

RM05/3/2

RM05/3/2 FCTNL TST 3 AH-F928A-MC
CZRMOAO FICHE 1 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



Table with multiple columns and rows of data, including headers like 'CZRM0A0', 'FCTNL TST 3', and various alphanumeric codes. The table contains a dense grid of data points, likely representing test results or system configurations. The text is small and difficult to read due to the high density and low contrast of the image.

RM05/3/2

RM05/3/2 FCTNL TST 3 AH-F928A-MC
CZRMOAO FICHE 2 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



Microfiche grid containing multiple frames of data, likely test results or technical specifications. The data is organized in columns and rows, with some frames containing headers and numerical values.

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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-F927A-MC
PRODUCT NAME: CZRMOAO RM05/3/2 FCTNL TST 3
DATE CREATED: APRIL 1980
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 TO THE PURSHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1980 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

- 1. INTRODUCTION
 - 1. ABSTRACT
 - 2. UNIT UNDER TEST
- 2. OPERATING REQUIREMENTS
 - 1. HARDWARE REQUIREMENTS
 - 2. MEDIA REQUIREMENTS
 - 3. PREREQUISITE DIAGNOSTIC PROGRAMS
- 3. OPERATING PROCEDURE
 - 1. LOADING
 - 2. SWITCH OPTIONS
 - 3. STARTING
 - 4. HALTING
 - 5. RESTARTING
- 4. OPERATOR INTERFACE
 - 1. PROGRAM I.D.
 - 2. CONSOLE DIALOGUE
 - 3. PROGRESS REPORTS
 - 4. PERFORMANCE REPORTS
 - 5. PROGRAM HALTS
 - 6. ERROR REPORTS
- 5. ENVIRONMENTAL SUPPORT
 - 1. PROCESSOR COMPATIBILITY
 - 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 RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RM05/3/2 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 RM05/3/2 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

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

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM05/3/2 DISK SUBSYSTEM WHICH CONSISTS OF THE RH MASSBUS CONTROLLER, THE RM05/3/2 MASSBUS ADAPTER, AND THE STORAGE MODULE DISK DRIVE. NOTE THAT A DISK PACK IS REQUIRED FOR TESTING AND IS CONSIDERED AN INTEGRAL OF THE STORAGE MODULE DISK DRIVE.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST:

PDP-11 PROCESSOR
16K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
1 TO 8 DISK DRIVES (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

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

2.2 MEDIA REQUIREMENTS

EACH UNIT BEING TESTED MUST BE LOADED WITH A SCRATCH DISK PACK BEFORE TESTING BEGINS ON THAT UNIT. THE DISK MUST BE FORMATTED AND CONTAIN A READABLE COPY OF THE MFG AND USR BAD SECTOR FILES.

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RM05/3/2 DISKLESS DIAGNOSTIC, PART 1 & 2

RM05/3/2 FUNCTIONAL TEST, PART 1 & 2

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

115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171

3.4 HALTING

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

NOTE: IF THE PROGRAM IS HALTED BY ANY OTHER MEANS, BAD HEADER INFORMATION MAY BE LEFT ON THE DISK PACK. THIS OF COURSE DEPENDS ON WHICH TEST IS BEING PERFORMED AT THE TIME OF THE HALT.

3.5 RESTARTING

THE PROGRAM CAN BE RESTARTED AT ADDRESS 200.

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

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

4.2 CONSOLE DIALOGUE

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

THE FIRST QUESTION TYPED OUT IS: "TYPE HELP TEXT (Y/N) ?". IF THE OPERATOR RESPONDS WITH A 'Y', THE PROGRAM WILL TYPE A BRIEF HELP MESSAGE WHICH WILL LIST SWITCH OPTIONS, ETC.

NOTE: THE FIRST QUESTION IS ONLY ASKED ON THE INITIAL PROGRAM START AND NOT ON SUBSEQUENT START-UP'S.

THE SECOND QUESTION TYPED IS, "CHANGE ADDRESSES (Y/N) ?". IF THE UNIBUS ADDRESS OF THE RH/RM IS NON STANDARD, THE OPERATOR SHOULD RESPOND WITH A 'Y', THEN ANSWER SUBSEQUENT QUESTIONS TO SPECIFY THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY. IF THE OPERATOR RESPONSE IS A 'N', THE PROGRAM SKIPS TO THE THIRD QUESTION.

THE THIRD 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 THE NUMBER(S) OF THE DRIVE(S) HE WANTS TESTED AND TERMINATE HIS INPUT WITH A 'CARRIAGE RETURN'.

NOTE: THE LONG VERSION OF THE THIRD QUESTION IS ONLY TYPED ON THE INITIAL PROGRAM START. ON ALL SUBSEQUENT STARTS, ONLY THE 'DRIVE(S):' PROMPT IS TYPED.

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

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

4.3 PROGRESS REPORTS

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

4.4 PERFORMANCE REPORT

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

4.5 PROGRAM HALTS

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

4.6 ERROR REPORTS

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

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET, AND PROVIDING THAT DATA THROUGHPUT ON THE SYSTEM IS SUFFICIENT TO SUSTAIN DATA TRANSFER OPERATIONS.

5.2 DUAL PORT CONFIGURATIONS

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

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RM05/3/2 ADAPTER BUT IS EXECUTABLE ON RM05/3/2 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 RM05/3/2 SUSBYSTEM FUNCTIONAL TEST.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE IS NOT USED DURING THE RM05/3/2 SUSBYSTEM FUNCTIONAL TEST. CAPABILITIES OF THE MASSBUS CONTROLLER.

5.5 ACT11, APT11 COMPATIBILITY

THE RM05/3/2 SUSBYSTEM 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 RM05/3/2 SUSBYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RM05/3/2 IS THE XXDP LOADING DEVICE.

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

6.0 TEST DESCRIPTION

CONTROLLER ACCESS TEST

PURPOSE:

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

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

PROCEDURE:

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

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

DEVICE AVAILABLE TEST

PURPOSE:

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

PROCEDURE:

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

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

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

.BUS TIMEOUT - THE MASSBUS CONTROLLER FAILED TO DETECT A NONEXISTENT DEVICE;

.DEVICE AVAILABLE - THE DEVICE IS AVAILABLE FOR TESTING.

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

DRIVE TYPE TEST

PURPOSE:

TO VERIFY THAT THE UNIT UNDER TEST IS AN RM05, RM03 OR RM02 SINGLE PORT OR DUAL PORT SUBSYSTEM.

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

PROCEDURE:

THIS TEST READS THE DRIVE TYPE REGISTER, RMDT, OF THE SELECTED DEVICE AND VERIFIES THAT THE DEVICE IS A SINGLE PORT OR DUAL PORT RM05/3/2 SUBSYSTEM. IF THE SELECTED DEVICE IS NOT AN RM05/3/2, THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH WILL SELECT THE NEXT DEVICE FOR TESTING.

WRITE/READ DATA TESTS

PURPOSE:

TO TEST WRITE DATA AND READ DATA FUNCTIONALITY OF THE RM05/3/2 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. THEN, THE TEST FORMATS THE SECTOR BEING USED, WHICH MAY VARY FROM THE PROGRAM LISTING, BECAUSE SECTORS ARE SUBSTITUTED DURING RUN TIME IF THE SELECTED SECTOR IS LISTED IN THE BAD BLOCK TABLE OF THE LAST TRACK. FOLLOWING THAT, THE TEST PERFORMS ANY EXPLICIT SEEKS REQUIRED FOR THE CONDITIONS OF THE TEST. REGISTERS ARE PRESET AND THE WRITE

DATA COMMAND IS EXECUTED. WHEN THE WRITE COMMAND IS COMPLETE, THE TEST STORES ALL SUBSYSTEM STATUS AND CHECKS FOR PRIMARY ERRORS WHICH PRECLUDE OTHER STATUS CHECKS. IF THERE ARE NO PRIMARY ERRORS, THE TEST VERIFIES THE RESULTS OF THE WRITE COMMAND AND THEN CHECKS FOR SECONDARY ERRORS. LOOP ADDRESSES ARE MODIFIED FOLLOWING THE SUCCESSFUL COMPLETION OF THE WRITE COMMAND IN ORDER TO SHORTEN EXECUTION TIMES AND ENHANCE SCOPING LOOPS, THEN THE PROGRAM EXECUTES THE READ DATA PORTION OF THE TEST, VERIFYING THE SAME TYPE OF ERRORS AS IN THE WRITE COMMAND.

NOTE:

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

WRITE, READ ZEROS TEST

THE TEST WRITES AND READS AN ALL ZEROS DATA FIELD, CAUSING

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

THE DRIVE TO USE NORMAL WRITE GATE THOUGHOUT THE WRITE PROCESS.

WRITE, WRITE CHECK ZEROS TEST

THE TEST WRITES AND WRITE CHECKS AN ALL ZEROS DATA FIELD, VERIFYING THAT THERE ARE NO ERRORS.

WRITE, WRITE CHECK ZEROS W/ WCE ERROR TEST

THE TEST WRITES AN ALL ZEROS DATA FIELD, THEN COMPLEMENTS EACH BIT IN THE LAST WORD OF THE WRITE BUFFER. A WRITE CHECK COMMAND IS EXECUTED AND THE TEST VERIFIES THAT THE CORRECT WRITE CHECK ERROR IS DETECTED.

WRITE, READ ONES TEST

THE TEST WRITES AND READS AN ALL ONES DATA FIELD, CAUSING THE DRIVE TO USE NORMAL WRITE GATE THOUGHOUT THE WRITE PROCESS.

WRITE, WRITE CHECK ONES TEST

THE TEST WRITES AND WRITE CHECKS AN ALL ONES DATA FIELD.

WRITE, WRITE CHECK ONES W/ WCE ERROR TEST

THE TEST WRITES AN ALL ONES DATA FIELD, THEN PERFORMS A WRITE CHECK DATA COMMAND AFTER COMPLEMENTING EACH BIT IN THE LAST WORD OF THE WRITE BUFFER. THE TEST VERIFIES THAT THE CORRECT WRITE CHECK ERROR IS DETECTED.

WRITE, WRITE CHECK MULTIPLE SECTORS TEST

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD

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

MOTION DURING DATA TRANSFER. THE WRITE DATA COMMAND FOLLOWS, WITH THE WORD COUNT EQUAL TO MULTIPLE SECTORS. THE SAME SECTORS ARE VERIFIED WITH A WRITE CHECK DATA COMMAND.

WRITE, READ WITH IMPLIED SEEK TEST

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

WRITE, WRITE CHECK WITH HEAD SWITCHING TEST

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE WRITE DATA COMMAND STARTS WITH CYLINDER 0, TRACK 0, SECTOR 31. THE WORD COUNT IS EQUAL TO MULTIPLE SECTORS WHICH CAUSES THE SUBSYSTEM TO SWITCH FROM TRACK 0 TO TRACK 1 AFTER THE FIRST OF THE MULTIPLE SECTORS ARE WRITTEN. THE SAME SECTORS ARE VERIFIED WITH A WRITE CHECK DATA COMMAND, USING THE SAME WORD COUNT AND STARTING SECTOR.

WRITE, WRITE CHECK WITH MID-TRANSFER SEEK TEST

THIS TEST WRITES MULTIPLE SECTORS STARTING WITH CYLINDER 0, LAST TRACK AND SECTOR 31., CAUSING A MID-TRANSFER SEEK AFTER THE FIRST SECTOR IS WRITTEN. THE SAME SECTORS ARE VERIFIED WITH A WRITE CHECK DATA COMMAND.

WRITE, READ W/ HCE ERROR TEST

A SECTOR IS FORMATTED WITH AN INCORRECT HEADER, THEN THE TEST WRITES AND READS DATA FROM THE SECTOR AND VERIFIES THAT THE CORRECT ERROR IS DETECTED. EACH BIT POSITION OF BOTH HEADER WORDS IS TESTED IN THIS MANNER.

WRITE, READ W/ HCI TEST

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

A SECTOR IS FORMATTED WITH AN INCORRECT HEADER, THEN WRITTEN AND READ WITH HEADER COMPARE INHIBITED. THE TEST VERIFIES THAT NO ERROR IS DETECTED.

WRITE, READ W/ IVC ERROR TEST

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

WRITE, READ W/ ABORT TEST

THE TEST SETS AN ERROR IN THE ERROR REGISTER AND EXECUTES A WRITE DATA COMMAND VERIFYING THAT 'PIP' REMAINS INACTIVE. THE SAME PROCEDURE IS USED FOR READ DATA COMMAND.

WRITE, READ EACH CURRENT LEVEL TEST

THE TEST WRITES AND READS ON EACH OF THE FOLLOWING CYLINDERS IN ORDER TO WRITE AT EACH POSSIBLE CURRENT THRESHOLD.

CYLINDER	0	110 MA
"	128	104 MA
"	256	97 MA
"	384	91 MA
"	512	84 MA
"	640	77 MA
"	768	70 MA

WRITE, WRITE CHECK W/ CURRENT LEVEL SWITCHING TEST

THE TEST WRITES 2 SECTORS STARTING WITH THE LAST SECTOR OF

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

CYLINDER 383. THE FIRST SECTOR IS WRITTEN AT ONE CURRENT LEVEL AND THE SECOND SECTOR IS WRITTEN AT ANOTHER CURRENT LEVEL. BOTH SECTORS ARE VERIFIED WITH A WRITE CHECK DATA COMMAND.

WRITE, READ EARLY PEAK SHIFT TEST

THIS TEST WRITES AND READS A SINGLE SECTOR USING A DATA PATTERN DESIGNED TO INTRODUCE EARLY PEAK SHIFT.

WRITE, READ MIXED PEAK SHIFT TEST

THIS TEST WRITES AND READS A SINGLE SECTOR USING A DATA PATTERN DESIGNED TO INTRODUCE MIXED PEAK SHIFT.

WRITE, READ EACH TRACK TEST

THE TEST WRITES AND READS WITH EACH HEAD USING A MIX OF EARLY, NORMAL AND LATE WRITE GATES.

READ, WRITES CHECK MULTIPLE SECTORS IN OFFSET MODE TEST

THE TEST EXECUTES READ DATA AND WRITE CHECK DATA COMMANDS IN OFFSET MODE IN BOTH OFFSET DIRECTIONS WITH THE TRANSFER SIZE OF TWO SECTORS AT A TIME.

1
486
487

488

489

490
491

```

;PROGRAM REVISION #001

.TITLE CZRMOAO RM05/3/2 FCTNL TST 3
;*COPYRIGHT (C) 1980
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY MIKE LEAVITT
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.
;*
.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

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK = 1100
ERROR = EMT          ;;BASIC DEFINITION OF ERROR CALL
SCOPE = IOT          ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS
HT = 11              ;;CODE FOR HORIZONTAL TAB
LF = 12              ;;CODE FOR LINE FEED
CR = 15              ;;CODE FOR CARRIAGE RETURN
CRLF = 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS = 177776         ;;PROCESSOR STATUS WORD
PSW=PS
STKLMT = 177774     ;;STACK LIMIT REGISTER
PIRQ = 177772       ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR = 177570       ;;HARDWARE SWITCH REGISTER
DDISP = 177570      ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0 = %0             ;;GENERAL REGISTER
R1 = %1             ;;GENERAL REGISTER
R2 = %2             ;;GENERAL REGISTER
R3 = %3             ;;GENERAL REGISTER

```

001100
104000
000004

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

000000
000001
000002
000003

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 RM REGISTER BIT DEFINITIONS

;*RMCS1 CONTROL STATUS REGISTER

```

492
493
494
495
496
497 004000 DVA = BIT11           ;DEVICE AVAILABLE-READ ONLY
498 000040 F4 = BIT05           ;FUNCTION CODE
499 000020 F3 = BIT04           ;FUNCTION CODE
500 000010 F2 = BIT03           ;FUNCTION CODE
501 000004 F1 = BIT02           ;FUNCTION CODE
502 000002 F0 = BIT01           ;FUNCTION CODE
503 000001 GO = BIT00          ;GO BIT
504 000077 FNCMSK = 000077     ;FUNCTION CODE MASK
505
    
```

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

```

506
507 000000 NOP = 000000         ;NOP COMMAND
508 000002 ILF02 = 000002      ;ILLEGAL COMMAND
509 000004 SEEK = 000004       ;SEEK COMMAND
510 000006 RECAL = 000006      ;RECALIBRATE COMMAND
511 000010 DRVCLR = 000010     ;DRIVE CLEAR COMMAND
512 000012 RELEASE = 000012    ;RELEASE COMMAND
513 000014 OFFSET = 000014     ;OFFSET COMMAND
514 000016 RTC = 000016        ;RETURN TO CENTERLINE COMMAND
515 000020 RIP = 000020        ;READ IN PRESET COMMAND
516 000022 PAKACK = 000022     ;PACK ACKNOWLEDGE COMMAND
517 000022 PACACK = PAKACK
518 000024 ILF24 = 000024      ;ILLEGAL COMMAND
519 000026 ILF26 = 000026      ;ILLEGAL COMMAND
    
```

```

520      000030      SEARCH = 000030      ;SEARCH COMMAND
523      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
524      000050      WCD     = 000050      ;WRITE CHECK DATA COMMAND
525      000052      WCH     = 000052      ;WRITE CHECK HEADER AND DATA
526      000054      ILF54  = 000054      ;ILLEGAL COMMAND
527      000056      ILF56  = 000056      ;ILLEGAL COMMAND
528      000060      WD      = 000060      ;WRITE DATA COMMAND
529      000062      WH      = 000062      ;WRITE HEADER AND DATA COMMAND
530      000064      ILF64  = 000064      ;ILLEGAL COMMAND
531      000066      ILF66  = 000066      ;ILLEGAL COMMAND
532      000070      RD      = 000070      ;READ DATA COMMAND
533      000072      RH      = 000072      ;READ HEADER AND DATA COMMAND
534      000074      ILF74  = 000074      ;ILLEGAL COMMAND
535      000076      ILF76  = 000076      ;ILLEGAL COMMAND
536
537      ;*RMDA  DISK ADDRESS REGISTER
538
539      ;TRACK ADDRESS DEFINITIONS
540      010000      TA16   = BIT12      ;TRACK ADDRESS 16.
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!'ICE!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	020024	SNGPRT	= 020024	;SINGLE PORT DRIVE TYPE (RM02)
653	024024	DULPRT	= 024024	;DUAL PORT DRIVE TYPE (RM02)
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	000200	OFD	= BIT07	;OFFSET FORWARD
661				
662		;*RMDC DESIRED CYLINDER ADDRESS REGISTER		
663				
664	001777	CYLSK	= 001777	;MASK FOR CYLINDER ADDRESS
665				
666		;*RMMR2 MAINTENANCE REGISTER #2		
667				
668		;READ ONLY BITS		
669	100000	RQA	= BIT15	;PORT A REQUEST
670	040000	RQB	= BIT14	;PORT B REQUEST
671	020000	TAG	= BIT13	;TAG CONTROL
672	010000	TST	= BIT12	;COMMAND SEQUENCE TEST BIT
673	004000	CC	= BIT11	;CONTROL OR CYLINDER TAG
674	002000	CH	= BIT10	;CONTROL OR HEAD TAG
675	001000	BB09	= BIT09	;TAG BUS
676	000400	BB08	= BIT08	;TAG BUS
677	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

```

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

```

727      001000      A17      = BIT09      :ADDRESS EXTENSION
728      000400      A16      = BIT08      :ADDRESS EXTENSION
729      000200      RDY      = BIT07      :READY-READ ONLY
730      000100      IE       = BIT06      :INTERRUPT ENABLE
731
732      ;*RMCS2 RH CONTROL STATUS REGISTER #2
733
734      100000      DLT      = BIT15      :DATA LATE-READ ONLY
735      040000      WCE      = BIT14      :WRITE CHECK ERROR-READ ONLY
736      020000      UPE      = BIT13      :UNIBUS PARITY ERROR
737      010000      NED      = BIT12      :NONEXISTANT DRIVE-READ ONLY
738      004000      NEM      = BIT11      :NONEXISTANT MEMORY-READ ONLY
739      002000      PGE      = BIT10      :PROGRAM ERROR-READ ONLY
740      001000      MXF      = BIT09      :MISSED TRANSFER
741      000400      MDPE     = BIT08      :MASSBUS DATA BUS PARITY ERROR-READ ONLY
742      000200      OR       = BIT07      :OUTPUT READY-READ ONLY
743      000100      IR       = BIT06      :INPUT READY-READ ONLY
744      000040      CLR      = BIT05      :CONTROLLER CLEAR
745      000020      PAT      = BIT04      :PARITY TEST
746      000010      BAI      = BIT03      :UNIBUS ADDRESS INCREMENT INHIBIT
749      000004      U2       = BIT02      :UNIT SELECT
          000002      U1       = BIT01      :UNIT SELECT
          000001      U0       = BIT00      :UNIT SELECT
750
751      ;UNIT SELECT MASK
752
753      000007      UNTMSK   = 7           ;UNIT SELECT MASK
754
755      ;*RMCS3 RH70 CONTROL STATUS REGISTER #3
756
757      100000      APE      = BIT15      :ADDRESS PARITY ERROR
758      040000      DPEHI    = BIT14      :DATA PARITY ERROR HIGH WORD
759      020000      DPELO    = BIT13      :DATA PARITY ERROR LOW WORD
760      010000      WCEHI    = BIT12      :WRITE CHECK ERROR HIGH WORD
761      004000      WCELO    = BIT11      :WRITE CHECK ERROR LOW WORD
762      002000      DBL      = BIT10      :DOUBLE WORD TRANSFER
763      000100      IE       = BIT06      :INTERRUPT ENABLE
764      000010      IPCK3    = BIT03      :INVERT PARITY CHECK
765      000004      IPCK2    = BIT02      :INVERT PARITY CHECK
766      000002      IPCK1    = BIT01      :INVERT PARITY CHECK
767      000001      IPCK0    = BIT00      :INVERT PARITY CHECK
768
769      .SBTTL RH11/RH70 CONTROLLER REGISTER INDEX VALUES
770
771      000000      RMCS1     = 00         ;CONTROL, STATUS REGISTER #1
772      000002      RMWC      = 02         ;WORD COUNT REGISTER
773      000004      RMBA      = 04         ;BUS ADDRESS REGISTER
774      000010      RMCS2     = 10         ;CONTROL, STATUS REGISTER #2
775      000022      RMDB      = 22         ;DATA BUFFER
776      000050      RMBAE     = 50         ;BUS ADDRESS EXTENSION
777      000052      RMCS3     = 52         ;CONTROL, STATUS REGISTER #3
778
779      176700      ABASE     = 176700     ;UNIBUS ADDRESS
780      120254      AVECT1    = 120254     ;UNIBUS VECTOR ADDRESS AND PRIORITY
781
782
783
784      .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
              .=174
DISPREG:    .WORD 0          ;;SOFTWARE DISPLAY REGISTER
SWREG:      .WORD 0          ;;SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)
000200      000137 005420
785          JMP @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
786          .SBTTL ACT11 HOOKS

              ;:*****
              ;HOOKS REQUIRED BY ACT11
000204      000046
000046      037650
              $SVPC=.          ;SAVE PC
              .=46
              $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
000052      000052
000052      000000
              .=52
              .WORD 0          ;;2)SET LOC.52 TO ZERO
              .=$SVPC          ;; RESTORE PC

787          .=1100
788          .SBTTL APT PARAMETER BLOCK
789          ;:*****
              ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
              ;:*****
000024      001100
000024      000024
              .SX=.          ;;SAVE CURRENT LOCATION
000044      000200
              .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
              200          ;;FOR APT START UP
              .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044      001100
              $APTHDR       ;;POINT TO APT HEADER BLOCK
              .=$X          ;;RESET LOCATION COUNTER
              ;:*****
              ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
              ;INTERFACE SPEC.

001100      $APTHD:
001100      000000
001102      001222
001104      000001
001106      000002
001110      000002
001112      000042
790          TAGADR=.
              $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
              $MADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
              $TSTM: .WORD 1          ;;RUN TIM OF LONGEST TEST
              $PASTM: .WORD 2         ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
              $UNITM: .WORD 2         ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
              .WORD $ETEND-$MAIL/2    ;;LENGTH MAILBOX-ETABLE(WORDS)

```


0

.SBTTL COMMON TAGS

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

001114	001114			SCMTAG: .TAGADR	::START OF COMMON TAGS
001114	000000			.WORD 0	::CONTAINS THE TEST NUMBER
001116	000			\$TSTNM: .BYTE 0	::CONTAINS ERROR FLAG
001117	000			\$ERFLG: .BYTE 0	::CONTAINS SUBTEST ITERATION COUNT
001120	000000			\$ICNT: .WORD 0	::CONTAINS SCOPE LOOP ADDRESS
001122	000000			\$LPADR: .WORD 0	::CONTAINS SCOPE RETURN FOR ERRORS
001124	000000			\$LPERR: .WORD 0	::CONTAINS TOTAL ERRORS DETECTED
001126	000000			\$ERTTL: .WORD 0	::CONTAINS ITEM CONTROL BYTE
001130	000			\$ITEMB: .BYTE 0	::CONTAINS MAX. ERRORS PER TEST
001131	001			\$ERMAX: .BYTE 1	::CONTAINS PC OF LAST ERROR INSTRUCTION
001132	000000			\$ERRPC: .WORD 0	::CONTAINS ADDRESS OF 'GOOD' DATA
001134	000000			\$GDADR: .WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
001136	000000			\$BDADR: .WORD 0	::CONTAINS 'GOOD' DATA
001140	000000			\$GDDAT: .WORD 0	::CONTAINS 'BAD' DATA
001142	000000			\$BDDAT: .WORD 0	::RESERVED--NOT TO BE USED
001144	000000			.WORD 0	
001146	000000			.WORD 0	
001150	000			\$AUTOB: .BYTE 0	::AUTOMATIC MODE INDICATOR
001151	000			\$INTAG: .BYTE 0	::INTERRUPT MODE INDICATOR
001152	000000			.WORD 0	
001154	177570			\$SWR: .WORD DSWR	::ADDRESS OF SWITCH REGISTER
001156	177570			\$DISPLAY: .WORD DDISP	::ADDRESS OF DISPLAY REGISTER
001160	177560			\$TKS: 177560	::TTY KBD STATUS
001162	177562			\$TKB: 177562	::TTY KBD BUFFER
001164	177564			\$TPS: 177564	::TTY PRINTER STATUS REG. ADDRESS
001166	177566			\$TPB: 177566	::TTY PRINTER BUFFER REG. ADDRESS
001170	000			\$NULL: .BYTE 0	::CONTAINS NULL CHARACTER FOR FILLS
001171	002			\$FILLS: .BYTE 2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			\$FILLC: .BYTE 12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			\$TPFLG: .BYTE 0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			\$TMP0: .WORD 0	::USER DEFINED
001176	000000			\$TMP1: .WORD 0	::USER DEFINED
001200	000000			\$TMP2: .WORD 0	::USER DEFINED
001202	000000			\$TMP3: .WORD 0	::USER DEFINED
001204	000000			\$TMP4: .WORD 0	::USER DEFINED
001206	000000			\$TIMES: 0	::MAX. NUMBER OF ITERATIONS
001210	000000			\$ESCAPE: 0	::ESCAPE ON ERROR ADDRESS
001212	207	377	377	\$BELL: .ASCIZ <207><377><377>	::CODE FOR BELL
001216	077			\$QUES: .ASCII /?/	::QUESTION MARK
001217	015			\$CRLF: .ASCII <15>	::CARRIAGE RETURN
001220	012	000		\$LF: .ASCIZ <12>	::LINE FEED

.SBTTL APT MAILBOX-ETABLE

\$EVEN
\$MAIL: ::APT MAILBOX
\$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ::TEST NUMBER

001222
001222 000000
001224 000000
001226 000000

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	\$DEVCM: .WORD	ADEVCM	::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		

```

0          .SBTTL  USER DEFINED TAGS

001326 000000 CTLFG: .WORD 0          :CONTAINS CONTROL-C FLAG
001330 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 RM05/3/2.

001332      000 LSTRK: .BYTE 0          :LO BYTE = 0
001333      000          .BYTE 0          :HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT
                                         :UNDER TEST. RM02/3 = 4., RM05 = 18.
  
