-355#	13-367#	13-376#
	13-380#	13-391#	13-397#	13-401#	13-403#	13-405#	13-414#	13-420#	13-424#	13-431#	13-449#	13-472#	13-480	13-480#
	13-492#	13-501#	13-505#	13-516#	13-522#	13-526#	13-528#	13-530#	13-537#	13-543#	13-547#	13-549#	13-551#	13-555#
	13-568#	13-576	13-576#	13-588#	13-597#	13-601#	13-612#	13-618#	13-622#	13-624#	13-626#	13-632#	13-638#	13-642#
	13-644#	13-646#	13-661#	13-669	13-669#	13-681#	13-690#	13-694#	13-705#	13-711#	13-715#	13-717#	13-719#	13-728#
	13-734#	13-738#	13-745#	13-761#	13-786#	13-794	13-794#	13-806#	13-815#	13-819#	13-830#	13-836#	13-840#	13-842#
	13-844#	13-850#	13-856#	13-860#	13-862#	13-864#	13-884	13-884#	13-896#	13-905#	13-909#	13-920#	13-926#	13-935#
	13-937#	13-939#	13-961	13-961#	13-973#	13-982#	13-986#	13-997#	13-:03#	13-:12#	13-:14#	13-:16#	13-:36#	13-:44
	13-:44#	13-:58#	13-:67#	13-:71#	13-:83#	13-:89#	13-:93#	13-:95#	13-:97#	13-:03#	13-:09#	13-:13#	13-:15#	13-:17#
	13-:37#	13-:45	13-:45#	13-:59#	13-:66#	13-:70#	13-:72#	13-:74#	13-:81#	13-:87#	13-:91#	13-:93#	13-:95#	13-:99#
	13-<20#	13-<28	13-<28#	13-<42#	13-<49#	13-<53#	13-<55#	13-<57#	13-<64#	13-<70#	13-<74#	13-<76#	13-<78#	13-<82#
	13-=05	13-=05#	13-=18#	13-=25#	13-=34#	13-=36#	13-=38#	13-=61	13-=61#	13-=72#	13-=79#	13-=88#	13-=90#	13-=92#
	13->13	13->13#	13->26#	13->33#	13->37#	13->39#	13->41#	13->69	13->69#	13->82#	13->89#	13->93#	13->95#	13->97#
	13-?18	13-?18#	13-?31#	13-?38#	13-?42#	13-?44#	13-?46#	13-?66#	13-?74	13-?74#	13-?86#	13-?97#	13-@04#	13-@08#
	13-@10#	13-@12#	13-@23#	13-@35#	13-@41#	13-@45#	13-@47#	13-@49#	13-@53#	13-@72#	13-@80	13-@80#	13-@88#	13-A04#
	13-A11#	13-A15#	13-A24#	13-A28	13-A28#	13-A35#	13-A42#	13-A46#	13-A55#	13-A71#	13-A79	13-A79#	13-A93#	13-B04#
	13-B11#	13-B15#	13-B17#	13-B19#	13-B30#	13-B41#	13-B47#	13-B51#	13-B58#	13-B65#	13-B86#	13-B94	13-B94#	13-C16#
	13-C23#	13-C27#	13-C29#	13-C31#	13-C38#	13-C44#	13-C48#	13-C61#	13-C74#	13-C87#	13-D00#	13-D12#	13-D40#	13-D48
	13-D48#	13-D70#	13-D77#	13-D81#	13-D83#	13-D85#	13-D92#	13-D98#	13-E02#	13-E09#	13-E21#	13-E47#	13-E61	13-E61#
	13-E75#	13-E78#	13-E80#	13-E82#	13-E86#	13-E88#								
ASNDA	7-0#	16-307*	16-344	16-374*	16-377	16-380	16-382*	16-383*	16-384	16-386*	16-425			
ASNDC	7-0#	16-306*	16-343	16-387*	16-388	16-424								
ASWREG	6-0	6-0												
ATA	4-559#	25-134	25-135	25-137	28-161	28-162	28-197	28-200	32-145	32-146	32-185	32-188	50-58	50-59
	50-60	50-63	50-64	50-67	50-68	50-69	50-70	50-71	50-72	50-73	50-74	50-75	50-76	50-79

	50-80	50-83	50-84	50-87	50-88														
ATESTN	6-0	6-0																	
ATNMSK	4-596#	33-55																	
ATNTBL	10-78	10-106	10-110	11-97	12-17	51-3#													
AUNIT	6-0	6-0																	
AUSWR	6-0	6-0																	
AVECT1	4-783#	6-0	6-0																
AVECT2	6-0	6-0																	
BACK	13-44	13-44#	13-52	13-52#	13-64	13-64#	13-75	13-75#	13-84	13-84#	13-88	13-88#	13-88#	13-88#	13-99				
	13-99#	13-105	13-105#	13-109	13-109#	13-109#	13-111	13-111#	13-111#	13-113	13-113#	13-113#	13-113#	13-120	13-120#				
	13-126	13-126#	13-130	13-130#	13-130#	13-132	13-132#	13-132#	13-134	13-134#	13-134#	13-134#	13-138	13-138#	13-156				
	13-156#	13-164	13-164#	13-176	13-176#	13-185	13-185#	13-189	13-189#	13-189#	13-201	13-201#	13-201#	13-207	13-207#				
	13-211	13-211#	13-211#	13-213	13-213#	13-213#	13-215	13-215#	13-215#	13-215#	13-223	13-223#	13-223#	13-229	13-229#				
	13-233#	13-233#	13-235	13-235#	13-235#	13-237	13-237#	13-237#	13-241	13-241#	13-254	13-254#	13-254#	13-262	13-262#				
	13-274	13-274#	13-283	13-283#	13-287	13-287#	13-287#	13-298	13-298#	13-304	13-304#	13-304#	13-308	13-308#	13-308#				
	13-310	13-310#	13-310#	13-312	13-312#	13-312#	13-318	13-318#	13-324	13-324#	13-328	13-328#	13-328#	13-328#	13-330				
	13-330#	13-330#	13-332	13-332#	13-332#	13-347	13-347#	13-355	13-355#	13-367	13-367#	13-367#	13-376	13-376#	13-380				
	13-380#	13-380#	13-391	13-391#	13-397	13-397#	13-401	13-401#	13-401#	13-403	13-403#	13-403#	13-403#	13-405	13-405#				
	13-405#	13-414	13-414#	13-420	13-420#	13-424	13-424#	13-424#	13-431	13-431#	13-431#	13-431#	13-449	13-449#	13-472				