```

```

;THE REGISTER INPUT BUFFER IS USED FOR
;STORING DRIVE STATUS
  
```

001334

GETBUF:

```

;REGISTER INPUT BUFFER
001334 000000 RMCS1I: .WORD 0          :CONTROL, STATUS REGISTER #1
001336 000000 RMWCI:  .WORD 0          :WORD COUNT REGISTER
001340 000000 RMBAI:  .WORD 0          :BUS ADDRESS REGISTER
001342 000000 RMDAI:  .WORD 0          :DISK ADDRESS REGISTER
001344 000000 RMCS2I: .WORD 0          :CONTROL, STATUS REGISTER #2
001346 000000 RMDSI:  .WORD 0          :DRIVE STATUS REGISTER
001350 000000 RMER1I: .WORD 0          :ERROR REGISTER #1
001352 000000 RMASI:  .WORD 0          :ATTENTION SUMMARY REGISTER
001354 000000 RMLAI:  .WORD 0          :LOOK AHEAD REGISTER
001356 000000 RMDBI:  .WORD 0          :DATA BUFFER
001360 000000 RMMR1I: .WORD 0          :MAINTENANCE REGISTER #1
001362 000000 RMDTI:  .WORD 0          :DRIVE TYPE REGISTER
001364 000000 RMSNI:  .WORD 0          :SERIAL NUMBER REGISTER
001366 000000 RMOFI:  .WORD 0          :OFFSET REGISTER
001370 000000 RMDCI:  .WORD 0          :DESIRED CYLINDER REGISTER
001372 000000 RMHRI:  .WORD 0          :HOLDING REGISTER
001374 000000 RMMR2I: .WORD 0          :MAINTENANCE REGISTER #2
001376 000000 RMER2I: .WORD 0          :ERROR REGISTER #2
001400 000000 RMEC1I: .WORD 0          :ECC POSITION REGISTER
001402 000000 RMEC2I: .WORD 0          :ECC PATTERN REGISTER
001404 000000 RMBAEI: .WORD 0          :BUS ADDRESS EXTENSION REGISTER
001406 000000 RMCS3I: .WORD 0          :CONTROL, STATUS REGISTER #3
  
```

```

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

001410

PUTBUF:

```

;REGISTER OUTPUT BUFFER
001410 000000 RMCS1O: .WORD 0          :CONTROL, STATUS REGISTER #1
001412 000000 RMWCO:  .WORD 0          :WORD COUNT REGISTER
001414 000000 RMBAO:  .WORD 0          :BUS ADDRESS REGISTER
001416 000000 RMDAO:  .WORD 0          :DISK ADDRESS REGISTER
001420 000000 RMCS2O: .WORD 0          :CONTROL, STATUS REGISTER #2
001422 000000 RMDSO:  .WORD 0          :DRIVE STATUS REGISTER
001424 000000 RMER1O: .WORD 0          :ERROR REGISTER #1
001426 000000 RMASO:  .WORD 0          :ATTENTION SUMMARY REGISTER
001430 000000 RMLAO:  .WORD 0          :LOOK AHEAD REGISTER
001432 000000 RMDBO:  .WORD 0          :DATA BUFFER
001434 000000 RMMR1O: .WORD 0          :MAINTENANCE REGISTER #1
001436 000000 RMDTO:  .WORD 0          :DRIVE TYPE REGISTER
  
```

001440	000000	RMSNO: .WORD	0	;SERIAL NUMBER REGISTER
001442	000000	RMFO: .WORD	0	;OFFSET REGISTER
001444	000000	RMDCO: .WORD	0	;DESIRED CYLINDER REGISTER
001446	000000	RMHRO: .WORD	0	;HOLDING REGISTER
001450	000000	RMMR20: .WORD	0	;MAINTENANCE REGISTER #2
001452	000000	RMER20: .WORD	0	;ERROR REGISTER #2
001454	000000	RMEC10: .WORD	0	;ECC POSITION REGISTER
001456	000000	RMEC20: .WORD	0	;ECC PATTERN REGISTER
001460	000000	RMBAE0: .WORD	0	;BUS ADDRESS EXTENSION REGISTER
001462	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.

001464	000000	TSTQUE: .WORD	0	;CONTAINS DEVICE POINTER
		.BLKW	8.	;TEST QUE FOR DEVICES UNDER TEST
001506	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	ASNDC: .WORD	0	;ASSIGNED DESIRED CYLINDER
001514	000000	ASNDA: .WORD	0	;ASSIGNED TRACK, AND SECTOR
001516	000000	CLKADR: .WORD	0	;UNIBUS ADDRESS OF KW11 CLOCK
001520	000000	CLKVCT: .WORD	0	;VECTOR ADDRESS OF KW11 CLOCK

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

001522		GETINX: .BLKB	23.	;GET INDEX TABLE
--------	--	---------------	-----	------------------

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

001551		PUTINX: .BLKB	23.	;PUT INDEX TABLE
--------	--	---------------	-----	------------------

;PUT TAGS HERE

0

.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

001600

\$ERRTB:

1
2
3
4

:ERROR 1 WRONG UNIT SELECTED

001600 072254
001602 076346
001604 076472
001606 076562

EMT1
EHT1
EDT1
EFT1

5
6
7

:ERROR 2 DEVICE WENT UNAVAILABLE

001610 072260
001612 076346
001614 076472
001616 076562

EMT2
EHT1
EDT1
EFT1

8
9
10

:ERROR 3 DEVICE WENT NONEXISTENT

001620 072266
001622 076346
001624 076472
001626 076562

EMT3
EHT1
EDT1
EFT1

11
12
13

:ERROR 4 CONTROLLER NOT READY

001630 072274
001632 076346
001634 076472
001636 076562

EMT4
EHT1
EDT1
EFT1

14
15
16

:ERROR 5 DRIVE NOT READY AND GO NOT RESET

001640 072302
001642 076346
001644 076472
001646 076562

EMT5
EHT1
EDT1
EFT1

17

ERROR POINTER TABLE

18				
19		:ERROR 6	UNEXPECTED VALUE FOR 'ATA' STATUS	
	001650	072310	EMT6	
	001652	076346	EHT1	
	001654	076472	EDT1	
	001656	076562	EFT1	
20				
21		:ERROR 7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER	
22				
	001660	072316	EMT7	
	001662	000000	0	
	001664	000000	0	
	001666	000000	0	
23				
24		:ERROR 10	DRIVE NOT READY BUT GO IS RESET	
25				
	001670	072324	EMT10	
	001672	076346	EHT1	
	001674	076472	EDT1	
	001676	076562	EFT1	
26				
27		:ERROR 11	GO NOT RESET BUT DRIVE IS READY	
28				
	001700	072330	EMT11	
	001702	076346	EHT1	
	001704	076472	EDT1	
	001706	076562	EFT1	
29				
30		:ERROR 12	INCORRECT FUNCTION CODE	
31				
	001710	072334	EMT12	
	001712	076346	EHT1	
	001714	076472	EDT1	
	001716	076562	EFT1	
32				
33		:ERROR 13	PARITY ERROR READING REMOTE REGISTERS	
34				
	001720	072342	EMT13	
	001722	076346	EHT1	
	001724	076472	EDT1	
	001726	076562	EFT1	
35				
36		:ERROR 14	TRANSFER ERROR IS INCORRECT	
37				
	001730	072354	EMT14	
	001732	076346	EHT1	
	001734	076472	EDT1	
	001736	076562	EFT1	
38				
39		:ERROR 15	INCORRECT WORD COUNT	

40	001740 072362	EMT15	
	001742 076346	EHT1	
	001744 076472	EDT1	
	001746 076562	EFT1	
41			
42		:ERROR 16	INCORRECT BUS ADDRESS
43			
	001750 072370	EMT16	
	001752 076346	EHT1	
	001754 076472	EDT1	
	001756 076562	EFT1	
44			
45		:ERROR 17	INCORRECT LBT STATUS
46			
	001760 072400	EMT17	
	001762 076346	EHT1	
	001764 076472	EDT1	
	001766 076562	EFT1	
47			
48		:ERROR 20	INCORRECT AOE
49			
	001770 072410	EMT20	
	001772 076346	EHT1	
	001774 076472	EDT1	
	001776 076562	EFT1	
50			
51		:ERROR 21	INCORRECT DISK ADDRESS
52			
	002000 072420	EMT21	
	002002 076346	EHT1	
	002004 076472	EDT1	
	002006 076562	EFT1	
53			
54		:ERROR 22	INCORRECT CYLINDER ADDRESS
55			
	002010 072430	EMT22	
	002012 076346	EHT1	
	002014 076472	EDT1	
	002016 076562	EFT1	
56			
57		:ERROR 23	INCORRECT WLE STATUS
58			
	002020 072440	EMT23	
	002022 076346	EHT1	
	002024 076472	EDT1	
	002026 076562	EFT1	
59			
60		:ERROR 24	INCORRECT UPE STATUS
61			

	002030	072450		EMT24
	002032	076346		EHT1
	002034	076472		EDT1
	002036	076562		EFT1
62				
63			:ERROR	25 INCORRECT WCF STATUS
64				
	002040	072460		EMT25
	002042	076346		EHT1
	002044	076472		EDT1
	002046	076562		EFT1
65				
66			:ERROR	26 INCORRECT WCE STATUS
67				
	002050	072470		EMT26
	002052	076346		EHT1
	002054	076472		EDT1
	002056	076562		EFT1
68				
69			:ERROR	27 INCORRECT MDPE STATUS
70				
	002060	072500		EMT27
	002062	076346		EHT1
	002064	076472		EDT1
	002066	076562		EFT1
71				
72			:ERROR	30 INCORRECT DCK STATUS
73				
	002070	072510		EMT30
	002072	076346		EHT1
	002074	076472		EDT1
	002076	076562		EFT1
74				
75			:ERROR	31 INCORRECT ECH STATUS
76				
	002100	072520		EMT31
	002102	076346		EHT1
	002104	076472		EDT1
	002106	076562		EFT1
77				
78			:ERROR	32 DLT SHOULD NOT BE SET
79				
	002110	072530		EMT32
	002112	076346		EHT1
	002114	076472		EDT1
	002116	076562		EFT1
80				
81			:ERROR	33 MXF SHOULD NOT BE SET
82				
	002120	072540		EMT33

	002122	076346	EHT1	
	002124	076472	EDT1	
	002126	076562	EFT1	
83				
84				:ERROR 34 DTE SHOULD NOT BE SET
85				
	002130	072550	EMT34	
	002132	076346	EHT1	
	002134	076472	EDT1	
	002136	076562	EFT1	
86				
87				:ERROR 35 INCORRECT HCRC STATUS
88				
	002140	072560	EMT35	
	002142	076346	EHT1	
	002144	076472	EDT1	
	002146	076562	EFT1	
89				
90				:ERROR 36 INCORRECT HCE STATUS
91				
	002150	072570	EMT36	
	002152	076346	EHT1	
	002154	076472	EDT1	
	002156	076562	EFT1	
92				
93				:ERROR 37 INCORRECT FER STATUS
94				
	002160	072600	EMT37	
	002162	076346	EHT1	
	002164	076472	EDT1	
	002166	076562	EFT1	
95				
96				:ERROR 40 DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
97				
	002170	072610	EMT40	
	002172	076346	EHT1	
	002174	076472	EDT1	
	002176	076562	EFT1	
98				
99				:ERROR 41 LOST 'MOL' DURING PACK ACKNOWLEDGE
100				
	002200	072616	EMT41	
	002202	076346	EHT1	
	002204	076472	EDT1	
	002206	076562	EFT1	
101				
102				:ERROR 42 UNSAFE ERROR DURING PACK ACKNOWLEDGE
103				
	002210	072626	EMT42	
	002212	076346	EHT1	

ERROR POINTER TABLE

	002214	076472	EDT1	
	002216	076562	EFT1	
104				
105				;ERROR 43 'OPI' ERROR DURING PACK ACKNOWLEDGE
106				
	002220	072640	EMT43	
	002222	076346	EHT1	
	002224	076472	EDT1	
	002226	076562	EFT1	
107				
108				;ERROR 44 'RMR' ERROR DURING PACK ACKNOWLEDGE
109				
	002230	072650	EMT44	
	002232	076346	EHT1	
	002234	076472	EDT1	
	002236	076562	EFT1	
110				
111				;ERROR 45 'ILR' ERROR DURING PACK ACKNOWLEDGE
112				
	002240	072660	EMT45	
	002242	076346	EHT1	
	002244	076472	EDT1	
	002246	076562	EFT1	
113				
114				;ERROR 46 'ILF' ERROR DURING PACK ACKNOWLEDGE
115				
	002250	072670	EMT46	
	002252	076346	EHT1	
	002254	076472	EDT1	
	002256	076562	EFT1	
116				
117				;ERROR 47 COMPOSITE ERROR STATUS IS INCORRECT
118				
	002260	072700	EMT47	
	002262	076346	EHT1	
	002264	076472	EDT1	
	002266	076562	EFT1	
119				
120				;ERROR 50 PARITY ERROR WRITING REMOTE REGISTERS
121				
	002270	072706	EMT50	
	002272	076346	EHT1	
	002274	076472	EDT1	
	002276	076562	EFT1	
122				
123				;ERROR 51 INCORRECT IAE STATUS DURING SEEK COMMAND
124				
	002300	072716	EMT51	
	002302	076346	EHT1	
	002304	076472	EDT1	

002306	076562	EFT1	
125			
126		:ERROR 52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE
127			
002310	072730	EMT52	
002312	076346	EHT1	
002314	076472	EDT1	
002316	076562	EFT1	
128			
129		:ERROR 53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE, ASSUME
130		:	ON CYLINDER LATCH DIDN'T RESET
131			
002320	072746	EMT53	
002322	076346	EHT1	
002324	076472	EDT1	
002326	076562	EFT1	
132			
133		:ERROR 54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND
134			
002330	072764	EMT54	
002332	076346	EHT1	
002334	076472	EDT1	
002336	076562	EFT1	
135			
136		:ERROR 55	DEVICE CHECK DURING SEEK COMMAND
137			
002340	072774	EMT55	
002342	076346	EHT1	
002344	076472	EDT1	
002346	076562	EFT1	
138			
139		:ERROR 56	PIP IS STILL SET AFTER SEEK - SKI IS RESET
140			
002350	073006	EMT56	
002352	076346	EHT1	
002354	076472	EDT1	
002356	076562	EFT1	
141			
142		:ERROR 57	ATA DID NOT SET DURING SEEK COMMAND
143			
002360	073024	EMT57	
002362	076346	EHT1	
002364	076472	EDT1	
002366	076562	EFT1	
144			
145		:ERROR 60	IVC ERROR DURING SEEK COMMAND - LOST
146		:	VOLUME VALID
147			
002370	073034	EMT60	
002372	076346	EHT1	

ERROR POINTER TABLE				
	002374 076472		EDT1	
	002376 076562		EFT1	
148				
149		:ERROR 61		ERRONEOUS IVC ERROR DURING SEEK COMMAND -
150		:		VOLUME VALID IS STIL SET
151				
	002400 073052		EMT61	
	002402 076346		EHT1	
	002404 076472		EDT1	
	002406 076562		EFT1	
152				
153		:ERROR 62		MOL IS ZERO, BUT OPI WAS NOT
154		:		REPORTED DURING SEEK COMMAND
155				
	002410 073072		EMT62	
	002412 076346		EHT1	
	002414 076472		EDT1	
	002416 076562		EFT1	
156				
157		:ERROR 63		UNUSED
158				
	002420 000000		0	
	002422 000000		0	
	002424 000000		0	
	002426 000000		0	
159				
160		:ERROR 64		DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
161				
	002430 073110		EMT64	
	002432 076346		EHT1	
	002434 076472		EDT1	
	002436 076562		EFT1	
162				
163		:ERROR 65		DRIVE EXECUTED A SEEK WITH ERROR SET
164				
	002440 073130		EMT65	
	002442 076346		EHT1	
	002444 076472		EDT1	
	002446 076562		EFT1	
165				
166		:ERROR 66		UNEXPECTED ERROR SET IN RMER1
167				
	002450 073150		EMT66	
	002452 076346		EHT1	
	002454 076472		EDT1	
	002456 076562		EFT1	
168				
169		:ERROR 67		UNEXPECTED ERROR SET IN RMER2
170				
	002460 073162		EMT67	

	002462 076346	EHT1	
	002464 076472	EDT1	
	002466 076562	EFT1	
171			
172			:ERROR 70 ERRONEOUS 'IAE' ERROR DURING RECALIBRATE
173			
	002470 073174	EMT70	
	002472 076346	EHT1	
	002474 076472	EDT1	
	002476 076562	EFT1	
174			
175			:ERROR 71 'ILF' ERROR DURING RECALIBRATE
176			
	002500 073204	EMT71	
	002502 076346	EHT1	
	002504 076472	EDT1	
	002506 076562	EFT1	
177			
178			:ERROR 72 'DPI' ERROR DURING RECALIBRATE DUE TO 'MOL' = 0
179			
	002510 073214	EMT72	
	002512 076346	EHT1	
	002514 076472	EDT1	
	002516 076562	EFT1	
180			
181			:ERROR 73 'DPI' ERROR DURING RECALIBRATE BECAUSE ON
182			: CYLINDER DIDNT DROP
183			
	002520 073232	EMT73	
	002522 076346	EHT1	
	002524 076472	EDT1	
	002526 076562	EFT1	
184			
185			:ERROR 74 'IVC' ERROR DURING RECALIBRATE - 'VV' = 0
186			
	002530 073250	EMT74	
	002532 076346	EHT1	
	002534 076472	EDT1	
	002536 076562	EFT1	
187			
188			:ERROR 75 ERRONEOUS 'IVC' ERROR DURING RECALIBRATE - 'VV' = 1
189			
	002540 073260	EMT75	
	002542 076346	EHT1	
	002544 076472	EDT1	
	002546 076562	EFT1	
190			
191			:ERROR 76 'SKI' ERROR DURING RECALIBRATE
192			
	002550 073300	EMT76	

ERROR POINTER TABLE			
	002552 076346		EHT1
	002554 076472		EDT1
	002556 076562		EFT1
193			
194		;ERROR 77	'DVC' OCCURRED DURING RECALIBRATE
195			
	002560 073310		EMT77
	002562 076346		EHT1
	002564 076472		EDT1
	002566 076562		EFT1
196			
197		;ERROR 100	LOST 'MOL' DURING RECALIBRATE - 'DPI' = 0
198			
	002570 073322		EMT100
	002572 076346		EHT1
	002574 076472		EDT1
	002576 076562		EFT1
199			
200		;ERROR 101	LOST 'VV' DURING RECALIBRATE - 'IVC' = 0
201			
	002600 073340		EMT101
	002602 076346		EHT1
	002604 076472		EDT1
	002606 076562		EFT1
202			
203		;ERROR 102	'ATA' DID NOT SET DURING RECALIBRATE
204			
	002610 073356		EMT102
	002612 076346		EHT1
	002614 076472		EDT1
	002616 076562		EFT1
205			
206		;ERROR 103	'DM' DID NOT RESET DURING RECALIBRATE
207			
	002620 073366		EMT103
	002622 076346		EHT1
	002624 076472		EDT1
	002626 076562		EFT1
208			
209		;ERROR 104	'PIP' IS STIL SET AFTER RECALIBRATE
210			
	002630 073400		EMT104
	002632 076346		EHT1
	002634 076472		EDT1
	002636 076562		EFT1
211			
212		;ERROR 105	UNEXPECTED 'ILR' ERROR DURING RECALIBRATE
213			
	002640 073416		EMT105
	002642 076346		EHT1

	002644 076472	EDT1	
	002646 076562	EFT1	
214			
215			
216			:ERROR 106 UNEXPECTED 'RMR' ERROR DURING RECALIBRATE
	002650 073426	EMT106	
	002652 076346	EHT1	
	002654 076472	EDT1	
	002656 076562	EFT1	
217			
218			
219			:ERROR 107 'UNS' ERROR DURING RECALIBRATE - AC POWER IS LOW
	002660 073436	EMT107	
	002662 076346	EHT1	
	002664 076472	EDT1	
	002666 076562	EFT1	
220			
221			
222			:ERROR 110 CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
	002670 073456	EMT110	
	002672 076370	EHT110	
	002674 076510	EDT110	
	002676 076600	EFT110	
223			
224			
225			:ERROR 111 NONEXISTENT DEVICE
	002700 073470	EMT111	
	002702 076374	EHT111	
	002704 076512	EDT111	
	002706 076602	EFT111	
226			
227			
228			:ERROR 112 DEVICE NOT AVAILABLE
	002710 073476	EMT112	
	002712 076374	EHT111	
	002714 076512	EDT111	
	002716 076602	EFT111	
229			
230			
231			:ERROR 113 BUS TIMEOUT-NED STATUS FAILURE
	002720 073504	EMT113	
	002722 000000	0	
	002724 000000	0	
	002726 000000	0	
232			
233			
234			:ERROR 114 DEVICE NOT AN RM05/3/2
	002730 073520	EMT114	
	002732 076400	EHT114	
	002734 076514	EDT114	

002736	076604	EFT114
235		
236		:ERROR 115 RMCS1 NOT INITIALIZED BY UNIBUS
237		
002740	073526	EMT115
002742	076346	EHT1
002744	076472	EDT1
002746	076562	EFT1
238		
239		:ERROR 116 RMBA NOT INITIALIZED BY UNIBUS
240		
002750	073536	EMT116
002752	076346	EHT1
002754	076472	EDT1
002756	076562	EFT1
241		
242		:ERROR 117 RMCS2 NOT INITIALIZED BY UNIBUS
243		
002760	073546	EMT117
002762	076346	EHT1
002764	076472	EDT1
002766	076562	EFT1
244		
245		:ERROR 120 RMER1 NOT INITIALIZED BY UNIBUS
246		
002770	073556	EMT120
002772	076346	EHT1
002774	076472	EDT1
002776	076562	EFT1
247		
248		:ERROR 121 RMAS NOT INITIALIZED BY UNIBUS
249		
003000	073566	EMT121
003002	076346	EHT1
003004	076472	EDT1
003006	076562	EFT1
250		
251		:ERROR 122 RMMR1 NOT INITIALIZED BY UNIBUS
252		
003010	073576	EMT122
003012	076346	EHT1
003014	076472	EDT1
003016	076562	EFT1
253		
254		:ERROR 123 RMDS NOT INITIALIZED BY UNIBUS
255		
003020	073606	EMT123
003022	076346	EHT1
003024	076472	EDT1
003026	076562	EFT1

256			
257			
258		:ERROR 124	RMEC2 NOT INITIALIZED BY UNIBUS
	003030	073616	EMT124
	003032	076346	EHT1
	003034	076472	EDT1
	003036	076562	EFT1
259			
260		:ERROR 125	RMWR2 NOT INITIALIZED BY UNIBUS
261			
	003040	073626	EMT125
	003042	076346	EHT1
	003044	076472	EDT1
	003046	076562	EFT1
262			
263		:ERROR 126	RMCS1 NOT CLEARED BY CONTROLLER CLEAR
264			
	003050	073636	EMT126
	003052	076346	EHT1
	003054	076472	EDT1
	003056	076562	EFT1
265			
266		:ERROR 127	RMBA NOT CLEARED BY CONTROLLER CLEAR
267			
	003060	073650	EMT127
	003062	076346	EHT1
	003064	076472	EDT1
	003066	076562	EFT1
268			
269		:ERROR 130	RMCS2 NOT CLEARED BY CONTROLLER CLEAR
270			
	003070	073662	EMT130
	003072	076346	EHT1
	003074	076472	EDT1
	003076	076562	EFT1
271			
272		:ERROR 131	RMER1 NOT CLEARED BY CONTROLLER CLEAR
273			
	003100	073674	EMT131
	003102	076346	EHT1
	003104	076472	EDT1
	003106	076562	EFT1
274			
275		:ERROR 132	RMAS NOT CLEARED BY CONTROLLER CLEAR
276			
	003110	073706	EMT132
	003112	076346	EHT1
	003114	076472	EDT1
	003116	076562	EFT1

277			
278			
279			:ERROR 133 RMMR1 NOT CLEARED BY CONTROLLER CLEAR
	003120	073720	EMT133
	003122	076346	EHT1
	003124	076472	EDT1
	003126	076562	EFT1
280			
281			
282			:ERROR 134 RMDS NOT CLEARED BY CONTROLLER CLEAR
	003130	073732	EMT134
	003132	076346	EHT1
	003134	076472	EDT1
	003136	076562	EFT1
283			
284			
285			:ERROR 135 RMEC2 NOT CLEARED BY CONTROLLER CLEAR
	003140	073744	EMT135
	003142	076346	EHT1
	003144	076472	EDT1
	003146	076562	EFT1
286			
287			
288			:ERROR 136 RMMR2 NOT CLEARED BY CONTROLLER CLEAR
	003150	073756	EMT136
	003152	076346	EHT1
	003154	076472	EDT1
	003156	076562	EFT1
289			
290			
291			:ERROR 137 RMCS1 NOT CLEARED BY ERROR CLEAR
	003160	073770	EMT137
	003162	076346	EHT1
	003164	076472	EDT1
	003166	076562	EFT1
292			
293			
294			:ERROR 140 RMCS2 NOT CLEARED BY ERROR CLEAR
	003170	074000	EMT140
	003172	076346	EHT1
	003174	076472	EDT1
	003176	076562	EFT1
295			
296			
297			:ERROR 141 RMCS1 NOT CLEARED BY DRIVE CLEAR
	003200	074010	EMT141
	003202	076346	EHT1
	003204	076472	EDT1
	003206	076562	EFT1
298			

299				
300			;ERROR 142	RMDS NOT CLEARED BY DRIVE CLEAR
	003210	074020		EMT142
	003212	076346		EHT1
	003214	076472		EDT1
	003216	076562		EFT1
301				
302			;ERROR 143	RMER1 NOT CLEARED BY DRIVE CLEAR
303				
	003220	074030		EMT143
	003222	076346		EHT1
	003224	076472		EDT1
	003226	076562		EFT1
304				
305			;ERROR 144	RMAS NOT CLEARED BY DRIVE CLEAR
306				
	003230	074040		EMT144
	003232	076346		EHT1
	003234	076472		EDT1
	003236	076562		EFT1
307				
308			;ERROR 145	RMMR1 NOT CLEARED BY DRIVE CLEAR
309				
	003240	074050		EMT145
	003242	076346		EHT1
	003244	076472		EDT1
	003246	076562		EFT1
310				
311			;ERROR 146	RMMR2 NOT CLEARED BY DRIVE CLEAR
312				
	003250	074060		EMT146
	003252	076346		EHT1
	003254	076472		EDT1
	003256	076562		EFT1
313				
314			;ERROR 147	RMER2 NOT CLEARED BY DRIVE CLEAR
315				
	003260	074070		EMT147
	003262	076346		EHT1
	003264	076472		EDT1
	003266	076562		EFT1
316				
317			;ERROR 150	RMEC2 NOT CLEARED BY DRIVE CLEAR
318				
	003270	074100		EMT150
	003272	076346		EHT1
	003274	076472		EDT1
	003276	076562		EFT1
319				
320			;ERROR 151	MEDIUM NOT ON LINE

321	003300 074110	EMT151	
	003302 076346	EHT1	
	003304 076472	EDT1	
	003306 076562	EFT1	
322			
323			:ERROR 152 DRIVE FAULT
324	003310 074122	EMT152	
	003312 076346	EHT1	
	003314 076472	EDT1	
	003316 076562	EFT1	
325			
326			:ERROR 153 UNSAFE SHOULD BE SET BECAUSE DVC IS SET
327	003320 074134	EMT153	
	003322 076346	EHT1	
	003324 076472	EDT1	
	003326 076562	EFT1	
328			
329			:ERROR 154 UNSAFE SHOULD NOT BE SET, AC IS LOW
330	003330 074152	EMT154	
	003332 076346	EHT1	
	003334 076472	EDT1	
	003336 076562	EFT1	
331			
332			:ERROR 155 VOLUME VALID NOT SET BY PACK ACK
333	003340 074170	EMT155	
	003342 076346	EHT1	
	003344 076472	EDT1	
	003346 076562	EFT1	
334			
335			:ERROR 156 OFFSET MODE NOT SET BY OFFSET COMMAND
336	003350 074202	EMT156	
	003352 076346	EHT1	
	003354 076472	EDT1	
	003356 076562	EFT1	
337			
338			:ERROR 157 OFFSET MODE NOT RESET BY RTC COMMAND
339	003360 074214	EMT157	
	003362 076346	EHT1	
	003364 076472	EDT1	
	003366 076562	EFT1	
340			
341			:ERROR 160 RMOF NOT RESET BY RIP COMMAND
342			

003370	074226	EMT160
003372	076346	EHT1
003374	076472	EDT1
003376	076562	EFT1
343		
344		
345		:ERROR 161 RMDA NOT RESET BY RIP COMMAND
003400	074236	EMT161
003402	076346	EHT1
003404	076472	EDT1
003406	076562	EFT1
346		
347		
348		:ERROR 162 RMDC NOT RESET BY RIP COMMAND
003410	074250	EMT162
003412	076346	EHT1
003414	076472	EDT1
003416	076562	EFT1
349		
350		
351		:ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
352		: WRITE BUFFER
003420	076142	EMT336
003422	076430	EHT336
003424	076526	EDT336
003426	076616	EFT336
353		
354		
355		:ERROR 164 OPI SHOULD NOT BE SET
003430	074272	EMT164
003432	076346	EHT1
003434	076472	EDT1
003436	076562	EFT1
356		
357		
358		:ERROR 165 IVC SHOULD NOT BE SET
003440	074300	EMT165
003442	076346	EHT1
003444	076472	EDT1
003446	076562	EFT1
359		
360		
361		:ERROR 166 IAE SHOULD NOT BE SET
003450	074306	EMT166
003452	076346	EHT1
003454	076472	EDT1
003456	076562	EFT1
362		
363		
364		:ERROR 167 NEM SHOULD NOT BE SET

003460	074314	EMT167
003462	076346	EHT1
003464	076472	EDT1
003466	076562	EFT1
365		
366	:ERROR 170	UNUSED
367		
003470	000000	0
003472	000000	0
003474	000000	0
003476	000000	0
368		
369	:ERROR 171	'ATA' NOT SET DURING RETURN TO CENTERLINE
370		
003500	074332	EMT171
003502	076346	EHT1
003504	076472	EDT1
003506	076562	EFT1
371		
372	:ERROR 172	'ATA' NOT SET BY OFFSET COMMAND
373		
003510	074342	EMT172
003512	076346	EHT1
003514	076472	EDT1
003516	076562	EFT1
374		
375	:ERROR 173	RMER2 NOT INITIALIZED BY UNIBUS INIT
376		
003520	074352	EMT173
003522	076346	EHT1
003524	076472	EDT1
003526	076562	EFT1
377		
378	:ERROR 174	RMER2 NOT INITIALIZED BY CONTROLLER CLEAR
379		
003530	074362	EMT174
003532	076346	EHT1
003534	076472	EDT1
003536	076562	EFT1
380		
381	:ERROR 175	SELECTED DEVICE IS IN WRITE PROTECT
382		
003540	074374	EMT175
003542	076346	EHT1
003544	076472	EDT1
003546	076562	EFT1
383		
384	:ERROR 176	CANNOT SET DIAGNOSTIC MODE
385		
003550	074402	EMT176

Line	Address	Code	Message
	003552	076346	EHT1
	003554	076472	EDT1
	003556	076562	EFT1
386			
387			
388			:ERROR 177 INCORRECT 'MOL' STATUS DURING DIAGNOSTIC MODE
	003560	074410	EMT177
	003562	076346	EHT1
	003564	076472	EDT1
	003566	076562	EFT1
389			
390			
391			:ERROR 200 INCORRECT 'PIP' STATUS DURING DIAGNOSTIC MODE
	003570	074422	EMT200
	003572	076346	EHT1
	003574	076472	EDT1
	003576	076562	EFT1
392			
393			
394			:ERROR 201 INCORRECT 'WRL' STATUS DURING DIAGNOSTIC MODE
	003600	074434	EMT201
	003602	076346	EHT1
	003604	076472	EDT1
	003606	076562	EFT1
395			
396			
397			:ERROR 202 INCORRECT 'SKI' STATUS DURING DIAGNOSTIC MODE
	003610	074446	EMT202
	003612	076346	EHT1
	003614	076472	EDT1
	003616	076562	EFT1
398			
399			
400			:ERROR 203 INCORRECT 'DVC' STATUS DURING DIAGNOSTIC MODE
	003620	074460	EMT203
	003622	076346	EHT1
	003624	076472	EDT1
	003626	076562	EFT1
401			
402			
403			:ERROR 204 'VV' WAS NOT RESET BY MAINTENANCE UNIT READY
	003630	074472	EMT204
	003632	076346	EHT1
	003634	076472	EDT1
	003636	076562	EFT1
404			
405			
406			:ERROR 205 SELECTED DEVICE HAS A PERSISTENT 'SKI' ERROR
	003640	074510	EMT205
	003642	076346	EHT1

003644	076472	EDT1	
003646	076562	EFT1	
407			
408		:ERROR	206 'LBC' DID NOT SET DURING DIAGNOSTIC MODE
409			
003650	074520	EMT206	
003652	076346	EHT1	
003654	076472	EDT1	
003656	076562	EFT1	
410			
411		:ERROR	207 UNEXPECTED LOSS OF 'MOL' - MEDIUM IS OFF LINE
412			
003660	074530	EMT207	
003662	076346	EHT1	
003664	076472	EDT1	
003666	076562	EFT1	
413			
414		:ERROR	210 UNEXPECTED LOSS OF VOLUME VALID - 'VV' = 0
415			
003670	074542	EMT210	
003672	076346	EHT1	
003674	076472	EDT1	
003676	076562	EFT1	
416			
417		:ERROR	211 UNEXPECTED MECHANICAL MOTION - 'PIP' = 1
418			
003700	074550	EMT211	
003702	076346	EHT1	
003704	076472	EDT1	
003706	076562	EFT1	
419			
420		:ERROR	212 UNEXPECTED DEVICE FAULT - 'DVC' = 1
421			
003710	074564	EMT212	
003712	076346	EHT1	
003714	076472	EDT1	
003716	076562	EFT1	
422			
423		:ERROR	213 UNEXPECTED SEEK INCOMPLETE ERROR - 'SKI' = 1
424			
003720	074600	EMT213	
003722	076346	EHT1	
003724	076472	EDT1	
003726	076562	EFT1	
425			
426		:ERROR	214 DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
427			
003730	074610	EMT214	
003732	076360	EHT2	
003734	076502	EDT2	

003736	076572	EFT2	
428			
429			
430			
003740	074630	EMT215	
003742	076360	EHT2	
003744	076502	EDT2	
003746	076572	EFT2	
431			
432			
433			
003750	074642	EMT216	
003752	076346	EHT1	
003754	076472	EDT1	
003756	076562	EFT1	
434			
435			
436			
003760	074652	EMT217	
003762	076346	EHT1	
003764	076472	EDT1	
003766	076562	EFT1	
437			
438			
439			
003770	074662	EMT220	
003772	076346	EHT1	
003774	076472	EDT1	
003776	076562	EFT1	
440			
441			
442			
004000	074672	EMT221	
004002	076346	EHT1	
004004	076472	EDT1	
004006	076562	EFT1	
443			
444			
445			
004010	074702	EMT222	
004012	076346	EHT1	
004014	076472	EDT1	
004016	076562	EFT1	
446			
447			
448			
004020	074712	EMT223	
004022	076404	EHT223	
004024	076516	EDT223	
004026	076606	EFT223	

449				
450				
451			:ERROR 224	UNUSED
	004030	000000	0	
	004032	000000	0	
	004034	000000	0	
	004036	000000	0	
452				
453			:ERROR 225	UNUSED
454				
	004040	000000	0	
	004042	000000	0	
	004044	000000	0	
	004046	000000	0	
455				
456			:ERROR 226	UNUSED
457				
	004050	000000	0	
	004052	000000	0	
	004054	000000	0	
	004056	000000	0	
458				
459			:ERROR 227	UNUSED
460				
	004060	000000	0	
	004062	000000	0	
	004064	000000	0	
	004066	000000	0	
461				
462			:ERROR 230	UNUSED
463				
	004070	000000	0	
	004072	000000	0	
	004074	000000	0	
	004076	000000	0	
464				
465			:ERROR 231	UNUSED
466				
	004100	000000	0	
	004102	000000	0	
	004104	000000	0	
	004106	000000	0	
467				
468			:ERROR 232	UNUSED
469				
	004110	000000	0	
	004112	000000	0	
	004114	000000	0	
	004116	000000	0	

470				
471			;ERROR 233	UNUSED
472	004120	000000	0	
	004122	000000	0	
	004124	000000	0	
	004126	000000	0	
473				
474			;ERROR 234	UNUSED
475	004130	000000	0	
	004132	000000	0	
	004134	000000	0	
	004136	000000	0	
476				
477			;ERROR 235	UNUSED
478	004140	000000	0	
	004142	000000	0	
	004144	000000	0	
	004146	000000	0	
479				
480			;ERROR 236	UNUSED
481	004150	000000	0	
	004152	000000	0	
	004154	000000	0	
	004156	000000	0	
482				
483			;ERROR 237	UNUSED
484	004160	000000	0	
	004162	000000	0	
	004164	000000	0	
	004166	000000	0	
485				
486			;ERROR 240	UNUSED
487	004170	000000	0	
	004172	000000	0	
	004174	000000	0	
	004176	000000	0	
488				
489			;ERROR 241	UNUSED
490	004200	000000	0	
	004202	000000	0	
	004204	000000	0	
	004206	000000	0	
491				

492				
493			:ERROR 242	UNUSED
	004210	000000	0	
	004212	000000	0	
	004214	000000	0	
	004216	000000	0	
494				
495			:ERROR 243	UNUSED
496				
	004220	000000	0	
	004222	000000	0	
	004224	000000	0	
	004226	000000	0	
497				
498			:ERROR 244	UNUSED
499				
	004230	000000	0	
	004232	000000	0	
	004234	000000	0	
	004236	000000	0	
500				
501			:ERROR 245	UNUSED
502				
	004240	000000	0	
	004242	000000	0	
	004244	000000	0	
	004246	000000	0	
503				
504			:ERROR 246	'ATA' NOT RESET BY GO WHEN 'ERR' = 0
505				
	004250	074766	EMT246	
	004252	076346	EHT1	
	004254	076472	EDT1	
	004256	076562	EFT1	
506				
507			:ERROR 247	'ATA' NOT RESET BY WRITING RMAS
508				
	004260	074776	EMT247	
	004262	076346	EHT1	
	004264	076472	EDT1	
	004266	076562	EFT1	
509				
510			:ERROR 250	'ATA' WAS RESET BY GO WHEN 'ERR' = 1
511				
	004270	075010	EMT250	
	004272	076346	EHT1	
	004274	076472	EDT1	
	004276	076562	EFT1	
512				
513			:ERROR 251	PROGRAM INTERRUPT WAS NOT GENERATED

514	004300	075024	EMT251
	004302	076360	EHT2
	004304	076502	EDT2
	004306	076572	EFT2
515			
516			:ERROR 252 PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
517			
	004310	075032	EMT252
	004312	076360	EHT2
	004314	076502	EDT2
	004316	076572	EFT2
518			
519			:ERROR 253 OFFSET MODE WAS NOT RESET BY WRITING RMDC
520			
	004320	075040	EMT253
	004322	076346	EHT1
	004324	076472	EDT1
	004326	076562	EFT1
521			
522			:ERROR 254 INCORRECT 'ILF' STATUS
523			
	004330	075056	EMT254
	004332	076346	EHT1
	004334	076472	EDT1
	004336	076562	EFT1
524			
525			:ERROR 255 INCORRECT 'ATA' STATUS
526			
	004340	075066	EMT255
	004342	076346	EHT1
	004344	076472	EDT1
	004346	076562	EFT1
527			
528			:ERROR 256 INCORRECT 'ILR' STATUS
529			
	004350	075076	EMT256
	004352	076416	EHT256
	004354	076516	EDT223
	004356	076606	EFT223
530			
531			:ERROR 257 INVALID IAE STATUS DURING SEARCH COMMAND
532			
	004360	075106	EMT257
	004362	076346	EHT1
	004364	076472	EDT1
	004366	076562	EFT1
533			
534			:ERROR 260 'IVC' WAS NOT DETECTED DURING SEARCH COMMAND
535			

004370	075120	EMT260
004372	076346	EHT1
004374	076472	EDT1
004376	076562	EFT1
536		
537		
538		:ERROR 261 DRIVE EXECUTED SEARCH WITH ERROR SET
004400	075132	EMT261
004402	076346	EHT1
004404	076472	EDT1
004406	076562	EFT1
539		
540		
541		:ERROR 262 'LBC' ERROR NOT SET DURING DIAGNOSTIC MODE
004410	075152	EMT262
004412	076346	EHT1
004414	076472	EDT1
004416	076562	EFT1
542		
543		
544		:ERROR 263 'SKI' ERROR DURING SEARCH COMMAND
004420	075162	EMT263
004422	076346	EHT1
004424	076472	EDT1
004426	076562	EFT1
545		
546		
547		:ERROR 264 'IVC' ERROR DURING SEARCH - LOST VOLUME VALID
004430	075172	EMT264
004432	076346	EHT1
004434	076472	EDT1
004436	076562	EFT1
548		
549		
550		:ERROR 265 ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
004440	075212	EMT265
004442	076346	EHT1
004444	076472	EDT1
004446	076562	EFT1
551		
552		
553		:ERROR 266 DEVICE FAULT (DVC) DURING SEARCH
004450	075232	EMT266
004452	076346	EHT1
004454	076472	EDT1
004456	076562	EFT1
554		
555		
556		:ERROR 267 SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER
557		: ADDRESS IS TOO LARGE

004460	075244	EMT267
004462	076346	EHT1
004464	076472	EDT1
004466	076562	EFT1
558		
559	:ERROR	270 OPI ERROR DURING SEARCH BECAUSE MOL = 0
560		
004470	075262	EMT270
004472	076346	EHT1
004474	076472	EDT1
004476	076562	EFT1
561		
562	:ERROR	271 OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
563	:	DIDN'T DROP
564		
004500	075276	EMT271
004502	076346	EHT1
004504	076472	EDT1
004506	076562	EFT1
565		
566	:ERROR	272 LOST MOL DURING SEARCH, OPI IS NOT SET
567		
004510	075314	EMT272
004512	076346	EHT1
004514	076472	EDT1
004516	076562	EFT1
568		
569	:ERROR	273 PIP STIL SET AFTER SEARCH
570		
004520	075332	EMT273
004522	076346	EHT1
004524	076472	EDT1
004526	076562	EFT1
571		
572	:ERROR	274 PARITY ERROR OCCURRED WHILE WRITING REMOTE
573	:	REGISTERS BUT MXF DID NOT SET
574		
004530	075350	EMT274
004532	076346	EHT1
004534	076472	EDT1
004536	076562	EFT1
575		
576	:ERROR	275 MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
577	:	COMMAND STARTED
578		
004540	075366	EMT275
004542	076346	EHT1
004544	076472	EDT1
004546	076562	EFT1
579		

ERROR POINTER TABLE

580		:ERROR 276	'OPI' ERROR DURING DATA TRANSFER BECAUSE 'MOL' WAS
581		:	ZERO
582			
	004550 075400	EMT276	
	004552 076346	EHT1	
	004554 076472	EDT1	
	004556 076562	EFT1	
583			
584		:ERROR 277	'OPI' ERROR DURING DATA TRANSFER BECAUSE 1) ON
585		:	CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
586		:	3) RUN TIMED OUT
587			
	004560 075414	EMT277	
	004562 076346	EHT1	
	004564 076472	EDT1	
	004566 076562	EFT1	
588			
589		:ERROR 300	'IVC' ERROR DURING DATA TRANSFER BECAUSE VOLUME
590		:	WAS NOT VALID
591			
	004570 075432	EMT300	
	004572 076346	EHT1	
	004574 076472	EDT1	
	004576 076562	EFT1	
592			
593		:ERROR 301	ERRONEOUS 'IVC' ERROR DURING DATA TRANSFER - VOLUME
594		:	IS VALID
595			
	004600 075452	EMT301	
	004602 076346	EHT1	
	004604 076472	EDT1	
	004606 076562	EFT1	
596			
597		:ERROR 302	'ILR' ERROR DURING DATA TRANSFER
598			
	004610 075474	EMT302	
	004612 076346	EHT1	
	004614 076472	EDT1	
	004616 076562	EFT1	
599			
600		:ERROR 303	'ILF' ERROR DURING DATA TRANSFER
601			
	004620 075506	EMT303	
	004622 076346	EHT1	
	004624 076472	EDT1	
	004626 076562	EFT1	
602			
603		:ERROR 304	'RMR' ERROR DURING DATA TRANSFER
604			
	004630 075520	EMT304	
	004632 076346	EHT1	

ERROR POINTER TABLE

	004634 076472		EDT1
	004636 076562		EFT1
605			
606		;ERROR 305	INCORRECT "IAE" STATUS DURING DATA TRANSFER
607			
	004640 075532		EMT305
	004642 076346		EHT1
	004644 076472		EDT1
	004646 076562		EFT1
608			
609		;ERROR 306	"SKI" ERROR DURING DATA TRANSFER
610			
	004650 075544		EMT306
	004652 076346		EHT1
	004654 076472		EDT1
	004656 076562		EFT1
611			
612		;ERROR 307	DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
613			
	004660 075554		EMT307
	004662 076346		EHT1
	004664 076472		EDT1
	004666 076562		EFT1
614			
615		;ERROR 310	DEVICE FAULT DURING DATA TRANSFER
616			
	004670 075574		EMT310
	004672 076346		EHT1
	004674 076472		EDT1
	004676 076562		EFT1
617			
618		;ERROR 311	LOSS OF BIT CLOCK DURING DATA TRANSFER
619			
	004700 075606		EMT311
	004702 076346		EHT1
	004704 076472		EDT1
	004706 076562		EFT1
620			
621		;ERROR 312	LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
622			
	004710 075620		EMT312
	004712 076346		EHT1
	004714 076472		EDT1
	004716 076562		EFT1
623			
624		;ERROR 313	UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
625			
	004720 075632		EMT313
	004722 076346		EHT1
	004724 076472		EDT1

Line	Code	Address	Label	Description
		004726 076562	EFT1	
626				
627				
628				
		004730 075652	EMT314	
		004732 076346	EHT1	
		004734 076472	EDT1	
		004736 076562	EFT1	
629				
630				
631				
		004740 075664	EMT315	
		004742 076346	EHT1	
		004744 076472	EDT1	
		004746 076562	EFT1	
632				
633				
634				
		004750 075676	EMT316	
		004752 076346	EHT1	
		004754 076472	EDT1	
		004756 076562	EFT1	
635				
636				
637				
		004760 075710	EMT317	
		004762 076346	EHT1	
		004764 076472	EDT1	
		004766 076562	EFT1	
638				
639				
640				
		004770 075720	EMT320	
		004772 076346	EHT1	
		004774 076472	EDT1	
		004776 076562	EFT1	
641				
642				
643				
		005000 075730	EMT321	
		005002 076346	EHT1	
		005004 076472	EDT1	
		005006 076562	EFT1	
644				
645				
646				
		005010 075740	EMT322	
		005012 076346	EHT1	
		005014 076472	EDT1	
		005016 076562	EFT1	

647				
648			:ERROR 323	DATA CHECK ERROR DURING DATA TRANSFER
649				
	005020	075746	EMT323	
	005022	076346	EHT1	
	005024	076472	EDT1	
	005026	076562	EFT1	
650				
651			:ERROR 324	CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
652				
	005030	075756	EMT324	
	005032	076346	EHT1	
	005034	076472	EDT1	
	005036	076562	EFT1	
653				
654			:ERROR 325	UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
655				
	005040	075770	EMT325	
	005042	076346	EHT1	
	005044	076472	EDT1	
	005046	076562	EFT1	
656				
657			:ERROR 326	DATA PARITY ERROR DURING READ COMMAND
658				
	005050	076002	EMT326	
	005052	076346	EHT1	
	005054	076472	EDT1	
	005056	076562	EFT1	
659				
660			:ERROR 327	OFFSET MODE NOT RESET BY WRITE COMMAND
661				
	005060	076020	EMT327	
	005062	076346	EHT1	
	005064	076472	EDT1	
	005066	076562	EFT1	
662				
663			:ERROR 330	DATA PARITY ERROR DURING WRITE COMMAND
664				
	005070	076032	EMT330	
	005072	076346	EHT1	
	005074	076472	EDT1	
	005076	076562	EFT1	
665				
666			:ERROR 331	WRITE CLOCK FAILURE DURING WRITE COMMAND
667				
	005100	076042	EMT331	
	005102	076346	EHT1	
	005104	076472	EDT1	
	005106	076562	EFT1	

668			
669			
670		:ERROR 332	DATA LATE ERROR DURING DATA TRANSFER
	005110	076054	EMT332
	005112	076346	EHT1
	005114	076472	EDT1
	005116	076562	EFT1
671			
672		:ERROR 333	PIP STIL SET AFTER DATA TRANSFER - SKI = 0
673			
	005120	076066	EMT333
	005122	076346	EHT1
	005124	076472	EDT1
	005126	076562	EFT1
674			
675		:ERROR 334	LOST MOL DURING DATA TRANSFER - OPI = 0
676			
	005130	076104	EMT334
	005132	076346	EHT1
	005134	076472	EDT1
	005136	076562	EFT1
677			
678		:ERROR 335	LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
679			
	005140	076122	EMT335
	005142	076346	EHT1
	005144	076472	EDT1
	005146	076562	EFT1
680			
681		:ERROR 336	DATA READ DOES NOT COMPARE WITH DATA WRITTEN
682			
	005150	076142	EMT336
	005152	076430	EHT336
	005154	076526	EDT336
	005156	076616	EFT336
683			
684		:ERROR 337	WRITE CHECK ERROR NOT DETECTED
685			
	005160	076152	EMT337
	005162	076442	EHT337
	005164	076536	EDT337
	005166	076626	EFT337
686			
687		:ERROR 340	WRITE CHECK ERROR AT UNEXPECTED ADDRESS
688			
	005170	076162	EMT340
	005172	076430	EHT336
	005174	076526	EDT336
	005176	076616	EFT336

689

690			:ERROR 341	INCORRECT DATA DURING WRITE CHECK ERROR
691	005200	076174		EMT341
	005202	076430		EHT336
	005204	076526		EDT336
	005206	076616		EFT336
692			:ERROR 342	"IVC" ERROR NOT DETECTED DURING DATA TRANSFER
693	005210	076202		EMT342
694	005212	076346		EHT1
	005214	076472		EDT1
	005216	076562		EFT1
695			:ERROR 343	"FER" NOT DETECTED DURING DATA TRANSFER
696	005220	076214		EMT343
697	005222	076346		EHT1
	005224	076472		EDT1
	005226	076562		EFT1
698			:ERROR 344	"HCE" NOT DETECTED DURING DATA TRANSFER
699	005230	076226		EMT344
700	005232	076454		EHT344
	005234	076546		EDT344
	005236	076636		EFT344
701			:ERROR 345	"BSE" NOT DETECTED DURING DATA TRANSFER
702	005240	076240		EMT345
703	005242	076346		EHT1
	005244	076472		EDT1
	005246	076562		EFT1
704			:ERROR 346	HEADER ERROR WAS DETECTED W/ HCI SET
705	005250	076250		EMT346
706	005252	076346		EHT1
	005254	076472		EDT1
	005256	076562		EFT1
707			:ERROR 347	DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
708	005260	076264		EMT347
709	005262	076346		EHT1
	005264	076472		EDT1
	005266	076562		EFT1
710			:ERROR 350	LOST VOLUME VALID DURING SEARCH - "IVC" = 0
711				

712

005270	076276	EMT350
005272	076346	EHT1
005274	076472	EDT1
005276	076562	EFT1

713

714 ;ERROR 351 'ATA' DID NOT SET DURING SEARCH

715

005300	076314	EMT351
005302	076346	EHT1
005304	076472	EDT1
005306	076562	EFT1

716

717 ;ERROR 352 PROGRAM TIMEOUT WHILE TESTING RMLA

718

005310	076324	EMT352
005312	000000	0
005314	000000	0
005316	000000	0

719

720 ;ERROR 353 LOOK AHEAD TEST FAILS

721

005320	076330	EMT353
005322	076466	EHT353
005324	076560	EDT353
005326	076646	EFT353

722

723 ;ERROR 354 BSE SHOULD NOT BE SET

724

005330	076340	EMT354
005332	076346	EHT1
005334	076472	EDT1
005336	076562	EFT1

725

726 ;PUT ERROR TABLE HERE

727

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

.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 005340 011600      BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
4 005342 005740      TST      -(R0)      ;ADJUST PC -2
5 005344 022626      CMP      (SP)+,(SP)+    ;RESTORE STACK POINTER
6 005346 104401 005354  TYPE     ,65$      ;:TYPE ASCIZ STRING
   005352 000417      BR       64$      ;:GET OVER THE ASCIZ
   65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 005412 010046      MOV      R0,-(SP)      ;SETUP FOR TYPING OUT PC
8 005414 104402      TYPOC
9 005416 000240      NOP
   ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;TO STOP ON UNEXPECTED TIMEOUT.
10
11
12      .SBTTL  START OF PROGRAM
13
14 005420 000240      START:  NOP
15 005422 005227 000000  INC      #0      ;TTY LOOP, WAIT FOR INCREMENT
16 005426 001375      BNE     .-4      ;OF WORD
17 005430 000005      RESET    ;RESET THE WORLD
18
19      .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     SENDCT,SEOPCT  ;;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     #,$SLPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
   MOV     #,$SLPERR      ;;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
   64$:  MOV     #65$, (SP) ;;SET UP FOR TRAP RETURN
   65$:  MOV     #SWREG,SWR  ;;POINT TO SOFTWARE SWR
   MOV     #DISPREG,DISPLAY
    
```



```

005654 012637 000004      66$:  MOV    (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
005660 005037 001230      CLR    $PASS                ;;CLEAR PASS COUNT
005664 132737 000200 001243  BITB   #APTSIZE,$ENVM      ;;TEST USER SIZE UNDER APT
005672 001403      BEQ    67$                  ;;YES,USE NON-APT SWITCH
005674 012737 001244 001154  MOV    #$$SWREG,$SWR      ;;NO,USE APT SWITCH REGISTER
005702
20      67$:
21      ;SETUP 'TIMEOUT' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
22      MOV    #BADTMO,ERRVEC ;;SETUP FOR UNEXPECTED TIMEOUT
23      MOV    #PR6,ERRVEC+2  ;;LEVEL 6
005716 012746 000300 000006  MOV    #PR6,-(SP)         ;;PUT NEW PS ON STACK
005722 012746 005730      MOV    #68$,-(SP)        ;;PUT NEW PC ON STACK
005726 000002      RTI                      ;;POP NEW PC AND PS
005730
24      68$:
25      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005730 005227 177777      INC    #-1                ;;FIRST TIME?
005734 001056      BNE    69$                ;;BRANCH IF NO
005736 022737 037650 000042  CMP    #SENDAD,@#42      ;;ACT-11?
005744 001452      BEQ    69$                ;;BRANCH IF YES
005746 104401 006014      TYPE   ,70$              ;;TYPE ASCIZ STRING
      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
005752 005737 000042      TST    @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
005756 001012      BNE    71$                ;;BRANCH IF YES
005760 123727 001242 000001  CMPB   $ENV,#1           ;;ARE WE RUNNING UNDER APT?
005766 001406      BEQ    71$                ;;BRANCH IF YES
005770 023727 001154 000176  CMP    $SWR,#SWREG       ;;SOFTWARE SWITCH REG SELECTED?
005776 001005      BNE    72$                ;;BRANCH IF NO
006000 104407      GTSWR                    ;;GET SOFT-SWR SETTINGS
006002 000403      BR    72$
006004 112737 000001 001150 71$:  MOVB   #1,$AUTOB         ;;SET AUTO-MODE INDICATOR
006012 000427      72$:
006012      BR    69$              ;;GET OVER THE ASCIZ
006072      ;;70$: .ASCIZ <CRLF>@CZRMOAO - RM05/3/2 FUNCTIONAL TEST, PART 3a<CRLF>
      69$:
26
27      ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
28      ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
29
30      CLR    XXDP            ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
31      CMPB   #16,@#41        ;;LOADED FROM AN RM05/3/2 ?
32      BNE    3$              ;;BRANCH IF NOT
33      MOV    @#40,XXDP       ;;GET DEVICE INDICATOR AND NUMBER
34      CMPB   #7,XXDP        ;;IS IT A VALID NUMBER ?
35      BHIS   1$              ;;YES
36      CLRB   XXDP           ;;NO, DEFAULT TO DRIVE 0
37      TST    @#42           ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
38      BEQ    2$              ;;BR IF NEITHER
39      TYPE   ,74$          ;;TYPE ASCIZ STRING
006142 000412      BR    73$              ;;GET OVER THE ASCIZ
      ;;74$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
      73$:
40      CLR    -(SP)          ;;CLEAR WORD ON STACK
41      MOVB   XXDP,(SP)      ;;GET DRIVE ADDRESS
42      TYPOS                    ;;TYPE THE ADDRESS
43      .BYTE  1              ;;ONLY 1 CHARACTER

```

```

44 006201 000 .BYTE 0 ;SUPRESS LEADING ZEROS
45 006202 104401 001217 TYPE $CRLF ;CR-LF
46 006206 000517 BR 3$ ;GET NUMBER OF DRIVES
47
48 006210 005227 177777 2$: INC #-1 ;FIRST TIME THRU HERE ?
49 006214 001114 BNE 3$ ;NO
50 006216 104401 006224 TYPE 76$ ;:TYPE ASCIZ STRING
006222 000410 BR 75$ ;:GET OVER THE ASCIZ
;:76$: .ASCIZ <CRLF>/TO TEST DRIVE /
;:75$:
51 006244 005046 CLR -(SP) ;CLEAR WORD ON STACK
52 006246 113716 001330 MOVB XXDP,(SP) ;GET DRIVE ADDRESS
53 006252 104403 TYPOS ;TYPE DRIVE ADDRESS
54 006254 001 .BYTE 1 ;ONLY 1 CHARACTER
55 006255 000 .BYTE 0 ;SUPRESS LEADING ZEROS
56 006256 104401 006264 TYPE 78$ ;:TYPE ASCIZ STRING
006262 000431 BR 77$ ;:GET OVER THE ASCIZ
;:78$: .ASCIZ /, HALT PROGRAM, REMOVE RRD PCK AND REPLACE IT/<CRLF>
;:77$:
57 006346 104401 006354 TYPE 79$ ;:TYPE ASCIZ STRING
006352 000435 BR 3$ ;:GET OVER THE ASCIZ
;:79$: .ASCIZ /WITH A WORK PCK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
;:3$:
006446
61 ;CHECK FOR AUTO MODE OR STANDLONE MODE
62 006446 105737 001150 TSTB $AUTOB ;RUNNING IN AUTO MODE ?
63 006452 001515 BEQ STANDALONE ;BR IF NO
64 006454 012737 000377 001300 MOV #377,$DEV M ;SET DEVICE MAP FOR ALL DRIVES
65
66 ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
67 006462 XSIZ:
68 006462 132737 000200 001243 BITB #BIT7,$ENVM ;SIZING ALLOWED ?
69 006470 001075 BNE 7$ ;NO
70
71 006472 005001 CLR R1 ;START FROM DRIVE 0
72 006474 013700 001276 MOV $BASE,R0 ;LOAD THE BASE ADDRESS
73
74 006500 136137 071616 001300 1$: BITB ATNTBL(R1),$DEV M ;IS DEVICE PRESENT IN MAP ?
75 006506 001462 BEQ 6$ ;BR IF NO
76 006510 012737 071416 006652 MOV #LODEV,5$ ;GET ADDRESS OF LOAD DEVICE MESSAGE
77 006516 005737 001330 TST XXDP ;LOADED FROM RM05/3/2 ?
78 006522 001403 BEQ 2$ ;NO
79 006524 123701 001330 CMPB XXDP,R1 ;IS THIS THE DRIVE ?
80 006530 001435 BEQ 3$ ;YES, TRY NEXT DRIVE
81
82 006532 012760 000040 000010 2$: MOV #CLR, RMCS2(R0) ;CLEAR MASS BUS
83 006540 010160 000010 MOV R1, RMCS2(R0) ;LOAD THE DRIVE ADDRESS
84 006544 005760 000012 TST RMD5(R0) ;TRY TO ACCESS AN RM DRIVE REGISTER
85 006550 012737 071436 006652 MOV #NOTPRS,5$ ;GET ADDRESS OF NOT PRESENT MESSAGE
86 006556 032760 010000 000010 BIT #NED, RMCS2(R0) ;IS DRIVE PRESENT ?
87 006564 001017 BNE 3$ ;BR IF NO
88 006566 012737 071453 006652 MOV #NOTAVL,5$ ;GET ADDRESS OF AVAILABLE MESSAGE
89 006574 032760 004000 000000 BIT #DVA, RMCS1(R0) ;IS DRIVE AVAILABLE ?
90 006602 001410 BEQ 3$ ;BR IF NO
91 006604 012737 071472 006652 MOV #OFFLIN,5$ ;GET ADDRESS OF OFF LINE MESSAGE
92 006612 032760 010000 000012 BIT #MOL, RMD5(R0) ;IS MEDIUM ON LINE ?
93 006620 001401 BEQ 3$ ;BR IF NO
94 006622 000414 BR 6$

```

```

95
96 006624 146137 071616 001300 3$: BICB ATNTBL(R1), $DEVM ;CLEAR DEVICE FROM BIT MAP
97 006632 104401 001217 4$: TYPE , $CRLF ;CR-LF
98 006636 104401 071410 TYPE ,MSGDRV ;TYPE 'DRIVE'
99 006642 010146 MOV R1, -(SP) ;SAVE R1 FOR TYPEOUT
    006644 104403 TYPOS ;GO TYPE--OCTAL ASCII
    006646 002 .BYTE 2 ;TYPE 2 DIGIT(S)
    006647 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
100 006650 104401 TYPE ;TYPE ERROR MESSAGE
101 006652 000000 5$: .WORD 0 ;ADDRESS OF MESSAGE GOES HERE
102
103 006654 005201 6$: INC R1 ;INCREMENT THE DRIVE ADDRESS
104 006656 020127 000007 CMP R1, #7 ;ALL DRIVES ARE CHECKED ?
105 006662 003706 BLE 1$ ;BRANCH IF NOT
106
107 006664 104401 001217 7$: TYPE , $CRLF ;CR-LF
108 006670 005004 CLR R4 ;THESE TWO LOOPS ARE ADDED TO
109 006672 005304 DEC R4 ;WAIT FOR TTY TO FINISH TYPING.
110 006674 001376 BNE .-2
111 006676 005304 DEC R4
112 006700 001376 BNE .-2
113
114 006702 000137 007612 JMP CMNSTART ;JUMP TO COMMON START
115
  
```

```
1          .SBTTL  STANDALONE INPUT ROUTINES
2
3          STANDALONE:
4 006706   004737   066146      JSR      PC,$TKINT      ;INITIALIZE CONSOLE
5
6 006712   005227   177777      INC      #-1           ;FIRST TIME THRU HERE ?
7 006716   001426                BEQ      3$           ;YES !!
8
9          ;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
10         1$:
11 006720   104401   070755      TYPE     ,CNSL00      ;MAINTAIN PREVIOUS PARAMETERS ?
12 006724   104411                RDCHR                ;GET RESPONSE
13 006726   012637   001176      MOV      (SP)+,$TMP1  ;ECHO RESPONSE
14 006732   104401   001176      TYPE     ,TMP1
15 006736   123727   001176   000131  CMPB    $TMP1,#'Y    ;YES RESPONSE ?
16 006744   001004                BNE     2$           ;NO!!
17 006746   104401   001217      TYPE     ,SCRLF      ;CR-LF
18 006752   000137   007612      JMP     CMNSTART     ;KEEP PREVIOUS PARAMETERS
19
20 006756   123727   001176   000116  2$:    CMPB    $TMP1,#'N    ;NO RESPONSE ?
21 006764   001427                BEQ     5$           ;YES, GET NEW PARAMETERS
22 006766   104401   071352      TYPE     ,CNSL08     ;NO, TYPE ' ILLEGAL INPUT '
23 006772   000752                BR      1$           ;TRY AGAIN
24
25         ;SEE IF OPERATOR WANTS HELP TEXT
26         3$:
27 006774   104401   070345      TYPE     ,MSHELP     ;WANT HELP ?
28 007000   104411                RDCHR                ;GET RESPONSE
29 007002   012637   001176      MOV      (SP)+,$TMP1  ;SAVE AND ECHO RESPONSE
30 007006   104401   001176      TYPE     ,TMP1
31 007012   123727   001176   000131  CMPB    $TMP1,#'Y    ;WAS IT A YES RESPONSE ?
32 007020   001407                BEQ     4$           ;YES
33 007022   123727   001176   000116  CMPB    $TMP1,#'N    ;WAS IT A NO RESPONSE ?
34 007030   001405                BEQ     5$           ;YES
35 007032   104401   071352      TYPE     ,CNSL08     ;NO, TYPE ' ILLEGAL INPUT '
36 007036   000756                BR      3$           ;TRY AGAIN
37 007040   104401   106314   4$:    TYPE     ,HELP       ;YES - TYPE HELP TEXT
38
39         ;SEE IF USER WANTS TO CHANGE ADDRESSES
40         5$:
41 007044   104401   001217      TYPE     ,SCRLF      ;CR-LF
42 007050   104401   070722      TYPE     ,UBUSQST    ;WANT TO CHANGE ADDRESS ?
43 007054   104411                RDCHR                ;GET RESPONSE
44 007056   012637   001176      MOV      (SP)+,$TMP1  ;SAVE AND ECHO RESPONSE
45 007062   104401   001176      TYPE     ,TMP1
46 007066   123727   001176   000131  CMPB    $TMP1,#'Y    ;WAS IT A YES RESPONSE ?
47 007074   001407                BEQ     6$           ;YES
48 007076   123727   001176   000116  CMPB    $TMP1,#'N    ;WAS IT A NO RESPONSE ?
49 007104   001525                BEQ     12$          ;YES
50 007106   104401   071352      TYPE     ,CNSL08     ;NO, TYPE ' ILLEGAL INPUT '
51 007112   000756                BR      5$+4        ;TRY AGAIN
52
53         ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
54         6$:
55 007114   104401   071010      TYPE     ,CNSL01     ;TYPE CURRENT BUS ADDRESS
56 007120   013746   001276      MOV     $BASE,-(SP)  ;;SAVE $BASE FOR TYPEOUT
          007124   104402                TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```

57 007126 104401 070336          TYPE      ,QUES          ;TYPE " ? "
58 007132 104413          RDOCT          ;GET NEW BUS ADDRESS
59 007134 012637 001176          MOV      (SP)+,$TMP1    ;CARRIAGE RETURN ?
60 007140 001412          BEQ      8$            ;YES-SKIP TO NEXT ENTRY
61 007142 022737 160000 001176    CMP      #160000,$TMP1  ;BASE ADDRESS IN I/O PAGE ?
62 007150 101403          BLOS     7$            ;YES
63 007152 104401 071026          TYPE      ,CNSL02      ;TYPE WARNING MESSAGE
64 007156 000760          BR       6$+4         ;TRY AGAIN
65 007160 013737 001176 001276 7$:  MOV      $TMP1,$BASE    ;STORE NEW BUS ADDRESS
66
67 007166 104401 071070          8$:      TYPE      ,CNSL03
68 007172 005046          CLR      -(SP)
69 007174 113716 001272          MOVVB   $VECT1,(SP)    ;GET CURRENT VECTOR ADDRESS
70 007200 104403          TYPOS
71 007202      003          .BYTE    3            ;TYPE 3 DIGITS
72 007203      000          .BYTE    0            ;SUPPRESS LEADING ZEROS
73 007204 104401 070336          TYPE      ,QUES          ;TYPE " ? "
74 007210 104413          RDOCT          ;GET NEW VECTOR ADDRESS
75 007212 012637 001176          MOV      (SP)+,$TMP1    ;CARRIAGE RETURN?
76 007216 001412          BEQ      10$           ;YES-SKIP TO NEXT ENTRY
77 007220 022737 001000 001176    CMP      #1000,$TMP1   ;VECTOR ADDRESS < 1000 ?
78 007226 101003          BHI      9$            ;YES!!
79 007230 104401 071110          TYPE      ,CNSL04      ;TYPE WARNING MESSAGE
80 007234 000754          BR       8$            ;RETRY
81 007236 113737 001176 001272 9$:  MOVVB   $TMP1,$VECT1   ;STORE NEW VECTOR ADDRESS
82
83 007244 104401 071144          10$:     TYPE      ,CNSL05
84 007250 005046          CLR      -(SP)
85 007252 113716 001273          MOVVB   $VECT1+1,(SP) ;GET CURRENT BR LEVEL
86 007256 006216          ASR      (SP)
87 007260 006216          ASR      (SP)
88 007262 006216          ASR      (SP)
89 007264 006216          ASR      (SP)
90 007266 006216          ASR      (SP)
91 007270 104403          TYPOS
92 007272      001          .BYTE    1            ;ONLY 1 DIGIT
93 007273      000          .BYTE    0            ;SUPPRESS LEADING ZEROS
94 007274 104401 070336          TYPE      ,QUES
95 007300 104413          RDOCT          ;GET NEW PRIORITY
96 007302 012637 001176          MOV      (SP)+,$TMP1    ;CARRIAGE RETURN ?
97 007306 001424          BEQ      12$           ;YES-SKIP TO NEXT ENTRY
98 007310 023727 001176 000007    CMP      $TMP1,#7      ;LEGAL PRIORITY ?
99 007316 002403          BLT     11$           ;YES!!
100 007320 104401 071156          TYPE      ,CNSL06      ;TYPE WARNING MESSAGE
101 007324 000747          BR       10$          ;TRY AGAIN
102 007326 006337 001176          11$:     ASL      $TMP1        ;STORE NEW PRIORITY
103 007332 006337 001176          ASL      $TMP1
104 007336 006337 001176          ASL      $TMP1
105 007342 006337 001176          ASL      $TMP1
106 007346 006337 001176          ASL      $TMP1
107 007352 113737 001176 001273    MOVVB   $TMP1,$VECT1+1
108
109          ;DIALOGUE TO INPUT DEVICE NUMBERS
110 007360          12$:
111 007360 005227 177777          INC      #-1          ;FIRST TIME THRU ?
112 007364 001002          BNE     13$          ;BR IF NO
113 007366 104401 071207          TYPE      ,CNSL07      ;TYPE INPUT INSTRUCTIONS

```

```

114 007372 104401 001217          13$:  TYPE      , $CRLF          ;CR-LF
115 007376 005037 001300          14$:  CLR      $DEVM          ;CLEAR DEVICE MAP
116 007402 104401 071374          TYPE      ,MSDRVS        ;TYPE 'DRIVE(S): '
117 007406 104411          RDCHR
118 007410 012637 001176          MOV      (SP)+,$TMP1      ;GET RESPONSE
119 007414 023727 001176 000101    CMP      $TMP1,#'A        ;IS INPUT 'A' ?
120 007422 001007          BNE      15$              ;NO
121 007424 104401 070331          TYPE      ,ALL            ;YES, TYPE 'ALL' AND GO
122 007430 012737 000377 001300    MOV      #377,$DEVM       ;SET DEVICE MAP FOR ALL DRIVES
123 007436 000137 006462          JMP      XSIZ             ;AUTO SIZE.
124
125 007442 023727 001176 000015    15$:  CMP      $TMP1,#CR        ;CARRIAGE RETURN ?
126 007450 001436          BEQ      17$              ;YES
127 007452 104401 001176          TYPE      , $TMP1         ;ECHO RESPONSE
128 007456 023727 001176 000060    CMP      $TMP1,#'0        ;NUMBER < 0 ?
129 007464 002430          BLT      17$              ;YES
130 007466 023727 001176 000067    CMP      $TMP1,#'7        ;NUMBER > 7 ?
131 007474 003427          BLE      18$              ;NO
132 007476 000423          BR       17$              ;ILLEGAL INPUT
133
134 007500 104411          16$:  RDCHR
135 007502 012637 001176          MOV      (SP)+,$TMP1      ;GET RESPONSE
136 007506 023727 001176 000015    CMP      $TMP1,#CR        ;CARRIAGE RETURN ?
137 007514 001432          BEQ      19$              ;YES
138 007516 104401 070342          TYPE      ,COMMA          ;TYPE ','
139 007522 104401 001176          TYPE      , $TMP1         ;ECHO RESPONSE
140 007526 023727 001176 000060    CMP      $TMP1,#'0        ;NUMBER < 0 ?
141 007534 002404          BLT      17$              ;YES
142 007536 023727 001176 000067    CMP      $TMP1,#'7        ;NUMBER > 7 ?
143 007544 003403          BLE      18$              ;NO
144 007546 104401 071352          17$:  TYPE      ,CNSL08        ;TYPE '' ?ILLEGAL INPUT''
145 007552 000711          BR       14$              ;RETRY
146
147 007554 013701 001176          18$:  MOV      $TMP1,R1         ;R1 = DRIVE NUMBER
148 007560 042701 177770          BIC      #^C7,R1
149 007564 156137 071616 001300    BISB    ATNTBL(R1),$DEVM  ;SET DEVICE IN MAP
150 007572 122737 000377 001300    CMPB    #377,$DEVM        ;DONE ?
151 007600 101337          BHI      16$              ;NO
152 007602 104401 001217          19$:  TYPE      , $CRLF          ;CR-LF
153 007606 000137 006462          JMP      XSIZ             ;GO SIZE DEVICES
154

```

```

1
2 007612
3 007612 013700 001300
4 007616 012701 001466
5 007622 010137 001464
6 007626 012702 000001
7 007632 005003
8 007634 030200
9 007636 001406
10 007640 010311
11 007642 116361 071616 000001
12 007650 062701 000002
13 007654 006302
14 007656 105702
15 007660 001402
16 007662 005203
17 007664 000763
18 007666 005011
19
20
21 007670 004737 044720
22 007674 000413
23 007676
24 007676 104000
24 007700 104401 007706
007704 000405
007720
25 007720 000000
26 007722 000765
27 007724
007724 005737 000042
007730 001012
007732 123727 001242 000001
007740 001406
007742 023727 001154 000176
007750 001005
007752 104407
007754 000403
007756 112737 000001 001150
007764
28
29 007764 000240
30 007766 105737 001300
31 007772 001007
32 007774 005737 000042
33 010000 001002
34 010002 000137 005420
35 010006 000137 037460
36
37 010012 105037 001116
38 010016 005037 001206
39 010022 005037 001326
40 010026 004737 066146
41 010032 012746 000300
010036 012746 010044

;ASSEMBLE TEST QUE FROM DEVICE MAP
CMNSTART:
MOV $DEVN,R0 ;R0 = DEVICE MAP
MOV #TSTQUE+2,R1 ;R1 = ADDRESS OF FIRST ENTRY IN QUE
MOV R1,TSTQUE ;INITIALIZE ENTRY POINTER
MOV #1,R2 ;R2 = DEVICE POINTER
CLR R3 ;R3 = DEVICE NUMBER
1$: BIT R2,R0 ;IS THIS DEVICE IN MAP ?
BEQ 2$ ;NO !!
MOV R3,(R1) ;YES - ENTER DEVICE NUMBER IN QUE
MOVB ATNTBL(R3),1(R1) ;ENTER ATTENTION BIT IN QUE
ADD #2,R1 ;ADVANCE ENTRY POINTER
2$: ASL R2 ;ADVANCE DEVICE POINTER
TSTB R2 ;DONE ALL DEVICES ?
BEQ 3$ ;YES
INC R3 ;ADVANCE DEVICE NUMBER
BR 1$ ;ENTER NEXT DEVICE
3$: CLR (R1) ;TERMINATE TEST QUE

;SIZE FOR CLOCK
JSR PC,SIZCLK ;SEE IF CLOCK PRESENT
BR 5$ ;YES - CLOCK IS PRESENT
4$: EMT
TYPE ,65$ ;;TYPE ASCIZ STRING
BR ,64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <CRLF>/PROG HLT/
64$: HALT ;PROGRAM HALT !!
BR 4$
5$: .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
BNE 66$ ;;BRANCH IF YES
CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
BEQ 66$ ;;BRANCH IF YES
CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
BNE 67$ ;;BRANCH IF NO
GTSWR ;;GET SOFT-SWR SETTINGS
BR 67$
66$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
67$:

READY: NOP ;READY TO START TEST
TSTB $DEVN ;ANY DRIVES IN MAP ?
BNE 2$ ;BR IF YES
TST @#42 ;ANY MONITOR PRESENT ?
BNE 1$ ;BR IF YES
JMP START ;JUMP TO START
1$: JMP $EOP ;RETURN CONTROL TO MONITOR
2$: CLRB $TSTNM ;RESET TEST NUMBER
CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
CLR CTLFG ;CLEAR CONTROL-C FLAG
JSR PC,$TKINT ;INITIALIZE TTY
MOV #PR6,-(SP) ;;PUT NEW PS ON STACK
MOV #64$,-(SP) ;;PUT NEW PC ON STACK

```

	010042	000002			RTI		::POP NEW PC AND PS
	010044						
42	010044	117737	171414	001234	64\$:	MOVB @TSTQUE,\$UNIT	:LOAD UNIT NUMBER
43	010052	005037	001510			CLR MEDENB	:CLEAR MEDIA ENABLE
44							
45							
46							:CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND DETERMINE THE LAST TRACK
47	010056	012737	002000	001332			:OF THE DIFFERENT DRIVE TYPES
48	010064	013700	001276			MOV #TA4,LSTRK	:ASSUME LAST TRACK FOR RM02/3 = 4.
49	010070	012760	000040	000010		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
50	010076	117760	171362	000010		MOV #CLR,RMCS2(R0)	:CLEAR MASSBUS
51	010104	016002	000026			MOVB @TSTQUE,RMCS2(R0)	:SELECT DEVICE UNDER TEST
52	010110	042702	177770			MOV RMDT(R0),R2	:GET RMDT AND
53	010114	022702	000007			BIC #177770,R2	:SAVE DRIVE TYPE BITS
54	010120	001003				CMP #7,R2	:IS IT AN RM05 ?
55	010122	012737	011000	001332		BNE 3\$:NO, MUST BE AN RM02 OR RM03
56	010130				3\$:	MOV #TA16!TA2,LSTRK	:YES--SET LAST TRACK = 18.
57							

1
2

 : *TEST 1 CONTROLLER ACCESS TEST

TST1:

010130	000004			SCOPE	:SCOPE CALL
010130	000240			NOP	:START OF TEST
010134	012706	001100		MOV #STACK,SP	:INITIALIZE STACK POINTER
010140	013700	001276		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
010144	013701	001464		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
010150	012737	000001	001226	MOV #1,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
3					
4	010156	005001		CLR R1	
5	010160	013746	000004	MOV ERRVEC,-(SP)	::PUSH ERRVEC ON STACK
6	010164	013746	000006	MOV ERRVEC+2,-(SP)	::PUSH ERRVEC+2 ON STACK
7	010170	012737	010262	MOV #3\$,ERRVEC	
8	010176	012737	000300	MOV #PR6,ERRVEC+2	
9					
10	010204	110160	000001	MOVB R1,RMCS1+1(R0)	:MOVE HI BYTE TO RMCS1
11	010210	010160	000002	MOV R1,RMWC(R0)	:MOVE WORD COUNT REGISTER
12	010214	016002	000002	MOV RMWC(R0),R2	
13	010220	010160	000004	MOV R1,RMBA(R0)	:MOVE BUS ADDRESS REGISTER
14	010224	016002	000004	MOV RMBA(R0),R2	
15	010230	010160	000010	MOV R1,RMCS2(R0)	:MOVE CONTROL STATUS REGISTER
16	010234	016002	000010	MOV RMCS2(R0),R2	
17	010240	010160	000022	MOV R1,RMDB(R0)	:MOVE DATA BUFFER
18	010244	016002	000022	MOV RMDB(R0),R2	
19	010250	012637	000006	MOV (SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
20	010254	012637	000004	MOV (SP)+,ERRVEC	::POP STACK INTO ERRVEC
21	010260	000415		BR 7\$:NO BUS TIMEOUT OCCURRED
22					
23	010262	022626		3\$: CMP (SP)+,(SP)+	:ADJUST STACK
24	010264	012637	000006	MOV (SP)+,ERRVEC+2	::POP STACK INTO ERRVEC+2
25	010270	012637	000004	MOV (SP)+,ERRVEC	::POP STACK INTO ERRVEC
26	010274	104110		EMT 110	
27	010276	005737	000042	TST @#42	:STAND ALONE MODE ?
28	010302	001002		BNE 5\$:NO!!
29	010304	000137	005420	JMP START	:YES-GO GET \$BASE
30					
31	010310	000137	037460	5\$: JMP \$EOP	:GO TO END OF PASS HANDLER
32	010314			7\$:	
33					
34					

 : *TEST 2 DEVICE AVAILABLE TEST

TST2:

010314	000004			SCOPE	:SCOPE CALL
010314	000240			NOP	:START OF TEST
010320	012706	001100		MOV #STACK,SP	:INITIALIZE STACK POINTER
010324	013700	001276		MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
010330	013701	001464		MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
010334	012737	000002	001226	MOV #2,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
35					
36	010342	004737	053604	JSR PC,CNTCLR	:GO ISSUE CONTROLLER CLEAR
	010346	000404		BR 2\$:GO TO 2\$ IF NO ERROR
	010350	000240		NOP	:RETURN HERE IF ERROR
	010352	104000		EMT	:ERROR NUMBER DEFINED BY SUBROUTINE

```

37 010354 000137 010474          JMP      7$          ;GO TO 7$ IF ERROR
38 010360          2$:      MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
39 010364 013746 000004          MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
40 010370 012737 010460 000004  MOV      #5$,ERRVEC
41 010376 013737 000300 000006  MOV      PR6,ERRVEC+2
42
43 010404 016037 000000 001176  MOV      RMCS1(R0),$TMP1 ;GET DVA STATUS
44 010412 016037 000010 001174  MOV      RMCS2(R0),$TMP0 ;GET NED STATUS
45 010420 012637 000006          MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
46 010424 012637 000004          MOV      (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
47 010430 032737 010000 001174  BIT      #NED,$TMP0      ;NONEXISTENT DEVICE ?
48 010436 001402          BEQ      3$          ;NO!!
49 010440 104111          EMT      111
50 010442 000414          BR      7$
51 010444 032737 004000 001176  3$:      BIT      #DVA,$TMP1    ;DEVICE AVAILABLE ?
52 010452 001012          BNE      9$          ;YES!!
53 010454 104112          EMT      112
54 010456 000406          BR      7$
55
56 010460 022626          5$:      CMP      (SP)+,(SP)+    ;ADJUST STACK
57 010462 012637 000006          MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
58 010466 012637 000004          MOV      (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
59 010472 104113          EMT      113
60 010474 000137 037422          7$:      JMP      $EOSP
61 010500          9$:
62
63

```

;*TEST 3 DRIVE TYPE TEST

TST3:

```

010500          SCOPE          ;SCOPE CALL
010500 000004          NOP          ;START OF TEST
010502 000240          MOV      #STACK,SP    ;INITIALIZE STACK POINTER
010504 012706 001100          MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
010510 013700 001276          MOV      TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
010514 013701 001464          MOV      #3,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
010520 012737 000003 001226
64
65 010526 004737 053604          JSR      PC,CNTCLR    ;GO ISSUE CONTROLLER CLEAR
010532 000404          BR      2$          ;GO TO 2$ IF NO ERROR
010534 000240          NOP          ;RETURN HERE IF ERROR
010536 104000          EMT          ;ERROR NUMBER DEFINED BY SUBROUTINE
010540 000137 010676          JMP      4$          ;GO TO 4$ IF ERROR
66 010544          2$:
67 010544 112737 000026 001522  MOVB     #RMDT,GETINX  ;SETUP GET INDEX TABLE
010552 112737 000200 001523  MOVB     #200,GETINX+1 ;SETUP TERMINATOR BYTE
010560 012737 010702 001362  MOV      #5$,RMDTI    ;SET RMDT INPUT BUFFER = 5$
010566 004737 044230          JSR      PC,GET      ;GO READ RMDT VIA GET SUBROUTINE
010572 000402          BR      3$          ;GO TO 3$ IF NO ERROR
010574 000240          NOP          ;RETURN HERE IF ERROR
010576 104000          EMT          ;ERROR DEFINED BY GET SUBROUTINE
68
69 010600 022737 020024 001362  3$:      CMP      #SNGPRT,RMDTI ;SINGLE PORT RM03 ?
70 010606 001435          BEQ      5$          ;YES !!
71 010610 022737 024024 001362  CMP      #DULPRT,RMDTI ;DUAL PORT RM03 ?
72 010616 001431          BEQ      5$          ;YES !!
73

```

```

74 010620 022737 020025 001362      CMP      #SNGPRT!BIT0,RMDTI      ;SINGLE PORT RM02 ?
75 010626 001425                      BEQ      5$                      ;YES !!
76 010630 022737 024025 001362      CMP      #DULPRT!BIT0,RMDTI      ;DUAL PORT RM02 ?
77 010636 001421                      BEQ      5$                      ;YES !!
78
79 010640 022737 020027 001362      CMP      #SNGPRT!BIT1!BIT0,RMDTI ;SINGLE PORT RM05 ?
80 010646 001415                      BEQ      5$                      ;YES !!
81 010650 022737 024027 001362      CMP      #DULPRT!BIT1!BIT0,RMDTI ;DUAL PORT RM05 ?
82 010656 001411                      BEQ      5$
83
84 010660 012737 020024 001176      MOV      #SNGPRT,$TMP1          ;LOAD ACCEPTABLE VALUES
85 010666 012737 024024 001200      MOV      #DULPRT,$TMP2
86 010674 104114                      EMT      114
87
88 010676 000137 037422      4$:    JMP      $EOSP              ;GO TO SUBPASS HANDLER.
89 010702      5$:
90
91
    ;*****
    ;*TEST 4      WRITE, READ ZEROS
    ;*****
    TST4:
    SCOPE                      ;SCOPE CALL
    NOP                        ;START OF TEST
    MOV      #STACK,SP         ;INITIALIZE STACK POINTER
    MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
    MOV      TSTQUE,R1         ;(R1) = DEVICE BEING TESTED
    MOV      #4,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX

    ;*****
    ;LOOP #1      FORMAT,WRITE,READ
    ;*****
    10$:
    ;PREPARE DEVICE FOR TEST
    JSR      PC,TSTPRP         ;PREPARE DEVICE FOR TEST
    .WORD   154130             ;TASK DESCRIPTOR AS FOLLOWS:
    ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
    ;VERIFY RECALIBRATION
    BR      20$                ;GO TO 20$ IF NO ERROR
    NOP                        ;RETURN HERE IF ERROR
    EMT
    JMP      350$              ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    ;GO TO 350$ IF ERROR

    ;LOAD PARAMETERS AND GENERATE DATA BUFFER
    20$:
    MOV      #0,RMDCO          ;CYLINDER = 0
    MOV      #0,RMDAO          ;TRACK = 0, SECTOR = 0
    MOV      #BUFONE,RMBAO     ;BUS ADDRESS
    MOV      #-258.,RMWCO      ;2 + 256. WORDS (2'S COMP)
    MOV      #FMT16,RMFO       ;16 BIT FORMAT
    MOV      #WH!GO,RMCS10     ;WRITE HEADER AND DATA COMMAND
    
```

100
 101
 102 010750
 103 010750 012737 000000 001444
 104 010756 012737 000000 001416
 105 010764 012737 106314 001414
 106 010772 012737 177376 001412
 107 011000 012737 010000 001442
 108 011006 012737 000063 001410
 109

```

110      011014 004737 041416      ;VERIFY THAT SECTOR IS NOT BAD
      011020 000405      JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
      011022 104401 070230      BR       25$           ;GO TO 25$ IF NO ERROR
      011026 104000      TYPE     ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
      011030 000137 011710      EMT     ;ERROR # DEFINED BY BADSCT SUBROUTINE
      011034      JMP     350$      ;GO TO 350$ IF ERROR
111 011034      25$:
112 011034 012737 071730 001174      MOV     #ZEROS,$TMP0      ;STARTING ADDRESS OF PATTERN
113 011042 012737 000001 001176      MOV     #1,$TMP1        ;RANGE OF PATTERN
114 011050 004737 043344      JSR     PC,GENBUF       ;GO GENERATE BUFFER FOR FORMAT
115
116      ;SETUP PARAMETERS AND EXECUTE COMMAND
117 011054 012702 001551      MOV     #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
118 011060 112722 000034      MOVB   #RMDC,(R2)+
119 011064 112722 000006      MOVB   #RMDA,(R2)+
120 011070 112722 000004      MOVB   #RMBA,(R2)+
121 011074 112722 000002      MOVB   #RMWC,(R2)+
122 011100 112722 000032      MOVB   #RMOF,(R2)+
123 011104 112722 000000      MOVB   #RMCS1,(R2)+
124 011110 112712 000200      MOVB   #200,(R2)      ;TERMINATE TABLE
125 011114      30$:
126 011114 004737 044500      JSR     PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011120 000404      BR     40$           ;GO TO 40$ IF NO ERROR
      011122 000240      NOP
      011124 104000      EMT     ;RETURN HERE IF ERROR
      011126 000137 011710      JMP     350$          ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 350$ IF ERROR
127 011132      40$:
128
129      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      011132 004737 044144      JSR     PC,GETSTS      ;GO TO GETSTS SUBROUTINE
130
131      ;WAIT FOR COMMAND TO COMPLETE
      011136 004737 045042      JSR     PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
132
133      ;GO GET REGISTER STATUS
134 011142 004737 044230      JSR     PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011146 000404      BR     50$           ;GO TO 50$ IF NO ERROR
      011150 000240      NOP                ;RETURN HERE IF ERROR
      011152 104000      EMT     ;ERROR # DEFINED BY GET SUBROUTINE
      011154 000137 011710      JMP     350$          ;GO TO 350$ IF ERROR
135 011160      50$:
136
137      ;VERIFY RESULTS OF WRITE COMMAND
138 011160 004737 057742      JSR     PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      011164 000405      BR     60$           ;GO TO 60$ IF NO ERROR
      011166 000240      NOP                ;RETURN HERE IF ERROR
      011170 104000      EMT     ;ERROR # DEFINED BY DTASTS SUBROUTINE
      011172 004736      JSR     PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      011174 000137 011710      JMP     350$          ;GO TO 350$ IF ERROR
139 011200      60$:
140
141      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
142 011200 012737 011210 001124      MOV     #70$,$LPERR
143 011206 000410      BR     80$           ;SKIP TO WRITE OPERATION
144
145      ;*****
146      ;LOOP #2      WRITE,READ

```

```

147
148 011210          70$:
149
150
151 011210 004737 040472 ;PREPARE DEVICE FOR TEST
      011214 154130      JSR   PC,TSTPRP      ;PREPARE DEVICE FOR TEST
                          .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                          ;CLEAR CONTROLLER & SELECT DEVICE
                                          ;VERIFY CONTROLLER CLEAR OPERATION
                                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                          ;VERIFY PACK ACKNOWLEDGE
                                          ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                          ;VERIFY RECALIBRATION
                                          ;GO TO 80$ IF NO ERROR
                                          ;RETURN HERE IF ERROR
                                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                          ;GO TO 350$ IF ERROR

      011216 000404      BR     80$
      011220 000240      NOP
      011222 104000      EMT
      011224 000137 011710 JMP   350$
152 011230          80$:
153
154          ;SETUP PARAMETERS AND EXECUTE COMMAND
155 011230          120$:
156 011230 012737 106320 001414 MOV   #BUFONE+4,RMBA0      ;MOVE MEMORY ADDRESS
157 011236 012737 177400 001412 MOV   #-256.,RMWCO      ;CHANGE WORD COUNT
158 011244 012737 000061 001410 MOV   #WD!GO,RMCS10     ;WRITE DATA COMMAND
159 011252 012702 001551      MOV   #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
160 011256 112722 000006      MOVB  #RMDA,(R2)+
161 011262 112722 000034      MOVB  #RMDC,(R2)+
162 011266 112722 000032      MOVB  #RMOF,(R2)+
163 011272 112722 000004      MOVB  #RMBA,(R2)+
164 011276 112722 000002      MOVB  #RMWC,(R2)+
165 011302 112722 000000      MOVB  #RMCS1,(R2)+
166 011306 112722 000200      MOVB  #200,(R2)+      ;TERMINATE TABLE
167
168 011312 004737 044500      JSR   PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011316 000404      BR     130$           ;GO TO 130$ IF NO ERROR
      011320 000240      NOP
      011322 104000      EMT
      011324 000137 011710 JMP   350$           ;RETURN HERE IF ERROR
                                          ;ERROR # DEFINED BY PUT SUBROUTINE
                                          ;GO TO 350$ IF ERROR
169 011330          130$:
170
171          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      011330 004737 044144      JSR   PC,GETSTS        ;GO TO GETSTS SUBROUTINE
172
173          ;WAIT FOR COMMAND TO COMPLETE
      011334 004737 045042      JSR   PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
174
175          ;GO GET REGISTER STATUS
176 011340 004737 044230      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011344 000404      BR     140$           ;GO TO 140$ IF NO ERROR
      011346 000240      NOP
      011350 104000      EMT
      011352 000137 011710 JMP   350$           ;RETURN HERE IF ERROR
                                          ;ERROR # DEFINED BY GET SUBROUTINE
                                          ;GO TO 350$ IF ERROR
177 011356          140$:
178
179          ;VERIFY RESULTS OF WRITE COMMAND
180 011356 004737 045226      JSR   PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      011362 000405      BR     150$           ;GO TO 150$ IF NO ERROR

```

```

011364 000240      NOP      ;RETURN HERE IF ERROR
011366 104000      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
011370 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
011372 000137 011710 JMP      350$      ;GO TO 350$ IF ERROR
181 011376      150$:
011376 004737 057742 JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
011402 000405      BR      160$      ;GO TO 160$ IF NO ERROR
011404 000240      NOP      ;RETURN HERE IF ERROR
011406 104000      EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
011410 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
011412 000137 011710 JMP      350$      ;GO TO 350$ IF ERROR
183 011416      160$:
184 011416 004737 046060 JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
011422 000405      BR      180$      ;GO TO 180$ IF NO ERROR
011424 000240      NOP      ;RETURN HERE IF ERROR
011426 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
011430 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
011432 000137 011710 JMP      350$      ;GO TO 350$ IF ERROR
185 011436      180$:
186
187
188 011436 012737 011446 001124 ;CHANGE LOOP ADDRESSES
189 011444 000410      MOV      #190$,$LPERR
190
191
192
193
194 011446      BR      200$      ;SKIP TO NEXT OPERATION
195
196
197 011446 004737 040472 ;PREPARE DEVICE FOR TEST
011452 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 200$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 350$ IF ERROR
011454 000404      BR      200$
011456 000240      NOP
011460 104000      EMT
011462 000137 011710 JMP      350$
198 011466      200$:
199
200
201 011466      ;SETUP PARAMETERS AND EXECUTE COMMAND
202 011466 012737 107324 001414 240$:
203 011474 012737 000071 001410      MOV      #BUFTWO+4,RMBAO ;CHANGE MEMORY ADDRESS
204 011502 012702 001551      MOV      #RD!GO,RMCS10 ;READ DATA COMMAND
205 011506 112722 000006      MOV      #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
206 011512 112722 000032      MOV      #RMDA,(R2)+
207 011516 112722 000034      MOV      #RMOF,(R2)+
208 011522 112722 000004      MOV      #RMDC,(R2)+
209 011526 112722 000002      MOV      #RMBA,(R2)+
210 011532 112722 000000      MOV      #RMWC,(R2)+
211 011536 112712 000200      MOV      #RMCS1,(R2)+
                                MOV      #200,(R2)

```

```

212
213 011542 004737 044500      JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011546 000404      BR     250$        ;GO TO 250$ IF NO ERROR
      011550 000240      NOP                    ;RETURN HERE IF ERROR
      011552 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      011554 000137 011710    JMP    350$        ;GO TO 350$ IF ERROR
214 011560      250$:
215
216      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      011560 004737 044144    JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
217
218      ;WAIT FOR COMMAND TO COMPLETE
      011564 004737 045042    JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
219
220      ;GO GET REGISTER STATUS
221 011570 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011574 000404      BR     260$        ;GO TO 260$ IF NO ERROR
      011576 000240      NOP                    ;RETURN HERE IF ERROR
      011600 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      011602 000137 011710    JMP    350$        ;GO TO 350$ IF ERROR
222 011606      260$:
223
224      ;VERIFY RESULTS OF READ COMMAND
225 011606 004737 045226      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      011612 000405      BR     270$        ;GO TO 270$ IF NO ERROR
      011614 000240      NOP                    ;RETURN HERE IF ERROR
      011616 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      011620 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      011622 000137 011710    JMP    350$        ;GO TO 350$ IF ERROR
226 011626      270$:
227 011626 004737 057742      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      011632 000405      BR     280$        ;GO TO 280$ IF NO ERROR
      011634 000240      NOP                    ;RETURN HERE IF ERROR
      011636 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      011640 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      011642 000137 011710    JMP    350$        ;GO TO 350$ IF ERROR
228 011646      280$:
229 011646 004737 046060      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      011652 000405      BR     300$        ;GO TO 300$ IF NO ERROR
      011654 000240      NOP                    ;RETURN HERE IF ERROR
      011656 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      011660 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      011662 000137 011710    JMP    350$        ;GO TO 350$ IF ERROR
230 011666      300$:
231
232      ;GO VERIFY DATA
233 011666 004737 043602      JSR    PC,CMPBUF   ;GO COMPARE WRITE, READ DATA BUFFERS
      011672 106320      .WORD  BUFOONE+4   ;STARTING ADDRESS OF WRITE BUFFER
      011674 107324      .WORD  BUFTWO+4   ;STARTING ADDRESS OF READ BUFFER
      011676 000404      BR     310$        ;GO TO 310$ IF NO ERROR
      011700 000240      NOP                    ;RETURN HERE IF ERROR
      011702 104000      EMT                    ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      011704 000137 011710    JMP    350$        ;GO TO 350$ IF ERROR
234 011710      310$:
235
236 011710      350$:
237

```

238

```

:*****
:*TEST 5      WRITE, WRITE CHECK ZEROS
:*****
TST5:

```

```

011710
011710 000004
011712 000240
011714 012706 001100
011720 013700 001276
011724 013701 001464
011730 012737 000005 001226

```

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #5,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

239

240

241

242

243 011736

244

245

```

246 011736 004737 040472
011742 154130

```

```

:*****
:LOOP #1      FORMAT,WRITE,WRITE CHECK DATA

```

10\$:

```

;PREPARE DEVICE FOR TEST
JSR PC,TSTPRP ;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 20$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR

```

```

011744 000404
011746 000240
011750 104000
011752 000137 012666

```

```

BR 20$
NOP
EMT
JMP 350$

```

247

248

249 011756

```

250 011756 012737 000000 001444
251 011764 012737 000000 001416
252 011772 012737 106314 001414
253 012000 012737 177376 001412
254 012006 012737 010000 001442
255 012014 012737 000063 001410

```

;LOAD PARAMETERS AND GENERATE DATA BUFFER

```

20$:
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #BUFONE,RMBAO ;BUS ADDRESS
MOV #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

```

256

257

```

012022 004737 041416
012026 000405
012030 104401 070230
012034 104000
012036 000137 012666

```

;VERIFY THAT SECTOR IS NOT BAD

```

JSR PC,BADSCT ;CALL BAD SECTOR MODULE
BR 25$ ;GO TO 25$ IF NO ERROR
TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR

```

258 012042

```

259 012042 012737 071730 001174
260 012050 012737 000001 001176
261 012056 004737 043344

```

25\$:

```

MOV #ZEROS,$TMP0 ;STARTING ADDRESS OF PATTERN
MOV #1,$TMP1 ;RANGE OF PATTERN
JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT

```

262

263

```

264 012062 012702 001551
265 012066 112722 000034
266 012072 112722 000006
267 012076 112722 000004
268 012102 112722 000002

```

;SETUP PARAMETERS AND EXECUTE COMMAND

```

MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
MOVB #RMDC,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+