	13-472#	13-480	13-480#	13-492	13-492#	13-501	13-501#	13-505	13-505#	13-505#	13-516	13-516#	13-516#	13-522	13-522#				
	13-526	13-526#	13-526#	13-528	13-528#	13-528#	13-530	13-530#	13-530#	13-537	13-537#	13-537#	13-543	13-543#	13-547				
	13-547#	13-547#	13-549	13-549#	13-549#	13-551	13-551#	13-551#	13-555	13-555#	13-568	13-568#	13-568#	13-576	13-576#				
	13-588	13-588#	13-597	13-597#	13-601	13-601#	13-601#	13-612	13-612#	13-618	13-618#	13-618#	13-622	13-622#	13-622#				
	13-624	13-624#	13-624#	13-626	13-626#	13-626#	13-632	13-632#	13-638	13-638#	13-642	13-642#	13-642#	13-642#	13-644				
	13-644#	13-644#	13-646	13-646#	13-646#	13-661	13-661#	13-669	13-669#	13-681	13-681#	13-681#	13-690	13-690#	13-694				
	13-694#	13-694#	13-705	13-705#	13-711	13-711#	13-715	13-715#	13-715#	13-717	13-717#	13-717#	13-719	13-719#	13-719#				
	13-719#	13-728	13-728#	13-734	13-734#	13-738	13-738#	13-738#	13-745	13-745#	13-745#	13-745#	13-761	13-761#	13-786				
	13-786#	13-794	13-794#	13-806	13-806#	13-815	13-815#	13-819	13-819#	13-819#	13-830	13-830#	13-836	13-836#	13-836#				
	13-840	13-840#	13-840#	13-842	13-842#	13-842#	13-844	13-844#	13-844#	13-850	13-850#	13-856	13-856#	13-860	13-860#				
	13-860#	13-860#	13-862	13-862#	13-862#	13-864	13-864#	13-864#	13-884	13-884#	13-896	13-896#	13-905	13-905#	13-905#				
	13-909	13-909#	13-909#	13-920	13-920#	13-926	13-926#	13-935	13-935#	13-935#	13-937	13-937#	13-937#	13-939	13-939#				
	13-939#	13-939#	13-961	13-961#	13-973	13-973#	13-982	13-982#	13-986	13-986#	13-986#	13-997	13-997#	13-997#	13-:03				
	13-:03#	13-:12	13-:12#	13-:12#	13-:14	13-:14#	13-:14#	13-:16	13-:16#	13-:16#	13-:36	13-:36#	13-:44	13-:44#	13-:44#				
	13-:58	13-:58#	13-:67	13-:67#	13-:71	13-:71#	13-:71#	13-:83	13-:83#	13-:89	13-:89#	13-:93	13-:93#	13-:93#	13-:93#				
	13-:95	13-:95#	13-:95#	13-:97	13-:97#	13-:97#	13-:03	13-:03#	13-:09	13-:09#	13-:13	13-:13#	13-:13#	13-:13#	13-:15				
	13-:15#	13-:15#	13-:17	13-:17#	13-:17#	13-:37	13-:37#	13-:45	13-:45#	13-:59	13-:59#	13-:66	13-:66#	13-:66#	13-:70				
	13-:70#	13-:70#	13-:72	13-:72#	13-:72#	13-:74	13-:74#	13-:74#	13-:81	13-:81#	13-:87	13-:87#	13-:87#	13-:91	13-:91#				
	13-:91#	13-:93	13-:93#	13-:95	13-:95#	13-:95#	13-:99	13-:99#	13-:99#	13-:20	13-:20#	13-:28	13-:28#	13-:28#	13-:42				
	13-<42#	13-<49	13-<49#	13-<53	13-<53#	13-<53#	13-<55	13-<55#	13-<55#	13-<57	13-<57#	13-<57#	13-<57#	13-<64	13-<64#				
	13-<70	13-<70#	13-<74	13-<74#	13-<74#	13-<76	13-<76#	13-<76#	13-<78	13-<78#	13-<78#	13-<82	13-<82#	13-<82#	13-:05				
	13-:05#	13-:18	13-:18#	13-:25	13-:25#	13-:34	13-:34#	13-:34#	13-:36	13-:36#	13-:36#	13-:38	13-:38#	13-:38#	13-:38#				
	13-:61	13-:61#	13-:72	13-:72#	13-:79	13-:79#	13-:88	13-:88#	13-:88#	13-:90	13-:90#	13-:90#	13-:92	13-:92#	13-:92#				
	13-:92#	13->13	13->13#	13->26	13->26#	13->33	13->33#	13->33#	13->37	13->37#	13->37#	13->39	13->39#	13->39#	13->41				
	13->41#	13->41#	13->69	13->69#	13->82	13->82#	13->89	13->89#	13->93	13->93#	13->93#	13->95	13->95#	13->95#	13->95#				
	13->97	13->97#	13->97#	13-?18	13-?18#	13-?31	13-?31#	13-?38	13-?38#	13-?42	13-?42#	13-?42#	13-?44	13-?44#	13-?44#				
	13-?44#	13-?46	13-?46#	13-?46#	13-?66	13-?66#	13-?74	13-?74#	13-?86	13-?86#	13-?97	13-?97#	13-?97#	13-@04	13-@04#				
	13-@08	13-@08#	13-@08#	13-@10	13-@10#	13-@10#	13-@12	13-@12#	13-@12#	13-@23	13-@23#	13-@35	13-@35#	13-@35#	13-@41				
	13-@41#	13-@45	13-@45#	13-@45#	13-@47	13-@47#	13-@47#	13-@49	13-@49#	13-@49#	13-@53	13-@53#	13-@72	13-@72#	13-@72#				
	13-@80	13-@80#	13-@88	13-@88#	13-A04	13-A04#	13-A11	13-A11#	13-A15	13-A15#	13-A15#	13-A24	13-A24#	13-A24#	13-A24#				
	13-A28	13-A28#	13-A35	13-A35#	13-A42	13-A42#	13-A46	13-A46#	13-A46#	13-A55	13-A55#	13-A55#	13-A71	13-A71#	13-A71#				
	13-A79	13-A79#	13-A93	13-A93#	13-B04	13-B04#	13-B11	13-B11#	13-B15	13-B15#	13-B15#	13-B17	13-B17#	13-B17#	13-B17#				
	13-B19	13-B19#	13-B19#	13-B30	13-B30#	13-B41	13-B41#	13-B47	13-B47#	13-B51	13-B51#	13-B51#	13-B58	13-B58#	13-B58#				
	13-B58#	13-B65	13-B65#	13-B65#	13-B86	13-B86#	13-B94	13-B94#	13-C16	13-C16#	13-C23	13-C23#	13-C27	13-C27#	13-C27#				
	13-C27#	13-C29	13-C29#	13-C29#	13-C31	13-C31#	13-C31#	13-C38	13-C38#	13-C44	13-C44#	13-C48	13-C48#	13-C48#	13-C48#				
	13-C61	13-C61#	13-C61#	13-C74	13-C74#	13-C74#	13-C87	13-C87#	13-C87#	13-D00	13-D00#	13-D00#	13-D12	13-D12#	13-D12#				
	13-D12#	13-D40	13-D40#	13-D48	13-D48#	13-D70	13-D70#	13-D77	13-D77#	13-D81	13-D81#	13-D81#	13-D83	13-D83#	13-D83#				

DRVSTS	16-104	33-9#												
DRY	4-567#	24-104	25-34	25-37	25-38	28-200	30-125	30-126	33-30					
DSWR	4-492#	6-0	10-23											
DTASTS	13-111	13-132	13-213	13-235	13-310	13-330	13-403	13-431	13-528	13-549	13-624	13-644	13-717	13-745
	13-842	13-862	13-937	13-:14	13-:95	13-:15	13-:72	13-:93	13-<55	13-<76	13-=36	13-=90	13->39	13->95
	13-?44	13-a10	13-a47	13-817	13-858	13-C29	13-C61	13-C74	13-C87	13-D00	13-D83	13-E09	16-199	34-17#
DTE	4-576#	4-590	34-283	34-301	34-304									
DTO	4-614#													
DULPRT	4-653#													
DVA	4-498#	10-89	20-43	21-37	24-65	24-68	27-37	30-31	33-17	33-18				
DVC	4-688#	28-132	30-115	32-92	32-130	32-133	32-266	34-236	34-240	34-243	34-288	35-89	35-94	35-97
EARLY	52-107#													
EBL	4-630#													
ECH	4-582#	4-590	16-221	18-25	25-479	25-494	25-512	25-518	34-427	50-77	50-78	50-85	50-86	
ECI	4-658#	18-21	25-514	34-424										
ECRC	4-634#													
ED1	55-1	59-1#												
ED110	55-4	59-2#												
ED111	55-5	59-3#												
ED114	55-7	59-5#												
ED223	55-8	59-6#												
ED336	55-10	59-8#												
ED337	55-11	55-12	59-10#											
ED353	55-14	59-11#												
EDT1	8-3	8-6	8-9	8-12	8-15	8-18	8-24	8-27	8-30	8-33	8-36	8-39	8-42	8-45
	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81	8-84	8-87
	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123	8-126	8-130
	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-157	8-160	8-163	8-166	8-169	8-172	8-175
	8-178	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212	8-215	8-218
	8-236	8-239	8-242	8-245	8-248	8-251	8-254	8-257	8-260	8-263	8-266	8-269	8-272	8-275
	8-278	8-281	8-284	8-287	8-290	8-293	8-296	8-299	8-302	8-305	8-308	8-311	8-314	8-317
	8-320	8-323	8-326	8-329	8-332	8-335	8-338	8-341	8-344	8-347	8-350	8-353	8-356	8-359
	8-366	8-369	8-372	8-375	8-378	8-381	8-384	8-387	8-390	8-393	8-396	8-399	8-402	8-405
	8-411	8-414	8-417	8-420	8-423	8-426	8-429	8-432	8-435	8-438	8-441	8-444	8-447	8-450
	8-504	8-507	8-510	8-513	8-516	8-519	8-522	8-525	8-528	8-531	8-534	8-537	8-540	8-543
	8-556	8-559	8-562	8-565	8-568	8-571	8-574	8-577	8-580	8-583	8-586	8-589	8-592	8-595
	8-606	8-609	8-612	8-615	8-618	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642	8-645
	8-648	8-651	8-654	8-657	8-660	8-663	8-666	8-669	8-672	8-675	8-678	8-681	8-684	8-687
	8-705	8-708	8-711	8-714	8-717	8-720	55-1#							
EDT110	8-221	55-4#												
EDT111	8-224	8-227	55-5#											
EDT114	55-7#													
EDT2	8-426	8-429	8-513	8-516	55-2#									
EDT223	8-447	8-528	55-8#											
EDT336	8-351	8-681	8-687	8-690	55-10#									
EDT337	8-684	55-11#												
EDT344	8-699	55-12#												
EDT353	8-720	55-14#												
EECC	4-639#													
EF110	56-4	60-1#												
EF111	56-1	56-5	60-2#											
EF114	56-7	56-8	56-14	60-3#										
EF336	56-10	56-11	56-12	60-4#										
EF337	60-5#													
EFT1	8-3	8-6	8-9	8-12	8-15	8-18	8-24	8-27	8-30	8-33	8-36	8-39	8-42	8-45
	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81	8-84	8-87

EMT152	8-323	53-108#
EMT153	8-326	53-109#
EMT154	8-329	53-110#
EMT155	8-332	53-111#
EMT156	8-335	53-112#
EMT157	8-338	53-113#
EMT16	8-42	53-16#
EMT160	8-341	53-114#
EMT161	8-344	53-115#
EMT162	8-347	53-116#
EMT163	53-117#	
EMT164	8-354	53-118#
EMT165	8-357	53-119#
EMT166	8-360	53-120#
EMT167	8-363	53-121#
EMT17	8-45	53-17#
EMT170	8-366	53-122#
EMT171	8-369	53-123#
EMT172	8-372	53-124#
EMT173	8-375	53-125#
EMT174	8-378	53-126#
EMT175	8-381	53-127#
EMT176	8-384	53-128#
EMT177	53-129#	
EMT2	8-6	53-4#
EMT20	8-48	53-18#
EMT200	8-390	53-130#
EMT201	8-393	53-131#
EMT202	8-396	53-132#
EMT203	8-399	53-133#