```



```

269 012106 112722 000032      MOVB  #RMOF,(R2)+
270 012112 112722 000000      MOVB  #RMC51,(R2)+
271 012116 112712 000200      MOVB  #200,(R2)                ;TERMINATE TABLE
272 012122
273 012122 004737 044500      30$: JSR   PC,PUT                ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      012126 000404          BR    40$                    ;GO TO 40$ IF NO ERROR
      012130 000240          NOP
      012132 104000          EMT
      012134 000137 012666    JMP   350$                   ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 350$ IF ERROR
274 012140
275
276      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      012140 004737 044144    JSR   PC,GETSTS              ;GO TO GETSTS SUBROUTINE
277
278      ;WAIT FOR COMMAND TO COMPLETE
      012144 004737 045042    JSR   PC,TIMOUT             ;GO TO TIMOUT SUBROUTINE
279
280      ;GO GET REGISTER STATUS
281 012150 004737 044230      JSR   PC,GET                ;GO READ REGISTER(S) WITH GET SUBROUTINE
      012154 000404          BR    50$                    ;GO TO 50$ IF NO ERROR
      012156 000240          NOP
      012160 104000          EMT
      012162 000137 012666    JMP   350$                   ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 350$ IF ERROR
282 012166
283
284      ;VERIFY RESULTS OF WRITE COMMAND
285 012166 004737 057742      JSR   PC,DTASTS             ;GO VERIFY RESULTS OF DATA TRANSFER
      012172 000405          BR    60$                    ;GO TO 60$ IF NO ERROR
      012174 000240          NOP
      012176 104000          EMT
      012200 004736          JSR   PC,@(SP)+              ;ERROR # DEFINED BY DTASTS SUBROUTINE
      012202 000137 012666    JMP   350$                   ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
286 012206
287
288      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
289 012206 012737 012216 001124  MOV   #70$,SLPERR
290 012214 000410          BR    80$                    ;SKIP TO WRITE OPERATION
291
292      ;*****
293      ;LOOP #2      WRITE,WRITE CHECK DATA
294
295 012216      70$:
296
297      ;PREPARE DEVICE FOR TEST
298 012216 004737 040472      JSR   PC,TSTPRP            ;PREPARE DEVICE FOR TEST
      012222 154130          .WORD 154130              ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      012224 000404          BR    80$                    ;GO TO 80$ IF NO ERROR
      012226 000240          NOP
      012230 104000          EMT
      012232 000137 012666    JMP   350$                   ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 350$ IF ERROR
    
```

```

299 012236      80$:
300
301      ;SETUP PARAMETERS AND EXECUTE COMMAND
302 012236      120$:
303 012236 012737 106320 001414      MOV      #BUFONE+4,RMBA0      ;CHANGE MEMORY ADDRESS
304 012244 012737 177400 001412      MOV      #-256.,RMWCO      ;CHANGE WORD COUNT
305 012252 012737 000061 001410      MOV      #WD!GO,RMCS10      ;WRITE DATA COMMAND
306 012260 012702 001551      MOV      #PUTINX,R2      ;LOAD PUT REGISTER INDEX TABLE
307 012264 112722 000006      MOVVB   #RMDA,(R2)+
308 012270 112722 000034      MOVVB   #RMDC,(R2)+
309 012274 112722 000032      MOVVB   #RMOF,(R2)+
310 012300 112722 000004      MOVVB   #RMBA,(R2)+
311 012304 112722 000002      MOVVB   #RMWC,(R2)+
312 012310 112722 000000      MOVVB   #RMCS1,(R2)+
313 012314 112722 000200      MOVVB   #200,(R2)+      ;TERMINATE TABLE
314
315 012320 004737 044500      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      012324 000404      BR      130$      ;GO TO 130$ IF NO ERROR
      012326 000240      NOP      ;RETURN HERE IF ERROR
      012330 104000      EMT      ;ERROR # DEFINED BY PUT SUBROUTINE
      012332 000137 012666      JMP      350$      ;GO TO 350$ IF ERROR
316 012336
317
318      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      012336 004737 044144      JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
319
320      ;WAIT FOR COMMAND TO COMPLETE
      012342 004737 045042      JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
321
322      ;GO GET REGISTER STATUS
323 012346 004737 044230      JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      012352 000404      BR      140$      ;GO TO 140$ IF NO ERROR
      012354 000240      NOP      ;RETURN HERE IF ERROR
      012356 104000      EMT      ;ERROR # DEFINED BY GET SUBROUTINE
      012360 000137 012666      JMP      350$      ;GO TO 350$ IF ERROR
324 012364
325
326      ;VERIFY RESULTS OF WRITE COMMAND
327 012364 004737 045226      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      012370 000405      BR      150$      ;GO TO 150$ IF NO ERROR
      012372 000240      NOP      ;RETURN HERE IF ERROR
      012374 104000      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
      012376 004736      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      012400 000137 012666      JMP      350$      ;GO TO 350$ IF ERROR
328 012404
329 012404 004737 057742      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      012410 000405      BR      160$      ;GO TO 160$ IF NO ERROR
      012412 000240      NOP      ;RETURN HERE IF ERROR
      012414 104000      EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
      012416 004736      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      012420 000137 012666      JMP      350$      ;GO TO 350$ IF ERROR
330 012424
331 012424 004737 046060      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      012430 000405      BR      180$      ;GO TO 180$ IF NO ERROR
      012432 000240      NOP      ;RETURN HERE IF ERROR
      012434 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
      012436 004736      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    
```

```

332 012440 000137 012666          JMP      350$          ;GO TO 350$ IF ERROR
333 012444          180$:
334          ;CHANGE LOOP ADDRESSES
335 012444 012737 012454 001124  MOV      #190$, $LPERR
336 012452 000410          BR        200$          ;SKIP TO NEXT OPERATION
337
338          ;*****
339          ;LOOP #3          WRITE CHECK DATA
340
341 012454          190$:
342
343          ;PREPARE DEVICE FOR TEST
344 012454 004737 040472          JSR      PC, TSTPRP      ;PREPARE DEVICE FOR TEST
012460 154130          .WORD    154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 200$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 350$ IF ERROR
012462 000404          BR        200$
012464 000240          NOP
012466 104000          EMT
012470 000137 012666          JMP      350$
345 012474          200$:
346
347          ;SETUP PARAMETERS AND EXECUTE COMMAND
348 012474          240$:
349 012474 012737 000051 001410  MOV      #WCD!GO, RMCS10      ;WRITE CHECK DATA DATA COMMAND
350 012502 012702 001551          MOV      #PUTINX, R2          ;LOAD PUT REGISTER INDEX TABLE
351 012506 112722 000006          MOVB    #RMDA, (R2)+
352 012512 112722 000032          MOVB    #RMOF, (R2)+
353 012516 112722 000034          MOVB    #RMDC, (R2)+
354 012522 112722 000004          MOVB    #RMBA, (R2)+
355 012526 112722 000002          MOVB    #RMWC, (R2)+
356 012532 112722 000000          MOVB    #RMCS1, (R2)+
357 012536 112712 000200          MOVB    #200, (R2)
358
359 012542 004737 044500          JSR      PC, PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
012546 000404          BR        250$          ;GO TO 250$ IF NO ERROR
012550 000240          NOP          ;RETURN HERE IF ERROR
012552 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
012554 000137 012666          JMP      350$          ;GO TO 350$ IF ERROR
360 012560          250$:
361
362          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
363 012560 004737 044144          JSR      PC, GETSTS      ;GO TO GETSTS SUBROUTINE
364
365          ;WAIT FOR COMMAND TO COMPLETE
366 012564 004737 045042          JSR      PC, TIMEOUT     ;GO TO TIMEOUT SUBROUTINE
367
367          ;GO GET REGISTER STATUS
012570 004737 044230          JSR      PC, GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
012574 000404          BR        260$          ;GO TO 260$ IF NO ERROR
012576 000240          NOP          ;RETURN HERE IF ERROR
    
```

```

012600 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
012602 000137 012666 JMP 350$ ;GO TO 350$ IF ERROR
368 012606 260$:
369
370 ;VERIFY RESULTS OF WRITE CHECK COMMAND
371 012606 004737 045226 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
012612 000405 BR 270$ ;GO TO 270$ IF NO ERROR
012614 000240 NOP ;RETURN HERE IF ERROR
012616 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
012620 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
012622 000137 012666 JMP 350$ ;GO TO 350$ IF ERROR
372 012626 270$:
373 012626 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
012632 000405 BR 280$ ;GO TO 280$ IF NO ERROR
012634 000240 NOP ;RETURN HERE IF ERROR
012636 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
012640 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
012642 000137 012666 JMP 350$ ;GO TO 350$ IF ERROR
374 012646 280$:
375 012646 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
012652 000405 BR 300$ ;GO TO 300$ IF NO ERROR
012654 000240 NOP ;RETURN HERE IF ERROR
012656 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
012660 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
012662 000137 012666 JMP 350$ ;GO TO 350$ IF ERROR
376 012666 300$:
377
378 012666 350$:
379
380
;*****
;*TEST 6 WRITE, WRITE CHECK ZEROS W/ WCE ERROR
;*****
TST6:
012666 000004 SCOPE ;SCOPE CALL
012670 000240 NOP ;START OF TEST
012672 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
012676 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
012702 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
012706 012737 000006 001226 MOV #6,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
381
382
383 ;*****
384 ;LOOP #1 FORMAT,WRITE,WRITE CHECK DATA
385 012714 10$:
386
387 ;PREPARE DEVICE FOR TEST
388 012714 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
012720 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
012722 000404 BR 20$ ;GO TO 20$ IF NO ERROR
012724 000240 NOP ;RETURN HERE IF ERROR
    
```

```

012726 104000          EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
012730 000137 014062  JMP          350$          ;GO TO 350$ IF ERROR
389
390          ;LOAD PARAMETERS AND GENERATE DATA BUFFER
391 012734          20$:
392 012734 012737 000000 001444      MOV          #0,RMDCO          ;CYLINDER = 0
393 012742 012737 000000 001416      MOV          #0,RMDAO          ;TRACK = 0, SECTOR = 0
394 012750 012737 106314 001414      MOV          #BUFONE,RMBAO     ;BUS ADDRESS
395 012756 012737 177376 001412      MOV          #-258.,RMWCO      ;2 + 256. WORDS (2'S COMP)
396 012764 012737 010000 001442      MOV          #FMT16,RMOFO      ;16 BIT FORMAT
397 012772 012737 000063 001410      MOV          #WH!GO,RMCS1O     ;WRITE HEADER AND DATA COMMAND
398
399          ;VERIFY THAT SECTOR IS NOT BAD
013000 004737 041416      JSR          PC,BADSCT         ;CALL BAD SECTOR MODULE
013004 000405          BR          25$              ;GO TO 25$ IF NO ERROR
013006 104401 070230      TYPE          ,SCTMSG         ;TYPE BAD SECTOR MESSAGE
013012 104000          EMT
013014 000137 014062      JMP          350$            ;ERROR # DEFINED BY BADSCT SUBROUTINE
                                ;GO TO 350$ IF ERROR
400 013020          25$:
401 013020 012737 071730 001174      MOV          #ZEROS,$TMP0      ;STARTING ADDRESS OF PATTERN
402 013026 012737 000001 001176      MOV          #1,$TMP1         ;RANGE OF PATTERN
403 013034 004737 043344          JSR          PC,GENBUF         ;GO GENERATE BUFFER FOR FORMAT
404
405 013040 012702 001551      MOV          #PUTINX,R2        ;R2 = BYTE ENTRY POSITION
406 013044 112722 000034      MOVB         #RMDC,(R2)+
407 013050 112722 000006      MOVB         #RMDA,(R2)+
408 013054 112722 000004      MOVB         #RMBA,(R2)+
409 013060 112722 000002      MOVB         #RMWC,(R2)+
410 013064 112722 000032      MOVB         #RMOF,(R2)+
411 013070 112722 000000      MOVB         #RMCS1,(R2)+
412 013074 112712 000200      MOVB         #200,(R2)
413 013100          30$:
414 013100 004737 044500      JSR          PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                ;GO TO 40$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY PUT SUBROUTINE
                                ;GO TO 350$ IF ERROR
                                40$:
013104 000404          BR          40$
013106 000240          NOP
013110 104000          EMT
013112 000137 014062      JMP          350$
415 013116          40$:
416
417          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
013116 004737 044144      JSR          PC,GETSTS        ;GO TO GETSTS SUBROUTINE
418
419          ;WAIT FOR COMMAND TO COMPLETE
013122 004737 045042      JSR          PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
420
421          ;GO EGT REGISTER STATUS
422 013126 004737 044230      JSR          PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                ;GO TO 50$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY GET SUBROUTINE
                                ;GO TO 350$ IF ERROR
                                50$:
013132 000404          BR          50$
013134 000240          NOP
013136 104000          EMT
013140 000137 014062      JMP          350$
423 013144          50$:
424
425          ;VERIFY RESULTS OF WRITE COMMAND
426 013144 004737 057742      JSR          PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
                                ;GO TO 60$ IF NO ERROR
                                ;RETURN HERE IF ERROR
013150 000405          BR          60$
013152 000240          NOP

```

```

013154 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
013156 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
013160 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
427 013164 60$:
428
429 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
430 013164 012737 013174 001124 MOV #70$,$LPERR
431 013172 000410 BR 80$ ;SKIP TO WRITE OPERATION
432
433 ;*****
434 ;LOOP #2 WRITE,WRITE CHECK DATA
435
436 013174 70$:
437
438 ;PREPARE DEVICE FOR TEST
439 013174 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
013200 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 80$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR
013202 000404 BR 80$
013204 000240 NOP
013206 104000 EMT
013210 000137 014062 JMP 350$
440 013214 80$:
441
442 ;SETUP PARAMETERS AND EXECUTE COMMAND
443 013214 120$:
444 013214 012737 177400 001412 MOV #-256.,RMWCO ;CHANGE WORD COUNT
445 013222 012737 106320 001414 MOV #BUFONE+4,RMBAO ;CHANGE MEMORY ADDRESS
446 013230 012737 000061 001410 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
447 013236 012702 001551 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
448 013242 112722 000006 MOVB #RMDA,(R2)+
449 013246 112722 000034 MOVB #RMDC,(R2)+
450 013252 112722 000032 MOVB #RMOF,(R2)+
451 013256 112722 000004 MOVB #RMBA,(R2)+
452 013262 112722 000002 MOVB #RMWC,(R2)+
453 013266 112722 000000 MOVB #RMCS1,(R2)+
454 013272 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
455
456 013276 004737 044500 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
013302 000404 BR 130$ ;GO TO 130$ IF NO ERROR
013304 000240 NOP ;RETURN HERE IF ERROR
013306 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
013310 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
457 013314 130$:
458
459 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
013314 004737 044144 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
460
461 ;WAIT FOR COMMAND TO COMPLETE
013320 004737 045042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
462

```

```

463 ;GO READ STATUS FOR WRITE COMMAND
464 013324 004737 044230 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    013330 000404 BR 140$ ;GO TO 140$ IF NO ERROR
    013332 000240 NOP ;RETURN HERE IF ERROR
    013334 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    013336 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
465 013342 140$:
466
467 ;CHECK FOR ERRORS DURING WRITE OPERATION
468 013342 004737 045226 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    013346 000405 BR 150$ ;GO TO 150$ IF NO ERROR
    013350 000240 NOP ;RETURN HERE IF ERROR
    013352 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    013354 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    013356 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
469 013362 150$:
470 013362 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    013366 000405 BR 160$ ;GO TO 160$ IF NO ERROR
    013370 000240 NOP ;RETURN HERE IF ERROR
    013372 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
    013374 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    013376 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
471 013402 160$:
472 013402 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    013406 000405 BR 180$ ;GO TO 180$ IF NO ERROR
    013410 000240 NOP ;RETURN HERE IF ERROR
    013412 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    013414 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    013416 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
473 013422 180$:
474
475 ;CHANGE LOOP ADDRESSES
476 013422 012737 013430 001124 MOV #190$,$LPERR
477
478 ;*****
479 ;LOOP #3 WRITE CHECK DATA
480
481 013430 190$:
482
483 013430 012703 000001 MOV #1,R3 ;R3+WCE BIT POSITION
484 013434 050337 107316 BIS R3,BUFTWO-2 ;CHANGE LAST WORD OF BUFFER
485
486 ;PREPARE DEVICE FOR TEST
487 013440 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    013444 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
    013446 000404 BR 200$ ;GO TO 200$ IF NO ERROR
    013450 000240 NOP ;RETURN HERE IF ERROR
    013452 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    013454 000137 014062 JMP 350$ ;GO TO 350$ IF ERROR
488 013460 200$:

```

```

489
490          :READ DATA FROM DEVICE
491 013460          240$:
492 013460 012737 000051 001410      MOV    #WCD!GO, RMCS10      ;WRITE CHECK DATA DATA COMMAND
493 013466 012702 001551              MOV    #PUTINX, R2          ;LOAD PUT REGISTER INDEX TABLE
494 013472 112722 000006              MOVB  #RMDA, (R2)+
495 013476 112722 000032              MOVB  #RMOF, (R2)+
496 013502 112722 000034              MOVB  #RMDC, (R2)+
497 013506 112722 000004              MOVB  #RMB A, (R2)+
498 013512 112722 000002              MOVB  #RMWC, (R2)+
499 013516 112722 000000              MOVB  #RMCS1, (R2)+
500 013522 112712 000200              MOVB  #200, (R2)
501
502 013526 004737 044500              JSR   PC, PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      013532 000404              BR    250$           ;GO TO 250$ IF NO ERROR
      013534 000240              NOP
      013536 104000              EMT          ;RETURN HERE IF ERROR
      013540 000137 014062              JMP   350$           ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 350$ IF ERROR
503 013544          250$:
504
505          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      013544 004737 044144              JSR   PC, GETSTS      ;GO TO GETSTS SUBROUTINE
506
507          ;WAIT FOR COMMAND TO COMPLETE
      013550 004737 045042              JSR   PC, TIMEOUT     ;GO TO TIMEOUT SUBROUTINE
508
509          ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
510 013554 004737 044230              JSR   PC, GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      013560 000404              BR    260$           ;GO TO 260$ IF NO ERROR
      013562 000240              NOP
      013564 104000              EMT          ;RETURN HERE IF ERROR
      013566 000137 014062              JMP   350$           ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 350$ IF ERROR
511 013572          260$:
512
513          ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
514 013572 004737 045226              JSR   PC, PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      013576 000405              BR    270$           ;GO TO 270$ IF NO ERROR
      013600 000240              NOP
      013602 104000              EMT          ;RETURN HERE IF ERROR
      013604 004736              JSR   PC, @(SP)+     ;ERROR # DEFINED BY PRIERR SUBROUTINE
      013606 000137 014062              JMP   350$           ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
515 013612          270$:
516 013612 032737 040000 001344      BIT   #WCE, RMCS2I    ;WAS 'WCE' DETECTED??
517 013620 001030              BNE   285$           ;YES!!
518
519 013622 004737 057742              JSR   PC, DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      013626 000405              BR    280$           ;GO TO 280$ IF NO ERROR
      013630 000240              NOP
      013632 104000              EMT          ;RETURN HERE IF ERROR
      013634 004736              JSR   PC, @(SP)+     ;ERROR # DEFINED BY DTASTS SUBROUTINE
      013636 000137 014062              JMP   350$           ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
520 013642          280$:
521 013642 013737 001344 001140      MOV   RMCS2I, $GDDAT  ;EXPECTED STATUS
522 013650 052737 040000 001140      BIS   #WCE, $GDDAT
523 013656 013737 001344 001142      MOV   RMCS2I, $BDDAT ;RECEIVED STATUS
524 013664 010337 001174              MOV   R3, $TMP0      ;FAILING BIT POSITION
525 013670 012737 107316 001176      MOV   #BUFTWO-2, $TMP1 ;FAILING ADDRESS
  
```



```

526 013676 104337          EMT      337
527 013700 000470          BR      350$
528 013702                285$:
529 013702 112737 000022 001522  MOVB   #RMDB,GETINX ;SETUP GET INDEX TABLE
    013710 112737 000200 001523  MOVB   #200,GETINX+1 ;SETUP TERMINATOR BYTE
    013716 004737 044230          JSR    PC,GET        ;GO READ RMDB VIA GET SUBROUTINE
    013722 000404          BR      290$         ;GO TO 290$ IF NO ERROR
    013724 000240          NOP
    013726 104000          EMT
    013730 000137 014062          JMP    350$         ;RETURN HERE IF ERROR
                                ;ERROR DEFINED BY GET SUBROUTINE
                                ;GO TO 350$ IF ERROR
530 013734                290$:
531 013734 013737 001356 001142  MOV    RMDBI,$BDDAT ;RECEIVED DATA
532 013742 013737 107316 001140  MOV    BUFTWO-2,$GDDAT ;EXPECTED DATA
533 013750 040337 001140          BIC    R3,$GDDAT
534 013754 012737 107316 001134  MOV    #BUFTWO-2,$GDADR ;EXPECTED ADDRESS
535 013762 013737 001340 001136  MOV    RMBAI,$BDADR  ;RECEIVED ADDRESS
536 013770 162737 000002 001136  SUB    #2,$BDADR    ;ADJUST BUS ADDRESS
537 013776 023737 001134 001136  CMP    $GDADR,$BDADR ;ADDRESSES OK??
538 014004 001402          BEQ    295$         ;YES!!
539 014006 104340          EMT      340
540 014010 000424          BR      350$
541 014012 023737 001140 001142 295$: CMP    $GDDAT,$BDDAT ;DATA OK??
542 014020 001402          BEQ    296$         ;YES!!
543 014022 104341          EMT      341
544 014024 000416          BR      350$
545 014026                296$:
546
547 014026 004737 046060          JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
    014032 000405          BR      300$         ;GO TO 300$ IF NO ERROR
    014034 000240          NOP        ;RETURN HERE IF ERROR
    014036 104000          EMT        ;ERROR # DEFINED BY SECERR SUBROUTINE
    014040 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
    014042 000137 014062          JMP    350$         ;GO TO 350$ IF ERROR
548 014046                300$:
549 014046 040337 107316          BIC    R3,BUFTWO-2 ;RESTORE DATA PATTERN
550 014052 006303          ASL    R3          ;SHIFT TO NEXT BIT
551 014054 001402          BEQ    350$         ;EXIT IF DONE
552 014056 000137 013434          JMP    195$        ;REPEAT TEST FOR NEXT DATA BIT
553 014062                350$:
554
555

```

```

*****
:*TEST 7      WRITE, READ ONES
*****
TST7:

```

```

014062 000004          SCOPE          ;SCOPE CALL
014062 000240          NOP          ;START OF TEST
014066 012706 001100          MOV    #STACK,SP  ;INITIALIZE STACK POINTER
014072 013700 001276          MOV    $BASE,R0   ;R0 = UNIBUS ADDRESS
014076 013701 001464          MOV    TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
014102 012737 000007 001226  MOV    #7,$TESTN  ;:SET TEST NUMBER IN APT MAIL BOX

```

```

556
557 *****
558 :LOOP #1      FORMAT,WRITE,READ
559
560 014110        10$:
561
562 :PREPARE THE DEVICE FOR TEST

```

```

563 014110 004737 040472      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      014114 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
      014116 000404          BR      20$          ;GO TO 20$ IF NO ERROR
      014120 000240          NOP
      014122 104000          EMT
      014124 000137 015070    JMP     350$        ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                           ;GO TO 350$ IF ERROR

564
565      ;LOAD PARAMETERS AND GENERATE DATA BUFFER
566 014130 20$:
567 014130 012737 000000 001444  MOV     #0,RMDCO      ;CYLINDER = 0
568 014136 012737 000000 001416  MOV     #0,RMDAO      ;TRACK = 0, SECTOR = 0
569 014144 012737 106314 001414  MOV     #BUFONE,RMBAO ;BUS ADDRESS
570 014152 012737 177376 001412  MOV     #-258.,RMWCO  ;2 + 256. WORDS (2'S COMP)
571 014160 012737 010000 001442  MOV     #FMT16,RMOFO  ;16 BIT FORMAT
572 014166 012737 000063 001410  MOV     #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
573
574      ;VERIFY THAT SECTOR IS NOT BAD
      014174 004737 041416      JSR     PC,BADSCT    ;CALL BAD SECTOR MODULE
      014200 000405          BR      25$          ;GO TO 25$ IF NO ERROR
      014202 104401 070230    TYPE    ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
      014206 104000          EMT
      014210 000137 015070    JMP     350$        ;ERROR # DEFINED BY BADSCT SUBROUTINE
                                           ;GO TO 350$ IF ERROR

575 014214 25$:
576 014214 012737 071666 001174  MOV     #ONES,$TMP0   ;STARTING ADDRESS OF PATTERN
577 014222 012737 000001 001176  MOV     #1,$TMP1     ;RANGE OF PATTERN
578 014230 004737 043344      JSR     PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
579
580      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
581 014234 012702 001551      MOV     #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
582 014240 112722 000034      MOV     #RMDC,(R2)+
583 014244 112722 000006      MOV     #RMDA,(R2)+
584 014250 112722 000004      MOV     #RMBA,(R2)+
585 014254 112722 000002      MOV     #RMWC,(R2)+
586 014260 112722 000032      MOV     #RMOF,(R2)+
587 014264 112722 000000      MOV     #RMCS1,(R2)+
588 014270 112712 000200      MOV     #200,(R2)   ;TERMINATE TABLE
589 014274 30$:
590
591      ;FORMAT THE DRIVE
592 014274 004737 044500      JSR     PC,PUT       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      014300 000404          BR      40$          ;GO TO 40$ IF NO ERROR
      014302 000240          NOP
      014304 104000          EMT
      014306 000137 015070    JMP     350$        ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY PUT SUBROUTINE
                                           ;GO TO 350$ IF ERROR

593 014312 40$:
594
595      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      014312 004737 044144      JSR     PC,GETSTS   ;GO TO GETSTS SUBROUTINE
596
597      ;WAIT FOR COMMAND TO COMPLETE

```

```

598 014316 004737 045042          JSR    PC,TIMOUT          ;GO TO TIMOUT SUBROUTINE
599                                     ;GO READ STATUS FOR FORMAT OPERATION
600 014322 004737 044230          JSR    PC,GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014326 000404                    BR     50$                 ;GO TO 50$ IF NO ERROR
    014330 000240                    NOP                                ;RETURN HERE IF ERROR
    014332 104000                    EMT                                ;ERROR # DEFINED BY GET SUBROUTINE
    014334 000137 015070          JMP    350$                ;GO TO 350$ IF ERROR
601 014340          50$:
602                                     ;VERIFY NO ERRORS DURING FORMAT
603                                     JSR    PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
604 014340 004737 057742          BR     60$                 ;GO TO 60$ IF NO ERROR
    014344 000405                    NOP                                ;RETURN HERE IF ERROR
    014346 000240                    EMT                                ;ERROR # DEFINED BY DTASTS SUBROUTINE
    014350 104000                    JSR    PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
    014352 004736                    JMP    350$                ;GO TO 350$ IF ERROR
    014354 000137 015070          60$:
605 014360          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
606                                     MOV    #70$,$LPERR
607                                     BR     80$                 ;SKIP TO WRITE OPERATION
608 014360 012737 014370 001124
609 014366 000410
610
611 ::*****
612 ;LOOP #2          WRITE,READ
613
614 014370          70$:
615
616
617 ;PREPARE DEVICE FOR TEST
618 014370 004737 040472          JSR    PC,TSTPRP          ;PREPARE DEVICE FOR TEST
    014374 154130                    .WORD 154130              ;TASK DESCRIPTOR AS FOLLOWS:
    ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
    ;VERIFY RECALIBRATION
    ;GO TO 80$ IF NO ERROR
    ;RETURN HERE IF ERROR
    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    ;GO TO 350$ IF ERROR
    BR     80$
    NOP
    EMT
    JMP    350$
619 014376 000404                    80$:
620 014400 000240
621 014402 104000
622 014404 000137 015070          ;WRITE DATA TO THE DRIVE
623 014410          120$:
624 014410 012737 106320 001414    MOV    #BUFONE+4,RMBAO    ;CHANGE MEMORY ADDRESS
625 014416 012737 177400 001412    MOV    #-256.,RMWCO      ;CHANGE WORD COUNT
626 014424 012737 000061 001410    MOV    #WD!GO,RMCS10     ;WRITE DATA COMMAND
627 014432 012702 001551                    MOV    #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
628 014436 112722 000006    MOVB  #RMDA,(R2)+
629 014442 112722 000034    MOVB  #RMDC,(R2)+
630 014446 112722 000032    MOVB  #RMOF,(R2)+
631 014452 112722 000004    MOVB  #RMBA,(R2)+
632 014456 112722 000002    MOVB  #RMWC,(R2)+
632 014462 112722 000000    MOVB  #RMCS1,(R2)+

```

```

633 014466 112722 000200          MOVB   #200,(R2)+          ;TERMINATE TABLE
634
635 014472 004737 044500          JSR    PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      014476 000404          BR     130$            ;GO TO 130$ IF NO ERROR
      014500 000240          NOP                    ;RETURN HERE IF ERROR
      014502 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      014504 000137 015070      JMP    350$            ;GO TO 350$ IF ERROR
636 014510          130$:
637
638          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
639 014510 004737 044144          JSR    PC,GETSTS       ;GO TO GETSTS SUBROUTINE
640
641          ;WAIT FOR COMMAND TO COMPLETE
642          JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
643 014520 004737 044230          ;GO READ STATUS FOR WRITE COMMAND
      014524 000404          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      014526 000240          BR     140$            ;GO TO 140$ IF NO ERROR
      014530 104000          NOP                    ;RETURN HERE IF ERROR
      014532 000137 015070      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      014536          JMP    350$            ;GO TO 350$ IF ERROR
644 014536          140$:
645
646          ;CHECK FOR ERRORS DURING WRITE OPERATION
647 014536 004737 045226          JSR    PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      014542 000405          BR     150$            ;GO TO 150$ IF NO ERROR
      014544 000240          NOP                    ;RETURN HERE IF ERROR
      014546 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      014550 004736          JSR    PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      014552 000137 015070      JMP    350$            ;GO TO 350$ IF ERROR
648 014556          150$:
649 014556 004737 057742          JSR    PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
      014562 000405          BR     160$            ;GO TO 160$ IF NO ERROR
      014564 000240          NOP                    ;RETURN HERE IF ERROR
      014566 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      014570 004736          JSR    PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      014572 000137 015070      JMP    350$            ;GO TO 350$ IF ERROR
650 014576          160$:
651 014576 004737 046060          JSR    PC,SECERR       ;GO CHECK FOR SECONDARY ERRORS
      014602 000405          BR     180$            ;GO TO 180$ IF NO ERROR
      014604 000240          NOP                    ;RETURN HERE IF ERROR
      014606 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      014610 004736          JSR    PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      014612 000137 015070      JMP    350$            ;GO TO 350$ IF ERROR
652 014616          180$:
653
654          ;CHANGE LOOP ADDRESSES
655 014616 012737 014626 001124      MOV    #190$,$LPERR
656 014624 000410          BR     200$            ;SKIP TO NEXT OPERATION
657
658          ;*****
659          ;LOOP #3          READ
660
661 014626          190$:
662
663          ;PREPARE DEVICE FOR TEST
664 014626 004737 040472          JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
  
```

```
014632 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 200$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR

014634 000404 BR 200$
014636 000240 NOP
014640 104000 EMT
014642 000137 015070 JMP 350$
665 014646 200$:
666
667 ;READ DATA FROM DEVICE
668 014646 240$:
669 014646 012737 107324 001414 MOV #BUFTWO+4, RMBA0 ;CHANGE MEMORY ADDRESS
670 014654 012737 000071 001410 MOV #RD!GO, RMCS10 ;READ DATA COMMAND
671 014662 012702 001551 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
672 014666 112722 000006 MOVB #RMDA, (R2)+
673 014672 112722 000032 MOVB #RMOF, (R2)+
674 014676 112722 000034 MOVB #RMDC, (R2)+
675 014702 112722 000004 MOVB #RMBA, (R2)+
676 014706 112722 000002 MOVB #RMWC, (R2)+
677 014712 112722 000000 MOVB #RMCS1, (R2)+
678 014716 112712 000200 MOVB #200, (R2)
679
680 014722 004737 044500 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
014726 000404 BR 250$ ;GO TO 250$ IF NO ERROR
014730 000240 NOP ;RETURN HERE IF ERROR
014732 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
014734 000137 015070 JMP 350$ ;GO TO 350$ IF ERROR
681 014740 250$:
682
683 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
684 014740 004737 044144 JSR PC, GETSTS ;GO TO GETSTS SUBROUTINE
685
686 ;WAIT FOR COMMAND TO COMPLETE
687 JSR PC, TIMEOUT ;GO TO TIMEOUT SUBROUTINE
688 014750 004737 044230 ;GO READ STATUS FOR READ OPERATION
014754 000404 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
014756 000240 BR 260$ ;GO TO 260$ IF NO ERROR
014760 104000 NOP ;RETURN HERE IF ERROR
014762 000137 015070 EMT ;ERROR # DEFINED BY GET SUBROUTINE
689 014766 260$: JMP 350$ ;GO TO 350$ IF ERROR
690
691 ;CHECK FOR ERRORS DURING READ OPERATION
692 014766 004737 045226 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
014772 000405 BR 270$ ;GO TO 270$ IF NO ERROR
014774 000240 NOP ;RETURN HERE IF ERROR
014776 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
015000 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
015002 000137 015070 JMP 350$ ;GO TO 350$ IF ERROR
693 015006 270$:
694 015006 004737 057742 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
```

```

015012 000405 BR 280$ ;GO TO 280$ IF NO ERROR
015014 000240 NOP ;RETURN HERE IF ERROR
015016 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
015020 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015022 000137 015070 JMP 350$ ;GO TO 350$ IF ERROR
695 015026 280$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
696 015026 004737 046060 BR 300$ ;GO TO 300$ IF NO ERROR
015032 000405 NOP ;RETURN HERE IF ERROR
015034 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
015036 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015040 004736 JMP 350$ ;GO TO 350$ IF ERROR
697 015042 000137 015070
698 015046 004737 043602 300$: JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
015052 106320 .WORD BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
015054 107324 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
015056 000404 BR 310$ ;GO TO 310$ IF NO ERROR
015060 000240 NOP ;RETURN HERE IF ERROR
015062 104000 EMT ;ERROR # DEFINED BY CMPBUF SUBROUTINE
015064 000137 015070 JMP 350$ ;GO TO 350$ IF ERROR
699 015070 310$:
700 350$:
701 015070
702
703