EMT204	8-402	53-134#
EMT205	8-405	53-135#
EMT206	8-408	53-136#
EMT207	8-411	53-137#
EMT21	8-51	53-19#
EMT210	8-414	53-138#
EMT211	8-417	53-139#
EMT212	8-420	53-140#
EMT213	8-423	53-141#
EMT214	8-426	53-142#
EMT215	8-429	53-143#
EMT216	8-432	53-144#
EMT217	8-435	53-145#
EMT22	8-54	53-20#
EMT220	8-438	53-146#
EMT221	8-441	53-147#
EMT222	8-444	53-148#
EMT223	8-447	53-149#
EMT224	8-450	53-150#
EMT225	8-453	53-151#
EMT226	8-456	53-152#
EMT227	8-459	53-153#
EMT23	8-57	53-21#
EMT230	8-462	53-154#
EMT231	8-465	53-155#
EMT232	8-468	53-156#

EMT233	8-471	53-157#
EMT234	53-158#	
EMT235	53-159#	
EMT236	53-160#	
EMT237	53-161#	
EMT24	8-60	53-22#
EMT240	53-162#	
EMT241	53-163#	
EMT242	53-164#	
EMT243	53-165#	
EMT244	53-166#	
EMT245	53-167#	
EMT246	8-504	53-168#
EMT247	8-507	53-169#
EMT25	8-63	53-23#
EMT250	8-510	53-170#
EMT251	8-513	53-171#
EMT252	8-516	53-172#
EMT253	8-519	53-173#
EMT254	8-522	53-174#
EMT255	8-525	53-175#
EMT256	8-528	53-176#
EMT257	8-531	53-177#
EMT26	8-66	53-24#
EMT260	8-534	53-178#
EMT261	8-537	53-179#
EMT262	8-540	53-180#
EMT263	8-543	53-181#
EMT264	8-546	53-182#
EMT265	8-549	53-183#
EMT266	8-552	53-184#
EMT267	8-556	53-185#
EMT27	8-69	53-25#
EMT270	8-559	53-186#
EMT271	8-563	53-187#
EMT272	8-566	53-188#
EMT273	8-569	53-189#
EMT274	8-573	53-190#
EMT275	8-577	53-191#
EMT276	8-581	53-192#
EMT277	8-586	53-193#
EMT3	8-9	53-5#
EMT30	8-72	53-26#
EMT300	8-590	53-194#
EMT301	8-594	53-195#
EMT302	8-597	53-196#
EMT303	8-600	53-197#
EMT304	8-603	53-198#
EMT305	8-606	53-199#
EMT306	8-609	53-200#
EMT307	8-612	53-201#
EMT31	8-75	53-27#
EMT310	8-615	53-202#
EMT311	8-618	53-203#
EMT312	8-621	53-204#
EMT313	8-624	53-205#

EMT314	8-627	53-206#
EMT315	8-630	53-207#
EMT316	8-633	53-208#
EMT317	8-636	53-209#
EMT32	8-78	53-28#
EMT320	8-639	53-210#
EMT321	8-642	53-211#
EMT322	8-645	53-212#
EMT323	8-648	53-213#
EMT324	8-651	53-214#
EMT325	8-654	53-215#
EMT326	8-657	53-216#
EMT327	8-660	53-217#
EMT33	8-81	53-29#
EMT330	8-663	53-218#
EMT331	8-666	53-219#
EMT332	8-669	53-220#
EMT333	8-672	53-221#
EMT334	8-675	53-222#
EMT335	8-678	53-223#
EMT336	8-351	8-681
EMT337	8-684	53-225#
EMT34	8-84	53-30#
EMT340	8-687	53-226#
EMT341	8-690	53-227#
EMT342	8-693	53-228#
EMT343	8-696	53-229#
EMT344	8-699	53-230#
EMT345	8-702	53-231#
EMT346	8-705	53-232#
EMT347	8-708	53-233#
EMT35	8-87	53-31#
EMT350	8-711	53-234#
EMT351	8-714	53-235#
EMT352	8-717	53-236#
EMT353	8-720	53-237#
EMT354	8-723	53-238#
EMT36	8-90	53-32#
EMT37	8-93	53-33#
EMT4	8-12	53-6#
EMT40	8-96	53-34#
EMT41	8-99	53-35#
EMT42	8-102	53-36#
EMT43	8-105	53-37#
EMT44	8-108	53-38#
EMT45	8-111	53-39#
EMT46	8-114	53-40#
EMT47	8-117	53-41#
EMT5	8-15	53-7#
EMT50	8-120	53-42#
EMT51	8-123	53-43#
EMT52	8-126	53-44#
EMT53	8-130	53-45#
EMT54	8-133	53-46#
EMT55	8-136	53-47#
EMT56	8-139	53-48#

53-224#

13-403	13-403#	13-403#	13-405	13-405#	13-405#	13-414	13-414#	13-414#	13-420	13-420#	13-420#	13-424	13-424#	
13-424#	13-431	13-431#	13-431#	13-449	13-449#	13-449#	13-472	13-472#	13-472#	13-480	13-480#	13-480#	13-492	
13-492#	13-492#	13-501	13-501#	13-501#	13-505	13-505#	13-505#	13-516	13-516#	13-516#	13-522	13-522#	13-522#	
13-526	13-526#	13-526#	13-528	13-528#	13-528#	13-530	13-530#	13-530#	13-537	13-537#	13-537#	13-543	13-543#	
13-543#	13-547	13-547#	13-547#	13-549	13-549#	13-549#	13-551	13-551#	13-551#	13-555	13-555#	13-555#	13-568	
13-568#	13-568#	13-576	13-576#	13-576#	13-588	13-588#	13-588#	13-597	13-597#	13-597#	13-601	13-601#	13-601#	
13-612	13-612#	13-612#	13-618	13-618#	13-618#	13-622	13-622#	13-622#	13-624	13-624#	13-624#	13-626	13-626#	
13-626#	13-632	13-632#	13-632#	13-638	13-638#	13-638#	13-642	13-642#	13-642#	13-644	13-644#	13-644#	13-646	
13-646#	13-646#	13-661	13-661#	13-661#	13-669	13-669#	13-669#	13-681	13-681#	13-681#	13-690	13-690#	13-690#	
13-694	13-694#	13-694#	13-705	13-705#	13-705#	13-711	13-711#	13-711#	13-715	13-715#	13-715#	13-717	13-717#	
13-717#	13-719	13-719#	13-719#	13-728	13-728#	13-728#	13-734	13-734#	13-734#	13-738	13-738#	13-738#	13-745	
13-745#	13-745#	13-761	13-761#	13-761#	13-786	13-786#	13-786#	13-794	13-794#	13-794#	13-806	13-806#	13-806#	
13-815	13-815#	13-815#	13-819	13-819#	13-819#	13-830	13-830#	13-830#	13-836	13-836#	13-836#	13-840	13-840#	
13-840#	13-842	13-842#	13-842#	13-844	13-844#	13-844#	13-850	13-850#	13-850#	13-856	13-856#	13-856#	13-860	
13-860#	13-860#	13-862	13-862#	13-862#	13-864	13-864#	13-864#	13-884	13-884#	13-884#	13-896	13-896#	13-896#	
13-905	13-905#	13-905#	13-909	13-909#	13-909#	13-920	13-920#	13-920#	13-926	13-926#	13-926#	13-935	13-935#	
13-935#	13-937	13-937#	13-937#	13-939	13-939#	13-939#	13-961	13-961#	13-961#	13-973	13-973#	13-973#	13-982	
13-982#	13-982#	13-986	13-986#	13-986#	13-997	13-997#	13-997#	13-:03	13-:03#	13-:03#	13-:12	13-:12#	13-:12#	
13-:14	13-:14#	13-:14#	13-:16	13-:16#	13-:16#	13-:36	13-:36#	13-:36#	13-:44	13-:44#	13-:44#	13-:58	13-:58#	
13-:58#	13-:67	13-:67#	13-:67#	13-:71	13-:71#	13-:71#	13-:83	13-:83#	13-:83#	13-:89	13-:89#	13-:89#	13-:93	
13-:93#	13-:93#	13-:95	13-:95#	13-:95#	13-:97	13-:97#	13-:97#	13-:03	13-:03#	13-:03#	13-:09	13-:09#	13-:09#	
13-:13	13-:13#	13-:13#	13-:15	13-:15#	13-:15#	13-:17	13-:17#	13-:17#	13-:37	13-:37#	13-:37#	13-:45	13-:45#	
13-:45#	13-:59	13-:59#	13-:59#	13-:66	13-:66#	13-:66#	13-:70	13-:70#	13-:70#	13-:72	13-:72#	13-:72#	13-:74	
13-:74#	13-:74#	13-:81	13-:81#	13-:81#	13-:87	13-:87#	13-:87#	13-:91	13-:91#	13-:91#	13-:93	13-:93#	13-:93#	
13-:95	13-:95#	13-:95#	13-:99	13-:99#	13-:99#	13-<20	13-<20#	13-<20#	13-<28	13-<28#	13-<28#	13-<42	13-<42#	
13-<42#	13-<49	13-<49#	13-<49#	13-<53	13-<53#	13-<53#	13-<55	13-<55#	13-<55#	13-<57	13-<57#	13-<57#	13-<64	
13-<64#	13-<64#	13-<70	13-<70#	13-<70#	13-<74	13-<74#	13-<74#	13-<76	13-<76#	13-<76#	13-<78	13-<78#	13-<78#	
13-<82	13-<82#	13-<82#	13-=05	13-=05#	13-=05#	13-=18	13-=18#	13-=18#	13-=25	13-=25#	13-=25#	13-=34	13-=34#	
13-=34#	13-=36	13-=36#	13-=36#	13-=38	13-=38#	13-=38#	13-=61	13-=61#	13-=61#	13-=72	13-=72#	13-=72#	13-=79	
13-=79#	13-=79#	13-=88	13-=88#	13-=88#	13-=90	13-=90#	13-=90#	13-=92	13-=92#	13-=92#	13->13	13->13#	13->13#	
13->26	13->26#	13->26#	13->33	13->33#	13->33#	13->37	13->37#	13->37#	13->39	13->39#	13->39#	13->41	13->41#	
13->41#	13->69	13->69#	13->69#	13->82	13->82#	13->82#	13->89	13->89#	13->89#	13->93	13->93#	13->93#	13->95	
13->95#	13->95#	13->97	13->97#	13->97#	13-?18	13-?18#	13-?18#	13-?31	13-?31#	13-?31#	13-?38	13-?38#	13-?38#	
13-?42	13-?42#	13-?42#	13-?44	13-?44#	13-?44#	13-?46	13-?46#	13-?46#	13-?66	13-?66#	13-?66#	13-?74	13-?74#	
13-?74#	13-?86	13-?86#	13-?86#	13-?97	13-?97#	13-?97#	13-@04	13-@04#	13-@04#	13-@08	13-@08#	13-@08#	13-@10	
13-@10#	13-@10#	13-@12	13-@12#	13-@12#	13-@23	13-@23#	13-@23#	13-@35	13-@35#	13-@35#	13-@41	13-@41#	13-@41#	
13-@45	13-@45#	13-@45#	13-@47	13-@47#	13-@47#	13-@49	13-@49#	13-@49#	13-@53	13-@53#	13-@53#	13-@72	13-@72#	
13-@72#	13-@80	13-@80#	13-@80#	13-@88	13-@88#	13-@88#	13-A04	13-A04#	13-A04#	13-A11	13-A11#	13-A11#	13-A15	
13-A15#	13-A15#	13-A24	13-A24#	13-A24#	13-A28	13-A28#	13-A28#	13-A35	13-A35#	13-A35#	13-A42	13-A42#	13-A42#	
13-A46	13-A46#	13-A46#	13-A55	13-A55#	13-A55#	13-A71	13-A71#	13-A71#	13-A79	13-A79#	13-A79#	13-A93	13-A93#	
13-A93#	13-B04	13-B04#	13-B04#	13-B11	13-B11#	13-B11#	13-B15	13-B15#	13-B15#	13-B17	13-B17#	13-B17#	13-B19	
13-B19#	13-B19#	13-B30	13-B30#	13-B30#	13-B41	13-B41#	13-B41#	13-B47	13-B47#	13-B47#	13-B51	13-B51#	13-B51#	
13-B58	13-B58#	13-B58#	13-B65	13-B65#	13-B65#	13-B86	13-B86#	13-B86#	13-B94	13-B94#	13-B94#	13-C16	13-C16#	
13-C16#	13-C23	13-C23#	13-C23#	13-C27	13-C27#	13-C27#	13-C29	13-C29#	13-C29#	13-C31	13-C31#	13-C31#	13-C38	
13-C38#	13-C38#	13-C44	13-C44#	13-C44#	13-C48	13-C48#	13-C48#	13-C61	13-C61#	13-C61#	13-C74	13-C74#	13-C74#	
13-C87	13-C87#	13-C87#	13-D00	13-D00#	13-D00#	13-D12	13-D12#	13-D12#	13-D40	13-D40#	13-D40#	13-D48	13-D48#	
13-D48#	13-D70	13-D70#	13-D70#	13-D77	13-D77#	13-D77#	13-D81	13-D81#	13-D81#	13-D83	13-D83#	13-D83#	13-D85	
13-D85#	13-D85#	13-D92	13-D92#	13-D92#	13-D98	13-D98#	13-D98#	13-E02	13-E02#	13-E02#	13-E09	13-E09#	13-E09#	
13-E21	13-E21#	13-E21#	13-E47	13-E47#	13-E47#	13-E61	13-E61#	13-E61#	13-E75	13-E75#	13-E75#	13-E78	13-E78#	