```

```

;*****
;*TEST 10 WRITE, WRITE CHECK ONES
;*****
TST10:

```

```

015070 000004 SCOPE ;SCOPE CALL
015070 000240 NOP ;START OF TEST
015072 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
015100 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
015104 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
015110 012737 000010 001226 MOV #10,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

```

;*****
;LOOP #1 FORMAT,WRITE,WRITE CHECK DATA

```

```

704
705
706
707
708 015116 10$:
709
710 ;PREPARE THE DEVICE FOR TEST
711 015116 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
015122 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 20$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR

```

```

015124 000404 BR 20$
015126 000240 NOP
015130 104000 EMT
015132 000137 016046 JMP 350$

```

```

;LOAD PARAMETERS AND GENERATE DATA BUFFER
20$:

```

```

712
713
714 015136

```

```

715 015136 012737 000000 001444      MOV      #0,RMDCO          ;CYLINDER = 0
716 015144 012737 000000 001416      MOV      #0,RMDAO          ;TRACK = 0, SECTOR = 0
717 015152 012737 106314 001414      MOV      #BUFONE,RMBAO     ;BUS ADDRESS
718 015160 012737 177376 001412      MOV      #-256,RMWCO       ;2 + 256. WORDS (2'S COMP)
719 015166 012737 010000 001442      MOV      #FMT16,RMOFO      ;16 BIT FORMAT
720 015174 012737 000063 001410      MOV      #WH!GO,RMCS10     ;WRITE HEADER AND DATA COMMAND
721
722      ;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
      BR      25$            ;GO TO 25$ IF NO ERROR
      TYPE    ,SCTMSG        ;TYPE BAD SECTOR MESSAGE
      EMT     ;ERROR # DEFINED BY BADSCT SUBROUTINE
      JMP     350$          ;GO TO 350$ IF ERROR
25$:
723 015222      MOV      #ONES,$TMP0      ;STARTING ADDRESS OF PATTERN
724 015222 012737 071666 001174      MOV      #1,$TMP1         ;RANGE OF PATTERN
725 015230 012737 000001 001176      JSR      PC,GENBUF        ;GO GENERATE BUFFER FOR FORMAT
726 015236 004737 043344
727
728      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
      MOV      #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
729 015242 012702 001551      MOV      #RMDC,(R2)+
730 015246 112722 000034      MOV      #RMDA,(R2)+
731 015252 112722 000006      MOV      #RMBA,(R2)+
732 015256 112722 000004      MOV      #RMWC,(R2)+
733 015262 112722 000002      MOV      #RMOF,(R2)+
734 015266 112722 000032      MOV      #RMCS1,(R2)+
735 015272 112722 000000
736 015276 112712 000200      MOV      #200,(R2)       ;TERMINATE TABLE
737 015302
738
739      ;FORMAT THE DRIVE
740 015302 004737 044500      JSR      PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR      40$            ;GO TO 40$ IF NO ERROR
      NOP     ;RETURN HERE IF ERROR
      EMT     ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP     350$          ;GO TO 350$ IF ERROR
40$:
741 015320
742
743      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
744
745      ;WAIT FOR COMMAND TO COMPLETE
      JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
746
747      ;GO READ STATUS FOR FORMAT OPERATION
748 015330 004737 044230      JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR      50$            ;GO TO 50$ IF NO ERROR
      NOP     ;RETURN HERE IF ERROR
      EMT     ;ERROR # DEFINED BY GET SUBROUTINE
      JMP     350$          ;GO TO 350$ IF ERROR
50$:
749 015346
750
751      ;VERIFY NO ERRORS DURING FORMAT
752 015346 004737 057742      JSR      PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      BR      60$            ;GO TO 60$ IF NO ERROR
      NOP     ;RETURN HERE IF ERROR
      EMT     ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
015352 000405
015354 000240
015356 104000
015360 004736
    
```

```

753 015362 000137 016046          JMP      350$          ;GO TO 350$ IF ERROR
754 015366          60$:
755          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
756 015366 012737 015376 001124  MOV      #70$, $LPERR
757 015374 000410          BR       80$          ;SKIP TO WRITE OPERATION
758
759          ;:*****
760          ;LOOP #2      WRITE,WRITE CHECK DATA
761
762 015376          70$:
763
764
765          ;PREPARE DEVICE FOR WRITE OPERATION
766 015376 004737 040472  JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
015402 154130          .WORD   154130    ;TASK DESCRIPTOR AS FOLLOWS:
          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
          ;CLEAR CONTROLLER & SELECT DEVICE
          ;VERIFY CONTROLLER CLEAR OPERATION
          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
          ;VERIFY PACK ACKNOWLEDGE
          ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
          ;VERIFY RECALIBRATION
          ;GO TO 80$ IF NO ERROR
          ;RETURN HERE IF ERROR
          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          ;GO TO 350$ IF ERROR
          BR       80$
          NOP
          EMT
          JMP      350$
767 015404 000404          BR       80$
768 015406 000240          NOP
769 015410 104000          EMT
770 015412 000137 016046  JMP      350$
771 015416          80$:
772 015416          ;WRITE DATA TO THE DRIVE
773 015416 012737 106320 001414  MOV      #BUFONE+4,RMBAD ;CHANGE MEMORY ADDRESS
774 015424 012737 177400 001412  MOV      #-256.,RMWCO   ;CHANGE WORD COUNT
775 015432 012737 000061 001410  MOV      #WD!GO,RMCS10 ;WRITE DATA COMMAND
776 015440 012702 001551          MOV      #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
777 015444 112722 000006          MOV      #RMDA,(R2)+
778 015450 112722 000034          MOV      #RMDC,(R2)+
779 015454 112722 000032          MOV      #RMOF,(R2)+
780 015460 112722 000004          MOV      #RMBA,(R2)+
781 015464 112722 000002          MOV      #RMWC,(R2)+
782 015470 112722 000000          MOV      #RMCS1,(R2)+
783 015474 112722 000200          MOV      #200,(R2)+ ;TERMINATE TABLE
784 015500 004737 044500          JSR      PC,PUT       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
785 015504 000404          BR       130$        ;GO TO 130$ IF NO ERROR
786 015506 000240          NOP          ;RETURN HERE IF ERROR
787 015510 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
788 015512 000137 016046  JMP      350$        ;GO TO 350$ IF ERROR
789 015516          130$:
790          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
          JSR      PC,GETSTS ;GO TO GETSTS SUBROUTINE
          ;WAIT FOR COMMAND TO COMPLETE
          JSR      PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
          ;GO READ STATUS FOR WRITE COMMAND
  
```



```
791 015526 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
    015532 000404      BR     140$        ;GO TO 140$ IF NO ERROR
    015534 000240      NOP                    ;RETURN HERE IF ERROR
    015536 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
    015540 000137 016046      JMP    350$        ;GO TO 350$ IF ERROR
792 015544      140$:
793
794      ;CHECK FOR ERRORS DURING WRITE OPERATION
795 015544 004737 045226      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
    015550 000405      BR     150$        ;GO TO 150$ IF NO ERROR
    015552 000240      NOP                    ;RETURN HERE IF ERROR
    015554 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
    015556 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
    015560 000137 016046      JMP    350$        ;GO TO 350$ IF ERROR
796 015564      150$:
797 015564 004737 057742      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
    015570 000405      BR     160$        ;GO TO 160$ IF NO ERROR
    015572 000240      NOP                    ;RETURN HERE IF ERROR
    015574 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
    015576 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
    015600 000137 016046      JMP    350$        ;GO TO 350$ IF ERROR
798 015604      160$:
799 015604 004737 046060      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
    015610 000405      BR     180$        ;GO TO 180$ IF NO ERROR
    015612 000240      NOP                    ;RETURN HERE IF ERROR
    015614 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
    015616 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
    015620 000137 016046      JMP    350$        ;GO TO 350$ IF ERROR
800 015624      180$:
801
802      ;CHANGE LOOP ADDRESSES
803 015624 012737 015634 001124  MOV    #190$,$LPERR
804 015632 000410      BR     200$        ;SKIP TO NEXT OPERATION
805
806      ;*****
807      ;LOOP #3      WRITE CHECK DATA
808
809 015634      190$:
810
811      ;PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
812 015634 004737 040472      JSR    PC,TSTPRP   ;PREPARE DEVICE FOR TEST
    015640 154130      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
    015642 000404      BR     200$        ;GO TO 200$ IF NO ERROR
    015644 000240      NOP                    ;RETURN HERE IF ERROR
    015646 104000      EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    015650 000137 016046      JMP    350$        ;GO TO 350$ IF ERROR
813 015654      200$:
814
815      ;WRITE CHECK DATA DATA FROM DEVICE
816 015654      240$:
```

```

817 015654 012737 000051 001410      MOV      #WCD!GO,RMCS10      ;WRITE CHECK DATA DATA COMMAND
818 015662 012702 001551      MOV      #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
819 015666 112722 000006      MOVB    #RMDA,(R2)+
820 015672 112722 000032      MOVB    #RMOF,(R2)+
821 015676 112722 000034      MOVB    #RMDC,(R2)+
822 015702 112722 000004      MOVB    #RMEA,(R2)+
823 015706 112722 000002      MOVB    #RMWC,(R2)+
824 015712 112722 000000      MOVB    #RMCS1,(R2)+
825 015716 112712 000200      MOVB    #200,(R2)
826
827 015722 004737 044500      JSR     PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      015726 000404      BR      250$              ;GO TO 250$ IF NO ERROR
      015730 000240      NOP
      015732 104000      EMT
      015734 000137 016046      JMP     350$              ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 350$ IF ERROR
828 015740      250$:
829
830      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR     PC,GETSTS      ;GO TO GETSTS SUBROUTINE
831 015740 004737 044144
832      ;WAIT FOR COMMAND TO COMPLETE
      JSR     PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
833 015744 004737 045042
834      ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
835 015750 004737 044230      JSR     PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      015754 000404      BR      260$              ;GO TO 260$ IF NO ERROR
      015756 000240      NOP
      015760 104000      EMT
      015762 000137 016046      JMP     350$              ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 350$ IF ERROR
836 015766      260$:
837
838      ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
839 015766 004737 045226      JSR     PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      015772 000405      BR      270$              ;GO TO 270$ IF NO ERROR
      015774 000240      NOP
      015776 104000      EMT
      016000 004736      JSR     PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      016002 000137 016046      JMP     350$              ;GO TO 350$ IF ERROR
840 016006      270$:
841 016006 004737 057742      JSR     PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      016012 000405      BR      280$              ;GO TO 280$ IF NO ERROR
      016014 000240      NOP
      016016 104000      EMT
      016020 004736      JSR     PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      016022 000137 016046      JMP     350$              ;GO TO 350$ IF ERROR
842 016026      280$:
843 016026 004737 046060      JSR     PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
      016032 000405      BR      300$              ;GO TO 300$ IF NO ERROR
      016034 000240      NOP
      016036 104000      EMT
      016040 004736      JSR     PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      016042 000137 016046      JMP     350$              ;GO TO 350$ IF ERROR
844 016046      300$:
845
846 016046      350$:
847
848      ;:*****

```

```

; *TEST 11 WRITE, WRITE CHECK ONES W/ WCE ERROR
; *****
TST11:
016046 000004 SCOPE ;SCOPE CALL
016050 000240 NOP ;START OF TEST
016052 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
016056 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
016062 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
016066 012737 000011 001226 MOV #11,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

849
850 ; *****
851 ; LOOP #1 FORMAT,WRITE,WRITE CHECK DATA
852
853 016074 10$:
854
855 ;PREPARE THE DEVICE FOR FORMAT OPERATION
856 016074 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
016100 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 20$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR

016102 000404 BR 20$
016104 000240 NOP
016106 104000 EMT
016110 000137 017244 JMP 350$
857 016114 20$:
858
859 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
860 016114 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
861 016122 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
862 016130 012737 106314 001414 MOV #BUFONE,RMBAO ;BUS ADDRESS
863 016136 012737 177376 001412 MOV #-258.,RMWCO ;WORD COUNT = 1 SECTOR
864 016144 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
865 016152 012737 000063 001410 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
866
867 ;VERIFY THAT SECTOR IS NOT BAD
016160 004737 041416 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
016164 000405 BR 25$ ;GO TO 25$ IF NO ERROR
016166 104401 070230 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
016172 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
016174 000137 017244 JMP 350$ ;GO TO 350$ IF ERROR
868 016200 25$:
869 016200 012737 071666 001174 MOV #ONES,$TMP0 ;STARTING ADDRESS OF PATTERN
870 016206 012737 000001 001176 MOV #1,$TMP1 ;RANGE OF PATTERN
871 016214 004737 043344 JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT
872
873 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
874 016220 012702 001551 MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
875 016224 112722 000034 MOVB #RMDC,(R2)+
876 016230 112722 000006 MOVB #RMDA,(R2)+
877 016234 112722 000004 MOVB #RMB A,(R2)+
878 016240 112722 000002 MOVB #RMWC,(R2)+
879 016244 112722 000032 MOVB #RMOF,(R2)+

```

```

880 016250 112722 000000          MOVB  #RMCS1,(R2)+
881 016254 112712 000200          MOVB  #200,(R2)          ;TERMINATE TABLE
882 016260          30$:
883
884          ;FORMAT THE DRIVE
885 016260 004737 044500          JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      016264 000404          BR    40$          ;GO TO 40$ IF NO ERROR
      016266 000240          NOP
      016270 104000          EMT          ;RETURN HERE IF ERROR
      016272 000137 017244          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
886 016276          JMP    350$          ;GO TO 350$ IF ERROR
887
888          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
889 016276 004737 044144          JSR   PC,GETSTS       ;GO TO GETSTS SUBROUTINE
890
891          ;WAIT FOR COMMAND TO COMPLETE
892          JSR   PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
893 016302 004737 045042
894
895          ;GO READ STATUS FOR FORMAT OPERATION
896 016306 004737 044230          JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      016312 000404          BR    50$          ;GO TO 50$ IF NO ERROR
      016314 000240          NOP          ;RETURN HERE IF ERROR
      016316 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
      016320 000137 017244          EMT          ;GO TO 350$ IF ERROR
897 016324          JMP    350$
898
899          ;VERIFY NO ERRORS DURING FORMAT
900 016324 004737 057742          JSR   PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
      016330 000405          BR    60$          ;GO TO 60$ IF NO ERROR
      016332 000240          NOP          ;RETURN HERE IF ERROR
      016334 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
      016336 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      016340 000137 017244          EMT          ;GO TO 350$ IF ERROR
901 016344          JMP    350$
902
903          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
904 016344 012737 016354 001124          MOV   #70$,$LPERR
905 016352 000410          BR    80$          ;SKIP TO WRITE OPERATION
906
907          ;*****
908          ;LOOP #2          WRITE,WRITE CHECK DATA
909
910          ;PREPARE DEVICE FOR WRITE OPERATION
911 016354          JSR   PC,TSTPRP       ;PREPARE DEVICE FOR TEST
      016360 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 80$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      BR    80$
      NOP
      EMT
016362 000404
016364 000240
016366 104000
  
```

```

911 016370 000137 017244      JMP      350$          ;GO TO 350$ IF ERROR
912 016374      80$:
913      ;WRITE DATA TO THE DRIVE
914 016374      120$:
915 016374 012737 177400 001412  MOV      #-256.,RMWCO ;CHANGE WORD COUNT
916 016402 012737 106320 001414  MOV      #BUFONE+4,RMBAO ;CHANGE MEMORY ADDRESS
917 016410 012737 000061 001410  MOV      #WD!GO,RMCS10 ;WRITE DATA COMMAND
918 016416 012702 001551      MOV      #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
919 016422 112722 000006      MOV      #RMDA,(R2)+
920 016426 112722 000034      MOV      #RMDC,(R2)+
921 016432 112722 000032      MOV      #RMOF,(R2)+
922 016436 112722 000004      MOV      #RMBA,(R2)+
923 016442 112722 000002      MOV      #RMWC,(R2)+
924 016446 112722 000000      MOV      #RMCS1,(R2)+
925 016452 112722 000200      MOV      #200,(R2)+ ;TERMINATE TABLE
926
927 016456 004737 044500      JSR      PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
928 016462 000404      BR      130$ ;GO TO 130$ IF NO ERROR
929 016464 000240      NOP ;RETURN HERE IF ERROR
930 016466 104000      EMT ;ERROR # DEFINED BY PUT SUBROUTINE
931 016470 000137 017244      JMP      350$ ;GO TO 350$ IF ERROR
932 016474      130$:
933      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
934      JSR      PC,GETSTS ;GO TO GETSTS SUBROUTINE
935 016474 004737 044144
936 016500 004737 045042      ;WAIT FOR COMMAND TO COMPLETE
937      JSR      PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
938 016500 004737 045042
939 016504 004737 044230      ;GO READ STATUS FOR WRITE COMMAND
940 016510 000404      JSR      PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
941 016512 000240      BR      140$ ;GO TO 140$ IF NO ERROR
942 016514 104000      NOP ;RETURN HERE IF ERROR
943 016516 000137 017244      EMT ;ERROR # DEFINED BY GET SUBROUTINE
944 016522 000137 017244      JMP      350$ ;GO TO 350$ IF ERROR
945 016522      140$:
946 016522 004737 045226      ;CHECK FOR ERRORS DURING WRITE OPERATION
947 016526 000405      JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
948 016530 000240      BR      150$ ;GO TO 150$ IF NO ERROR
949 016532 104000      NOP ;RETURN HERE IF ERROR
950 016534 004736      EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
951 016536 000137 017244      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
952 016542 000137 017244      JMP      350$ ;GO TO 350$ IF ERROR
953 016542      150$:
954 016542 004737 057742      JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
955 016546 000405      BR      160$ ;GO TO 160$ IF NO ERROR
956 016550 000240      NOP ;RETURN HERE IF ERROR
957 016552 104000      EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
958 016554 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
959 016556 000137 017244      JMP      350$ ;GO TO 350$ IF ERROR
960 016562      160$:
961 016562 004737 046060      JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
962 016566 000405      BR      180$ ;GO TO 180$ IF NO ERROR
963 016570 000240      NOP ;RETURN HERE IF ERROR
964 016572 104000      EMT ;ERROR # DEFINED BY SECERR SUBROUTINE

```

```

016574 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
016576 000137 017244   JMP    350$              ;GO TO 350$ IF ERROR
944 016602          180$:
945
946          ;CHANGE LOOP ADDRESSES
947 016602 012737 016610 001124   MOV    #190$,$LPERR
948
949          ;*****
950          ;LOOP #3          WRITE CHECK DATA
951
952 016610          190$:
953
954 016610 012703 000001          MOV    #1,R3              ;R3+WCE BIT POSITION
955 016614 040337 107316          BIC    R3,BUFTWO-2       ;CHANGE LAST WORD OF BUFFER
956
957          ;PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
958 016620 004737 040472          JSR    PC,TSTPRP        ;PREPARE DEVICE FOR TEST
016624 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 200$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 350$ IF ERROR
016626 000404          BR     200$
016630 000240          NOP
016632 104000          EMT
016634 000137 017244          JMP    350$
959 016640          200$:
960
961          ;READ DATA FROM DEVICE
962 016640          240$:
963 016640 012737 000051 001410   MOV    #WCD!GO,RMCS10    ;WRITE CHECK DATA DATA COMMAND
964 016646 012702 001551          MOV    #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
965 016652 112722 000006          MOVB   #RMDA,(R2)+
966 016656 112722 000032          MOVB   #RMOF,(R2)+
967 016662 112722 000034          MOVB   #RMDC,(R2)+
968 016666 112722 000004          MOVB   #RMBA,(R2)+
969 016672 112722 000002          MOVB   #RMWC,(R2)+
970 016676 112722 000000          MOVB   #RMCS1,(R2)+
971 016702 112712 000200          MOVB   #200,(R2)
972
973 016706 004737 044500          JSR    PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
016712 000404          BR     250$             ;GO TO 250$ IF NO ERROR
016714 000240          NOP                    ;RETURN HERE IF ERROR
016716 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
016720 000137 017244          JMP    350$             ;GO TO 350$ IF ERROR
974 016724          250$:
975
976          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
977 016724 004737 044144          JSR    PC,GETSTS        ;GO TO GETSTS SUBROUTINE
978
979          ;WAIT FOR COMMAND TO COMPLETE
016730 004737 045042          JSR    PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
980
;GO READ STATUS FOR WRITE CHECK DATA OPERATION
    
```

```

981 016734 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
    016740 000404      BR     260$       ;GO TO 260$ IF NO ERROR
    016742 000240      NOP                    ;RETURN HERE IF ERROR
    016744 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
    016746 000137 017244      JMP    350$       ;GO TO 350$ IF ERROR
982 016752      260$:
983
984      ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
985 016752 004737 045226      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
    016756 000405      BR     270$       ;GO TO 270$ IF NO ERROR
    016760 000240      NOP                    ;RETURN HERE IF ERROR
    016762 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
    016764 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
    016766 000137 017244      JMP    350$       ;GO TO 350$ IF ERROR
986 016772      270$:
987 016772 032737 040000 001344  BIT    #WCE,RMCS2I  ;WAS 'WCE' DETECTED??
988 017000 001030      BNE    285$       ;YES!!
989 017002 004737 057742      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
    017006 000405      BR     280$       ;GO TO 280$ IF NO ERROR
    017010 000240      NOP                    ;RETURN HERE IF ERROR
    017012 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
    017014 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
    017016 000137 017244      JMP    350$       ;GO TO 350$ IF ERROR
990 017022      280$:
991 017022 013737 001344 001140  MOV    RMCS2I,$GDDAT ;EXPECTED STATUS
992 017030 052737 040000 001140  BIS    #WCE,$GDDAT
993 017036 013737 001344 001142  MOV    RMCS2I,$BDDAT ;RECEIVED STATUS
994 017044 010337 001174      MOV    R3,$TMP0     ;FAILING BIT POSITION
995 017050 012737 107316 001176  MOV    #BUFTWO-2,$TMP1 ;FAILING ADDRESS
996 017056 104337      EMT    337
997 017060 000471      BR     350$
998
999 017062      285$:
1000 017062 112737 000022 001522  MOVB   #RMDB,GETINX  ;SETUP GET INDEX TABLE
    017070 112737 000200 001523  MOVB   #200,GETINX+1 ;SETUP TERMINATOR BYTE
    017076 012737 017244 001356  MOV    #350$,RMDBI  ;SET RMDB INPUT BUFFER = 350$
    017104 004737 044230      JSR    PC,GET      ;GO READ RMDB VIA GET SUBROUTINE
    017110 000402      BR     290$       ;GO TO 290$ IF NO ERROR
    017112 000240      NOP                    ;RETURN HERE IF ERROR
    017114 104000      EMT                    ;ERROR DEFINED BY GET SUBROUTINE
1001
1002 017116      290$:
1003 017116 013737 001356 001142  MOV    RMDBI,$BDDAT ;RECEIVED DATA
1004 017124 013737 107316 001140  MOV    BUFTWO-2,$GDDAT ;EXPECTED DATA
1005 017132 050337 001140      BIS    R3,$GDDAT
1006 017136 012737 107316 001134  MOV    #BUFTWO-2,$GDADR ;EXPECTED ADDRESS
1007 017144 013737 001340 001136  MOV    RMBAI,$BDADR  ;RECEIVED ADDRESS
1008 017152 162737 000002 001136  SUB    #2,$BDADR     ;CORRECT MEMORY ADDRESS
1009 017160 023737 001134 001136  CMP    $GDADR,$BDADR ;ADDRESSES OK??
1010 017166 001402      BEQ    295$       ;YES!!
1011 017170 104340      EMT    340
1012 017172 000424      BR     350$
1013 017174 023737 001140 001142 295$:  CMP    $GDDAT,$BDDAT ;DATA OK??
1014 017202 001402      BEQ    296$       ;YES!!
1015 017204 104341      EMT    341
1016 017206 000416      BR     350$
1017 017210      296$:

```

```

1018
1019 017210 004737 046060      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      017214 000405          BR     300$           ;GO TO 300$ IF NO ERROR
      017216 000240          NOP                    ;RETURN HERE IF ERROR
      017220 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      017222 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      017224 000137 017244    JMP    350$           ;GO TO 350$ IF ERROR
1020 017230          300$:
1021 017230 050337 107316    BIS    R3,BUFTWO-2    ;RESTORE DATA PATTERN
1022 017234 006303          ASL    R3              ;SHIFT TO NEXT BIT
1023 017236 001402          BEQ    350$           ;EXIT IF DONE
1024 017240 000137 016614    JMP    350$           ;REPEAT TEST FOR NEXT DATA BIT
1025 017244          350$:
1026
1027
;*****
;*TEST 12      WRITE, WRITE CHECK MULTIPLE SECTORS
;*****
TST12:
      017244          SCOPE          ;SCOPE CALL
      017244 000004          NOP          ;START OF TEST
      017246 000240          MOV     #STACK,SP    ;INITIALIZE STACK POINTER
      017250 012706 001100    MOV     $BASE,R0     ;R0 = UNIBUS ADDRESS
      017254 013700 001276    MOV     TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
      017260 013701 001464    MOV     #12,$TESTN  ;:SET TEST NUMBER IN APT MAIL BOX
      017264 012737 000012 001226
1028
1029
1030          ;*****
1031          ;LOOP #1      FORMAT,WRITE,WRITE CHECK
1032          ;*****
1033          10$:
1034          ;PREPARE THE DEVICE FOR FORMAT OPERATION
1035 017272 004737 040472    JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      017276 154130          .WORD  154130        ;TASK DESCRIPTOR AS FOLLOWS:
                                                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                                                ;CLEAR CONTROLLER & SELECT DEVICE
                                                                ;VERIFY CONTROLLER CLEAR OPERATION
                                                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                                                ;VERIFY PACK ACKNOWLEDGE
                                                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                                                ;VERIFY RECALIBRATION
                                                                ;GO TO 20$ IF NO ERROR
                                                                ;RETURN HERE IF ERROR
                                                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                                                ;GO TO 350$ IF ERROR
      017300 000404          BR     20$           ;GO TO 20$ IF NO ERROR
      017302 000240          NOP                    ;RETURN HERE IF ERROR
      017304 104000          EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      017306 000137 020250    JMP    350$           ;GO TO 350$ IF ERROR
1036 017312          20$:
1037
1038          ;LOAD PARAMETERS AND GENERATE DATA BUFFER
1039 017312 012737 000000 001444    MOV     #0,RMDCO     ;CYLINDER = 0
1040 017320 012737 000000 001416    MOV     #0,RMDAO     ;TRACK = 0, SECTOR = 0
1041 017326 012737 106314 001414    MOV     #BUFONE,RMBAO ;BUS ADDRESS
1042 017334 012737 176774 001412    MOV     #-258.*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2'S COMP)
1043 017342 012737 010000 001442    MOV     #FMT16,RMOFO ;16 BIT FORMAT
1044 017350 012737 000063 001410    MOV     #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
1045
1046          ;VERIFY THAT SECTOR IS NOT BAD
      017356 004737 041416    JSR    PC,BADSECT   ;CALL BAD SECTOR MODULE
      017362 000405          BR     25$           ;GO TO 25$ IF NO ERROR
  
```