13-E80	13-E80#	13-E82	13-E82#	13-E82#	13-E86	13-E86#	13-E86#	13-E88	13-E88#	13-E88#				
FMT16	4-657#	13-140	13-142	13-151	13-249	13-342	13-467	13-563	13-656	13-778	13-876	13-953	13-:31	13-:31
	13-<14	13-<97	13-=53	13->49	13->52	13->62	13-?11	13-?61	13-@67	13-A65	13-B26	13-B67	13-B79	13-C54
	13-D35	13-E42	16-59	16-274	16-283	17-32	17-34	18-27	25-447	28-59	34-183			
FNC DTB	25-126	25-442	50-55#											
FNCMSK	4-505#	33-17	34-345											
GENBUF	13-48	13-160	13-258	13-351	13-476	13-572	13-665	13-790	13-:40	13-:41	13-<24	13-?70	13-@76	13-A75

NOTAVL	10-105	49-30#												
NOTPRS	10-102	49-29#												
NOTRM	10-96	49-27#												
NSA	4-647#													
OCC	4-628#													
OFD	4-661#	13-255	13-257											
OFFSET	4-514#	13-?78	13-215											
OM	4-569#	32-145	32-201	32-204	33-29	34-457	34-460							
ONES	13-474	13-570	13-663	52-21#										
OPE	4-684#													
OPI	4-575#	25-178	28-145	28-169	31-51	31-69	31-73	32-24	32-61	32-64	32-155	34-83	34-86	34-534
	35-42	50-57	50-58	50-59	50-60	50-61	50-62	50-63	50-64	50-65	50-66	50-67	50-68	50-69
	50-70	50-71	50-72	50-73	50-74	50-75	50-76	50-77	50-78	50-79	50-80	50-81	50-82	50-83
	50-84	50-85	50-86	50-87	50-88									
OR	4-745#													
PACACK	4-518#	16-118												
PAKACK	4-517#	4-518	15-125											
PAR	4-585#	24-135	24-139	24-151	28-30	32-24	32-29	32-34	34-41	34-46				
PAT	4-748#													
PDA	4-635#													
PFECH	43-59	43-164#												
PFECH1	43-164	43-165#												
PFECH2	43-164	43-168#												
PFECH3	43-164	43-171#												
PFECH4	43-164	43-173#												
PGE	4-742#													
PGM	4-565#	33-29												
PHA	4-636#													
PIP	4-561#	15-180	16-149	28-161	28-182	28-187	32-145	32-216	32-221	33-29	34-509	34-515	34-520	35-35
	35-72	35-77												
PIRQ	4-492#													
PIRQVE	4-492#													
PLFS	4-633#													
PRO	4-492#													
PR1	4-492#													
PR2	4-492#													
PR3	4-492#	41-5												
PR4	4-492#													
PR5	4-492#	12-50	41-9											
PR6	4-492#	10-26	13-7	20-40	21-34	22-5	23-11	27-22	29-14					
PR7	4-492#													
PRIERR	13-109	13-130	13-211	13-233	13-308	13-328	13-401	13-424	13-526	13-547	13-622	13-642	13-715	13-738
	13-840	13-860	13-935	13-:12	13-:93	13-:13	13-:70	13-:91	13-<53	13-<74	13-=34	13-=88	13->37	13->93
	13-?42	13-208	13-245	13-A15	13-A46	13-B15	13-B51	13-C27	13-C48	13-D81	13-E02	13-E86	24-39#	
PS	4-492	4-492#												
PSEL	4-729#													
PSW	4-492#													
PUT	13-64	13-75	13-99	13-120	13-176	13-201	13-223	13-274	13-298	13-318	13-367	13-391	13-414	13-492
	13-516	13-537	13-588	13-612	13-632	13-681	13-705	13-728	13-806	13-830	13-850	13-896	13-920	13-973
	13-997	13-:58	13-:83	13-:03	13-:59	13-:81	13-<42	13-<64	13-=18	13-=72	13->26	13->82	13-?31	13-?86
	13-?97	13-223	13-235	13-288	13-A04	13-A35	13-A93	13-B04	13-B30	13-B41	13-C16	13-C38	13-D70	13-D92
	13-E75	15-128	15-186	16-87	16-119	16-153	16-182	21-28#						
PUTBUF	7-0#	16-48	16-292*	21-31										
PUTINX	7-0#	13-57	13-93	13-169	13-195	13-267	13-292	13-360	13-385	13-485	13-510	13-581	13-606	13-674
	13-699	13-799	13-824	13-889	13-914	13-966	13-991	13-:51	13-:77	13-:50	13-<33	13-=07	13-=63	13->17
	13->73	13-?22	13-?79	13-?91	13-216	13-229	13-285*	13-286*	13-294	13-A84	13-C07	13-D61	13-E66	15-126*

RMCS31	7-0#													
RMCS30	7-0#													
RMDA	4-701#	13-58	13-170	13-268	13-361	13-486	13-582	13-675	13-800	13-890	13-967	13-:52	13-:51	13-<34
	13-08	13-64	13->18	13->74	13-?23	13-?80	13-a17	13-a96	13-A85	13-C08	13-D62	13-E67	16-63	
RMDAI	7-0#	25-628	25-699	59-14	59-16									
RMDAO	7-0#	13-39*	13-150*	13-248*	13-341*	13-466*	13-562*	13-655*	13-781*	13-875*	13-943*	13-951*	13-952*	13-:20*
	13-:21*	13-:30*	13-:30*	13-:39	13-<03*	13-<04	13-<13*	13-<22	13-<86*	13-<87	13-<95*	13-<96*	13-42*	13-43*
	13-51*	13-52*	13-96*	13-97*	13->05*	13->45*	13->46	13->51*	13->60*	13->61*	13-?01*	13-?02	13-?10*	13-?60*
	13-?68	13-a66*	13-a74	13-A64*	13-881*	13-D34*	13-E40*	13-E92*	13-E93	13-E96*	13-E97*	13-E98	13-F01*	16-57*
	16-233*	16-238*	16-241*	16-242	16-248	16-250	16-273*	16-281*	16-307	16-425*	17-24	25-588	25-589	25-628
	28-54	28-57	28-61	28-65	34-178	34-181	34-185	34-189						
RMDB	4-778#	13-18*	13-19	13-446	13-758	19-20								
RMDBI	7-0#	13-450	13-762											
RMDBO	7-0#													
RMDC	4-710#	13-59	13-171	13-269	13-362	13-487	13-583	13-676	13-801	13-891	13-968	13-:53	13-:52	13-<35
	13-09	13-65	13->19	13->75	13-?24	13-?81	13-a18	13-a97	13-A86	13-C09	13-D63	13-E68	16-64	
RMDCI	7-0#	25-712	59-14	59-16										
RMDCO	7-0#	13-38*	13-149*	13-247*	13-340*	13-465*	13-561*	13-654*	13-780*	13-878*	13-955*	13-:29*	13-:48	13-:49*
	13-:75*	13-:29*	13-<12*	13-<99*	13-55*	13->04*	13->59*	13-?09*	13-?50*	13-?51	13-?59*	13-a65*	13-A63*	13-880*
	13-D33*	13-E39*	13-E49	13-F02*	13-F03	16-56*	16-239	16-268	16-272*	16-306	16-416*	16-424*	17-23	25-587
	28-50	34-174												
RMDS	4-702#	10-86	10-98											
RMDSI	7-0#	15-122	15-180	16-115	16-149	24-104	25-34	25-36	25-81	25-98	25-132	25-136	25-192	25-219
	25-354	25-654	28-96	28-151	28-160	28-167	28-171	28-173	28-182	28-186	28-188	28-197	28-199	28-201
	28-211	28-215	28-217	30-124	31-23	31-25	31-27	31-37	31-39	31-41	32-68	32-106	32-144	32-153
	32-157	32-159	32-169	32-173	32-174	32-185	32-187	32-189	32-201	32-203	32-205	32-216	32-220	32-222
	33-28	34-88	34-90	34-119	34-256	34-322	34-457	34-459	34-461	34-508	34-515	34-519	34-521	34-532
	34-536	34-538	34-548	34-552	34-554	35-34	35-40	35-44	35-46	35-56	35-60	35-61*	35-62	35-72
	35-76	35-78	59-13											
RMDSO	7-0#													
RMDT	4-707#	10-92	10-94	43-26										
RMDT1	7-0#	59-5	59-17											
RMDTO	7-0#													
RMEC1	4-714#													
RMEC11	7-0#	18-32	59-14											
RMEC10	7-0#													
RMEC2	4-715#	19-15												
RMEC21	7-0#	18-53	19-14	30-92	33-94	59-15								
RMEC20	7-0#													
RMER1	4-703#	24-143												
RMER11	7-0#	13-855	13-860	13-861	13-C58	13-C63	13-C64	13-C97	13-D02	13-D03	13-E06	13-E11	13-E12	16-221
	18-23	18-25	24-135	24-150	24-152	25-76	25-152	25-179	25-222	25-237	25-280	25-357	25-383	25-398
	25-413	25-497	25-516	25-519	25-651	25-667	25-682	25-694	26-24	28-30	28-37	28-71	28-145	28-148
	28-169	30-69	31-51	31-55	31-57	31-58	31-69	31-71	31-72	31-83	31-85	31-86	31-97	31-99
	31-100	31-111	31-113	31-114	32-24	32-29	32-33	32-35	32-45	32-47	32-49	32-61	32-63	32-65
	32-78	32-80	32-82	32-155	32-232	32-236	32-238	32-240	32-250	32-252	32-254	32-264	32-268	32-270
	33-41	34-41	34-45	34-47	34-83	34-85	34-87	34-128	34-131	34-133	34-135	34-145	34-147	34-149
	34-159	34-161	34-163	34-193	34-283	34-286	34-290	34-292	34-301	34-303	34-305	34-315	34-317	34-318
	34-352	34-356	34-358	34-360	34-369	34-371	34-373	34-382	34-384	34-386	34-395	34-397	34-399	34-417
	34-419	34-421	34-427	34-483	34-485	34-487	34-534	35-42	59-13					
RMER10	7-0#	24-139												
RMER2	4-713#													
RMER21	7-0#	13-931	13-:08	13-:30	13-:84	13-A17	13-A19	13-A20	13-A48	13-A50	13-A51	13-C71	13-C76	13-C77
	13-C84	13-C89	13-C90	15-55	16-83	16-224	16-231	24-137	25-78	25-196	25-252	25-428	25-459	26-27
	28-32	28-90	28-93	28-104	28-132	28-135	28-184	28-213	30-114	32-31	32-92	32-99	32-101	32-103
	32-116	32-118	32-120	32-130	32-132	32-134	32-171	32-218	32-266	33-103	34-43	34-111	34-114	34-116

TST5	13-337#	41-1													
TST6	13-462#	41-1													
TST7	13-558#	41-1													
TSTNMB	43-36*	43-37*	43-39	43-151#											
TSTPRP	13-52	13-164	13-262	13-355	13-480	13-576	13-669	13-794	13-884	13-961	13-:44	13-:45	13-<28	13-=05	
	13-=61	13->13	13->69	13-?18	13-?74	13-?80	13-A28	13-A79	13-B94	13-D48	13-E61	15-38#			
TSTQUE	7-0#	12-8	12-9*	12-51	13-2	13-33	13-146	13-244	13-337	13-462	13-558	13-651	13-775	13-872	
	13-948	13-:26	13-:26	13-<09	13-<92	13-=48	13->02	13->57	13-?07	13-?56	13-?62	13-A60	13-873	13-D27	
	13-E36	14-11	14-13*	14-17	14-17*	27-24	29-17	30-54	33-53						
TYPBN	43-141	46-1#													
TYPDS	14-20	14-20	43-135	46-1#											
TYPE	10-6	10-28	10-42	10-48	10-53	10-59	10-76	10-80	10-82	10-96	10-102	10-105	10-109	10-117	
	10-120	10-121	10-122	10-129	11-11	11-16	11-17	11-19	11-20	11-28	11-32	11-34	11-40	11-44	
	11-48	11-54	11-61	11-62	11-64	11-69	11-75	11-86	11-87	11-92	11-100	12-3	12-6	12-7	
	12-14	12-25	12-30	12-60	12-61	13-44	13-156	13-254	13-347	13-472	13-568	13-661	13-786	13-:36	
	13-:37	13-<20	13-?66	13-?72	13-A71	13-B86	13-D40	13-E47	14-20	14-20	14-20	37-1	38-1	39-1	
	40-1	42-1	42-1	43-20	43-21	43-31	43-32	43-38	43-43	43-45	43-53	43-96	43-102	43-106	
	43-118	43-124	43-126	43-144	43-146	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	
	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	46-1#	47-1			
TYPOC	10-8	11-33	11-47	43-138	44-1	46-1#									
TYPON	46-1#														
TYPOS	10-45	10-56	10-81	12-16	12-62	43-22	43-39	43-44	43-46	46-1#					
U0	4-752#														
U1	4-752#														
U2	4-752#														
UNS	4-574#	30-70	31-51	31-55	31-59	32-232	32-264	32-269	34-283	34-286	34-291				
UNMSK	4-756#	24-48	24-50												
UNTOFF	10-109	49-31#													
UNTON	10-120	49-32#													
UPE	4-739#	25-266													
USE	4-695#	13-C52	17-29												
USRFIL	16-279	16-282	16-333	61-12#											
VV	4-568#	15-122	16-115	25-192	28-96	28-161	28-162	28-211	28-216	31-23	31-26	32-106	32-145	32-146	
	32-169	32-175	33-29	34-90	34-119	34-509	34-510	34-548	34-553	35-35	35-36	35-56	35-61		
WC	4-638#	30-80	33-70												
WCD	4-525#														
WCE	4-738#	13-428	13-434	13-742	13-747	25-165	50-77	50-78							
WCEHI	4-763#														
WCELO	4-764#														
WCF	4-583#	4-590	25-236	34-483	34-486										
WCH	4-526#	13-316	13-412	13-630	13-726	13-848	13-:01								
WD	4-529#														
WH	4-530#	13-42	13-92	13-154	13-194	13-252	13-291	13-345	13-384	13-470	13-509	13-566	13-605	13-659	
	13-698	13-784	13-823	13-:34	13-:76	13-:35	13-:49	13-<18	13-<32	13-?64	13-?90	13-?70	13-?92	13-A69	
	13-A83	13-B02	13-B84	13-C06	13-D38	13-D60	13-E45	13-E65	34-346						
WLE	4-577#	4-590	25-217	25-221	25-234	25-249	25-264	34-283	34-315	34-319	34-327	50-81	50-82		
WRL	4-563#	25-219	34-322												
XSIZ	10-70#	11-71	11-101												
XXDP	7-0#	10-33*	10-36*	10-37	10-39*	10-44	10-55	10-113	10-115						
Y	49-36#														
ZEROS	13-46	13-158	13-256	13-349	13-788	13-:38	13-A73	13-B88	13-D42	52-38#					

TRMTRP	46-1#																	
TYPBIN	4-492#																	
TYPDEC	4-492#	14-20	14-20															
TYPNAM	4-480#	4-492#	10-28															
TYPNUM	4-492#																	
TYPOCS	4-492#	10-81	12-16	12-62	43-22	43-39	43-44	43-46										
TYPOCT	4-492#	11-33	44-1															
TYPTXT	4-492#	10-6	10-42	10-53	10-59	12-30	14-20	14-20										
XPER	4-12#	8-3	8-6	8-9	8-12	8-15	8-18	8-21	8-24	8-27	8-30	8-33	8-36	8-39				
	8-42	8-45	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81				
	8-84	8-87	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123				
	8-126	8-130	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-157	8-160	8-163	8-166	8-169				
	8-172	8-175	8-178	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212				
	8-215	8-218	8-221	8-224	8-227	8-230	8-233	8-236	8-239	8-242	8-245	8-248	8-251	8-254				
	8-257	8-260	8-263	8-266	8-269	8-272	8-275	8-278	8-281	8-284	8-287	8-290	8-293	8-296				
	8-299	8-302	8-305	8-308	8-311	8-314	8-317	8-320	8-323	8-326	8-329	8-332	8-335	8-338				
	8-341	8-344	8-347	8-351	8-354	8-357	8-360	8-363	8-366	8-369	8-372	8-375	8-378	8-381				
	8-384	8-387	8-390	8-393	8-396	8-399	8-402	8-405	8-408	8-411	8-414	8-417	8-420	8-423				
	8-426	8-429	8-432	8-435	8-438	8-441	8-444	8-447	8-450	8-453	8-456	8-459	8-462	8-465				
	8-468	8-471	8-474	8-477	8-480	8-483	8-486	8-489	8-492	8-495	8-498	8-501	8-504	8-507				
	8-510	8-513	8-516	8-519	8-522	8-525	8-528	8-531	8-534	8-537	8-540	8-543	8-546	8-549				
	8-552	8-556	8-559	8-563	8-566	8-569	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600				
	8-603	8-606	8-609	8-612	8-615	8-618	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642				
	8-645	8-648	8-651	8-654	8-657	8-660	8-663	8-666	8-669	8-672	8-675	8-678	8-681	8-684				
	8-687	8-690	8-693	8-696	8-699	8-702	8-705	8-708	8-711	8-714	8-717	8-720	8-723					