```

017364 104401 070230      TYPE      ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
017370 104000      EMT              ;ERROR # DEFINED BY BADSCT SUBROUTINE
017372 000137 020250      JMP        350$      ;GO TO 350$ IF ERROR
1047 017376      25$:
1048 017376 012737 071730 001174      MOV        #ZEROS,$TMP0      ;STARTING ADDRESS OF PATTERN
1049 017404 012737 000001 001176      MOV        #1,$TMP1          ;RANGE OF PATTERN
1050 017412 004737 043344      JSR        PC,GENBUF        ;GO GENERATE BUFFER FOR FORMAT
1051
1052      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
1053 017416 012702 001551      MOV        #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
1054 017422 112722 000034      MOVB      #RMDC,(R2)+
1055 017426 112722 000006      MOVB      #RMDA,(R2)+
1056 017432 112722 000004      MOVB      #RMBA,(R2)+
1057 017436 112722 000002      MOVB      #RMWC,(R2)+
1058 017442 112722 000032      MOVB      #RMOF,(R2)+
1059 017446 112722 000000      MOVB      #RMCS1,(R2)+
1060 017452 112712 000200      MOVB      #200,(R2)
1061 017456      30$:
1062
1063      ;FORMAT THE DRIVE
1064 017456 004737 044500      JSR        PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
017462 000404      BR        40$            ;GO TO 40$ IF NO ERROR
017464 000240      NOP
017466 104000      EMT
017470 000137 020250      JMP        350$          ;RETURN HERE IF ERROR
1065 017474      40$:
1066
1067      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
017474 004737 044144      JSR        PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1068
1069      ;WAIT FOR COMMAND TO COMPLETE
017500 004737 045042      JSR        PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
1070
1071      ;GO READ STATUS FOR FORMAT OPERATION
1072 017504 004737 044230      JSR        PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
017510 000404      BR        50$            ;GO TO 50$ IF NO ERROR
017512 000240      NOP
017514 104000      EMT
017516 000137 020250      JMP        350$          ;RETURN HERE IF ERROR
1073 017522      50$:
1074
1075      ;VERIFY NO ERRORS DURING FORMAT
1076 017522 004737 057742      JSR        PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
017526 000405      BR        60$            ;GO TO 60$ IF NO ERROR
017530 000240      NOP
017532 104000      EMT
017534 004736      JSR        PC,@(SP)+      ;ERROR # DEFINED BY DTASTS SUBROUTINE
017536 000137 020250      JMP        350$          ;GO BACK FOR MORE ERROR CHECKS
1077 017542      60$:
1078
1079      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
1080 017542 012737 017614 001124      MOV        #70$,$LPERR
1081
1082      ;REGENERATE DATA BUFFER
1083 017550 012737 177000 001412      MOV        #-256.*2,RMWC   ;CHANGE WORD COUNT
1084 017556 012737 106314 001414      MOV        #BUFONE,RMBAO   ;CHANGE BUS ADDRESS
1085 017564 012737 000060 001410      MOV        #WD,RMCS10     ;WRITE DATA
  
```

```

1086 017572 012737 071666 001174      MOV    #ONES,$TMP0      ;STARTING ADDRESS OF PATTERN
1087 017600 012737 000001 001176      MOV    #1,$TMP1        ;RANGE OF PATTERN
1088 017606 004737 043344              JSR    PC,GENBUF
1089 017612 000410              BR     80$              ;SKIP TO WRITE OPERATION
1090
1091      ;:*****
1092      ;:LOOP #2          WRITE,WRITE CHECK
1093
1094 017614      70$:
1095
1096
1097      ;:PREPARE DEVICE FOR WRITE OPERATION
1098 017614 004737 040472      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      017620 154130      .WORD 154130          ;TASK DESCRIPTOR AS FOLLOWS:
      ;:SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;:CLEAR CONTROLLER & SELECT DEVICE
      ;:VERIFY CONTROLLER CLEAR OPERATION
      ;:PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;:VERIFY PACK ACKNOWLEDGE
      ;:RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;:VERIFY RECALIBRATION
      ;:GO TO 80$ IF NO ERROR
      ;:RETURN HERE IF ERROR
      ;:ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;:GO TO 350$ IF ERROR
      017622 000404      BR     80$
      017624 000240      NOP
      017626 104000      EMT
      017630 000137 020250      JMP    350$
1099 017634      80$:
1100
1101      ;:WRITE DATA TO THE DRIVE
1102 017634      120$:
1103 017634 012737 000061 001410      MOV    #WD!GO,RMCS10   ;WRITE DATA COMMAND
1104 017642 012702 001551              MOV    #PUTINX,R2      ;LOAD PUT REGISTER INDEX TABLE
1105 017646 112722 000006              MOV    #RMDA,(R2)+
1106 017652 112722 000034              MOV    #RMDC,(R2)+
1107 017656 112722 000032              MOV    #RMOF,(R2)+
1108 017662 112722 000004              MOV    #RMBA,(R2)+
1109 017666 112722 000002              MOV    #RMWC,(R2)+
1110 017672 112722 000000              MOV    #RMCS1,(R2)+
1111 017676 112722 000200              MOV    #200,(R2)+      ;TERMINATE TABLE
1112
1113 017702 004737 044500      JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      017706 000404      BR     130$           ;GO TO 130$ IF NO ERROR
      017710 000240      NOP
      017712 104000      EMT
      017714 000137 020250      JMP    350$           ;RETURN HERE IF ERROR
      ;:ERROR # DEFINED BY PUT SUBROUTINE
      ;:GO TO 350$ IF ERROR
1114 017720      130$:
1115
1116      ;:SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR    PC,GETSTS    ;GO TO GETSTS SUBROUTINE
1117
1118      ;:WAIT FOR COMMAND TO COMPLETE
      JSR    PC,TIMOUT    ;GO TO TIMOUT SUBROUTINE
1119
1120      ;:GO READ STATUS FOR WRITE COMMAND
1121 017730 004737 044230      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      017734 000404      BR     140$           ;GO TO 140$ IF NO ERROR
      017736 000240      NOP
      017740 104000      EMT
      ;:RETURN HERE IF ERROR
      ;:ERROR # DEFINED BY GET SUBROUTINE
  
```

```

1122 017742 000137 020250          JMP      350$          ;GO TO 350$ IF ERROR
1123 017746          140$:
1124          ;CHECK FOR ERRORS DURING WRITE OPERATION
1125 017746 004737 045226          JSR      PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      017752 000405          BR       150$          ;GO TO 150$ IF NO ERROR
      017754 000240          NOP      ;RETURN HERE IF ERROR
      017756 104000          EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
      017760 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      017762 000137 020250          JMP      350$          ;GO TO 350$ IF ERROR
1126 017766          150$:
1127 017766 004737 057742          JSR      PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
      017772 000405          BR       160$          ;GO TO 160$ IF NO ERROR
      017774 000240          NOP      ;RETURN HERE IF ERROR
      017776 104000          EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
      020000 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020002 000137 020250          JMP      350$          ;GO TO 350$ IF ERROR
1128 020006          160$:
1129 020006 004737 046060          JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
      020012 000405          BR       180$          ;GO TO 180$ IF NO ERROR
      020014 000240          NOP      ;RETURN HERE IF ERROR
      020016 104000          EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
      020020 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020022 000137 020250          JMP      350$          ;GO TO 350$ IF ERROR
1130 020026          180$:
1131
1132          ;CHANGE LOOP ADDRESSES
1133 020026 012737 020036 001124    MOV      #190$,$LPERR
1134 020034 000410          BR       200$          ;SKIP TO NEXT OPERATION
1135
1136          ;:*****
1137          ;LOOP #3          WRITE CHECK
1138
1139 020036          190$:
1140
1141          ;PREPARE DEVICE FOR READ OPERATION
1142 020036 004737 040472          JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      020042 154130          .WORD   154130        ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      ;GO TO 200$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 350$ IF ERROR
      020044 000404          BR       200$
      020046 000240          NOP
      020050 104000          EMT
      020052 000137 020250          JMP      350$
1143 020056          200$:
1144
1145          ;READ DATA FROM DEVICE
1146 020056          240$:
1147 020056 012737 000051 001410    MOV      #WCD!GO,RMCS10 ;READ DATA COMMAND
1148 020064 012702 001551          MOV      #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
1149 020070 112722 000006          MOVB    #RMDA,(R2)+
1150 020074 112722 000032          MOVB    #RMOF,(R2)+
  
```

```

1151 020100 112722 000034      MOVB  #RMDC,(R2)+
1152 020104 112722 000004      MOVB  #RMBA,(R2)+
1153 020110 112722 000002      MOVB  #RMWC,(R2)+
1154 020114 112722 000000      MOVB  #RMCS1,(R2)+
1155 020120 112712 000200      MOVB  #200,(R2)
1156
1157 020124 004737 044500      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020130 000404      BR    250$           ;GO TO 250$ IF NO ERROR
      020132 000240      NOP
      020134 104000      EMT
      020136 000137 020250      JMP   350$           ;ERROR # DEFINED BY PUT SUBROUTINE
                          ;GO TO 350$ IF ERROR
1158 020142      250$:
1159
1160      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      020142 004737 044144      JSR   PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1161
1162      ;WAIT FOR COMMAND TO COMPLETE
      020146 004737 045042      JSR   PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
1163
1164      ;GO READ STATUS FOR READ OPERATION
1165 020152 004737 044230      JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020156 000404      BR    260$           ;GO TO 260$ IF NO ERROR
      020160 000240      NOP
      020162 104000      EMT
      020164 000137 020250      JMP   350$           ;ERROR # DEFINED BY GET SUBROUTINE
                          ;GO TO 350$ IF ERROR
1166 020170      260$:
1167
1168      ;CHECK FOR ERRORS DURING READ OPERATION
1169 020170 004737 045226      JSR   PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      020174 000405      BR    270$           ;GO TO 270$ IF NO ERROR
      020176 000240      NOP
      020200 104000      EMT
      020202 004736      JSR   PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020204 000137 020250      JMP   350$           ;GO TO 350$ IF ERROR
1170 020210      270$:
1171 020210 004737 057742      JSR   PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
      020214 000405      BR    280$           ;GO TO 280$ IF NO ERROR
      020216 000240      NOP
      020220 104000      EMT
      020222 004736      JSR   PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020224 000137 020250      JMP   350$           ;GO TO 350$ IF ERROR
1172 020230      280$:
1173 020230 004737 046060      JSR   PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
      020234 000405      BR    300$           ;GO TO 300$ IF NO ERROR
      020236 000240      NOP
      020240 104000      EMT
      020242 004736      JSR   PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      020244 000137 020250      JMP   350$           ;GO TO 350$ IF ERROR
1174 020250      300$:
1175
1176 020250      350$:
1177
1178      ;*****
      ;*TEST 13      WRITE, READ W/ IMPLIED SEEK
      ;*****
      TST13:      SCOPE          ;SCOPE CALL
  
```

```

020252 000240      NOP      ;START OF TEST
020254 012706 001100  MOV     #STACK,SP ;INITIALIZE STACK POINTER
020260 013700 001276  MOV     $BASE,R0  ;R0 = UNIBUS ADDRESS
020264 013701 001464  MOV     TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
020270 012737 000013 001226  MOV     #13,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

1179
1180      ;:*****
1181      ;:LOOP #1      FORMAT,WRITE,READ
1182
1183 020276      10$:
1184
1185      ;:PREPARE THE DEVICE FOR FORMAT OPERATION
1186 020276 004737 040472  JSR     PC,TSTPRP ;PREPARE DEVICE FOR TEST
020302 154130      .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 20$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 350$ IF ERROR

020304 000404      BR      20$
020306 000240      NOP
020310 104000      EMT
020312 000137 021470  JMP     350$

1187
1188      ;:LOAD PARAMETERS AND GENERATE DATA BUFFER
1189 020316      20$:
1190 020316 012737 000000 001444  MOV     #0,RMDCO ;CYLINDER = 0
1191 020324 012737 000000 001416  MOV     #0,RMDAO ;TRACK = 0, SECTOR = 0
1192 020332 013737 001444 021472  MOV     RMDCO,360$ ;SAVE DESIRED CYLINDER
1193 020340 012737 106314 001414  MOV     #BUFONE,RMBAO ;BUS ADDRESS
1194 020346 012737 177376 001412  MOV     #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
1195 020354 012737 010000 001442  MOV     #FMT16,RMOFO ;16 BIT FORMAT
1196 020362 012737 000063 001410  MOV     #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
1197
1198      ;:VERIFY THAT SECTOR IS NOT BAD
020370 004737 041416  JSR     PC,BADSCT ;CALL BAD SECTOR MODULE
020374 000405      BR      25$ ;GO TO 25$ IF NO ERROR
020376 104401 070230  TYPE    ,SCTMSG ;TYPE BAD SECTOR MESSAGE
020402 104000      EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
020404 000137 021470  JMP     350$ ;GO TO 350$ IF ERROR

1199 020410      25$:
1200 020410 012737 071666 001174  MOV     #ONES,$TMP0 ;STARTING ADDRESS OF PATTERN
1201 020416 012737 000001 001176  MOV     #1,$TMP1 ;RANGE OF PATTERN
1202 020424 004737 043344      JSR     PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT
1203
1204      ;:LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
1205 020430 012702 001551      MOV     #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
1206 020434 112722 000034      MOV     #RMDC,(R2)+
1207 020440 112722 000006      MOV     #RMDA,(R2)+
1208 020444 112722 000004      MOV     #RMBA,(R2)+
1209 020450 112722 000002      MOV     #RMWC,(R2)+
1210 020454 112722 000032      MOV     #RMOF,(R2)+
1211 020460 112722 000000      MOV     #RMCS1,(R2)+
1212 020464 112712 000200      MOV     #200,(R2) ;TERMINATE TABLE
1213 020470      30$:

```

```

1214
1215
1216 020470 004737 044500      ;FORMAT THE DRIVE
      020474 000404          JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020476 000240          BR    40$           ;GO TO 40$ IF NO ERROR
      020500 104000          NOP                    ;RETURN HERE IF ERROR
      020502 000137 021470    EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      020506                    JMP   350$          ;GO TO 350$ IF ERROR
1217 020506                    40$:
1218
1219      020506 004737 044144    ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      020506                    JSR   PC,GETSTS       ;GO TO GETSTS SUBROUTINE
1220
1221      020512 004737 045042    ;WAIT FOR COMMAND TO COMPLETE
      020512                    JSR   PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
1222
1223      020516 004737 044230    ;GO READ STATUS FOR FORMAT OPERATION
      020522 000404          JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020524 000240          BR    50$           ;GO TO 50$ IF NO ERROR
      020526 104000          NOP                    ;RETURN HERE IF ERROR
      020530 000137 021470    EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      020534                    JMP   350$          ;GO TO 350$ IF ERROR
1225 020534                    50$:
1226
1227      020534 004737 057742    ;VERIFY NO ERRORS DURING FORMAT
      020540 000405          JSR   PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
      020542 000240          BR    60$           ;GO TO 60$ IF NO ERROR
      020544 104000          NOP                    ;RETURN HERE IF ERROR
      020546 004736          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      020550 000137 021470    JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      020554                    JMP   350$          ;GO TO 350$ IF ERROR
1229 020554                    60$:
1230
1231      020554 012737 020564 001124 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
      020562 000410          MOV   #70$, $LPERR
      020562                    BR    80$           ;SKIP TO WRITE OPERATION
1232
1233
1234
1235      ::*****
1236      ;LOOP #2          WRITE,READ
1237
1238 020564                    70$:
1239
1240      020564 004737 040472    ;PREPARE DEVICE FOR WRITE OPERATION
      020570 154130          JSR   PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      020570                    .WORD 154130          ;TASK DESCRIPTOR AS FOLLOWS:
      020572 000404          BR    80$           ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      020574 000240          NOP                    ;CLEAR CONTROLLER & SELECT DEVICE
      020576 104000          EMT                    ;VERIFY CONTROLLER CLEAR OPERATION
      020600 000137 021470    JMP   350$          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      020604                    MOV   #822.,RMDCO      ;VERIFY PACK ACKNOWLEDGE
      020604                    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      020604                    ;VERIFY RECALIBRATION
      020604                    ;GO TO 80$ IF NO ERROR
      020604                    ;RETURN HERE IF ERROR
      020604                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      020604                    ;GO TO 350$ IF ERROR
1242 020604                    80$:
1243 020604 012737 001466 001444    MOV   #822.,RMDCO      ;SEEK TO LAST CYLINDER
  
```

```

1244 020612 012737 000005 001410      MOV      #SEEK!GO,RMCS10      ;WRITE SEEK COMMAND
1245
1246 020620 004737 044500      JSR      PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020624 000404      BR      90$                 ;GO TO 90$ IF NO ERROR
      020626 000240      NOP                      ;RETURN HERE IF ERROR
      020630 104000      EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
      020632 000137 021470      JMP      350$              ;GO TO 350$ IF ERROR
1247 020636
90$:
1248
1249      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      020636 004737 044144      JSR      PC,GETSTS          ;GO TO GETSTS SUBROUTINE
1250
1251      ;WAIT FOR COMMAND TO COMPLETE
      020642 004737 045042      JSR      PC,TIMOUT         ;GO TO TIMOUT SUBROUTINE
1252
1253      ;GO GET REGISTER STATUS
1254 020646 004737 044230      JSR      PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020652 000404      BR      100$              ;GO TO 100$ IF NO ERROR
      020654 000240      NOP                      ;RETURN HERE IF ERROR
      020656 104000      EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
      020660 000137 021470      JMP      350$              ;GO TO 350$ IF ERROR
1255 020664
100$:
1256
1257      ;VERIFY RESULTS OF SEEK COMMAND
1258 020664 004737 052344      JSR      PC,SEKSTS         ;GO VERIFY RESULTS OF SEEK OPERATION
      020670 000405      BR      110$              ;GO TO 110$ IF NO ERROR
      020672 000240      NOP                      ;RETURN HERE IF ERROR
      020674 104000      EMT                      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      020676 004736      JSR      PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      020700 000137 021470      JMP      350$              ;GO TO 350$ IF ERROR
1259 020704
110$:
1260
1261      ;SETUP PARAMETERS AND EXECUTE WRITE DATA COMMAND
1262 020704
120$:
1263 020704 013737 021472 001444      MOV      360$,RMDCO        ;RESTORE CYLINDER
1264 020712 012737 177400 001412      MOV      #-256.,RMWCO     ;CHANGE WORD COUNT
1265 020720 012737 106320 001414      MOV      #BUFONE+4,RMBAO  ;CHANGE BUS ADDRESS
1266 020726 012737 000061 001410      MOV      #WD!GO,RMCS10    ;WRITE DATA COMMAND
1267 020734 012702 001551      MOV      #PUTINX,R2       ;LOAD PUT REGISTER INDEX TABLE
1268 020740 112722 000006      MOV      #RMDA,(R2)+
1269 020744 112722 000034      MOV      #RMDC,(R2)+
1270 020750 112722 000032      MOV      #RMOF,(R2)+
1271 020754 112722 000004      MOV      #RMBA,(R2)+
1272 020760 112722 000002      MOV      #RMWC,(R2)+
1273 020764 112722 000000      MOV      #RMCS1,(R2)+
1274 020770 112722 000200      MOV      #200,(R2)+
1275
      ;TERMINATE TABLE
1276 020774 004737 044500      JSR      PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021000 000404      BR      130$              ;GO TO 130$ IF NO ERROR
      021002 000240      NOP                      ;RETURN HERE IF ERROR
      021004 104000      EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
      021006 000137 021470      JMP      350$              ;GO TO 350$ IF ERROR
1277 021012
130$:
1278
1279      ;WAIT FOR COMMAND TO COMPLETE
1280 021012 004737 045042      JSR      PC,TIMOUT         ;GO TO TIMOUT SUBROUTINE
  
```

```

1281 ;GO READ STATUS FOR WRITE COMMAND
1282 021016 004737 044230 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    021022 000404 BR 140$ ;GO TO 140$ IF NO ERROR
    021024 000240 NOP ;RETURN HERE IF ERROR
    021026 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
    021030 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1283 021034 140$:
1284
1285 ;CHECK FOR ERRORS DURING WRITE OPERATION
1286 021034 004737 045226 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    021040 000405 BR 150$ ;GO TO 150$ IF NO ERROR
    021042 000240 NOP ;RETURN HERE IF ERROR
    021044 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    021046 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    021050 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1287 021054 150$:
1288 021054 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    021060 000405 BR 160$ ;GO TO 160$ IF NO ERROR
    021062 000240 NOP ;RETURN HERE IF ERROR
    021064 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
    021066 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    021070 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1289 021074 160$:
1290 021074 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    021100 000405 BR 180$ ;GO TO 180$ IF NO ERROR
    021102 000240 NOP ;RETURN HERE IF ERROR
    021104 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    021106 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    021110 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1291 021114 180$:
1292
1293 ;CHANGE LOOP ADDRESSES
1294 021114 012737 021124 001124 MOV #190$, $LPERR
1295 021122 000410 BR 200$ ;SKIP TO NEXT OPERATION
1296
1297 ;*****
1298 ;LOOP #3 READ
1299
1300 021124 190$:
1301
1302 ;PREPARE DEVICE FOR READ OPERATION
1303 021124 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    021130 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
    ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
    ;VERIFY RECALIBRATION
    021132 000404 BR 200$ ;GO TO 200$ IF NO ERROR
    021134 000240 NOP ;RETURN HERE IF ERROR
    021136 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    021140 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1304 021144 200$:
1305 021144 012737 001466 001444 MOV #822.,RMDCO ;SEEK TO LAST CYLINDER
1306 021152 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;WRITE SEEK COMMAND
  
```



```

1307
1308 021160 004737 044500      JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021164 000404      BR     210$        ;GO TO 210$ IF NO ERROR
      021166 000240      NOP                    ;RETURN HERE IF ERROR
      021170 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      021172 000137 021470      JMP    350$        ;GO TO 350$ IF ERROR
1309 021176
210$:
1310
1311      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      021176 004737 044144      JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
1312
1313      ;WAIT FOR COMMAND TO COMPLETE
      021202 004737 045042      JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
1314
1315      ;GO GET REGISTER STATUS
1316 021206 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021212 000404      BR     220$        ;GO TO 220$ IF NO ERROR
      021214 000240      NOP                    ;RETURN HERE IF ERROR
      021216 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      021220 000137 021470      JMP    350$        ;GO TO 350$ IF ERROR
1317 021224
220$:
1318
1319      ;VERIFY RESULTS OF SEEK COMMAND
1320 021224 004737 052344      JSR    PC,SEKSTS   ;GO VERIFY RESULTS OF SEEK OPERATION
      021230 000405      BR     230$        ;GO TO 230$ IF NO ERROR
      021232 000240      NOP                    ;RETURN HERE IF ERROR
      021234 104000      EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      021236 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      021240 000137 021470      JMP    350$        ;GO TO 350$ IF ERROR
1321 021244
230$:
1322
1323      ;SETUP PARAMETERS AND EXECUTE READ DATA COMMAND
1324 021244
240$:
1325 021244 013737 021472 001444      MOV    360$,RMDCO   ;RESTORE CYLINDER
1326 021252 012737 000071 001410      MOV    #RD!GO,RMCS10 ;READ DATA COMMAND
1327 021260 012737 107324 001414      MOV    #BUFTWO+4,RMBAO ;LOAD STARTING BUFFER ADDRESS
1328 021266 012702 001551      MOV    #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
1329 021272 112722 000006      MOVB   #RMDA,(R2)+
1330 021276 112722 000032      MOVB   #RMOF,(R2)+
1331 021302 112722 000034      MOVB   #RMDC,(R2)+
1332 021306 112722 000004      MOVB   #RMBA,(R2)+
1333 021312 112722 000002      MOVB   #RMWC,(R2)+
1334 021316 112722 000000      MOVB   #RMCS1,(R2)+
1335 021322 112712 000200      MOVB   #200,(R2)
1336
1337 021326 004737 044500      JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021332 000404      BR     250$        ;GO TO 250$ IF NO ERROR
      021334 000240      NOP                    ;RETURN HERE IF ERROR
      021336 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      021340 000137 021470      JMP    350$        ;GO TO 350$ IF ERROR
1338 021344
250$:
1339
1340      ;WAIT FOR COMMAND TO COMPLETE
      021344 004737 045042      JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
1341
1342      ;GO READ STATUS FOR READ OPERATION
1343 021350 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
  
```

```

021354 000404 BR 260$ ;GO TO 260$ IF NO ERROR
021356 000240 NOP ;RETURN HERE IF ERROR
021360 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
021362 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1344 021366 260$:
1345
1346 ;CHECK FOR ERRORS DURING READ OPERATION
1347 021366 004737 045226 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
021372 000405 BR 270$ ;GO TO 270$ IF NO ERROR
021374 000240 NOP ;RETURN HERE IF ERROR
021376 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
021400 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021402 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1348 021406 270$:
1349 021406 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
021412 000405 BR 280$ ;GO TO 280$ IF NO ERROR
021414 000240 NOP ;RETURN HERE IF ERROR
021416 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
021420 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021422 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1350 021426 280$:
1351 021426 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
021432 000405 BR 300$ ;GO TO 300$ IF NO ERROR
021434 000240 NOP ;RETURN HERE IF ERROR
021436 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
021440 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021442 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1352 021446 300$:
1353 021446 004737 043602 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
021452 106320 .WORD BUFOONE+4 ;STARTING ADDRESS OF WRITE BUFFER
021454 107324 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
021456 000404 BR 310$ ;GO TO 310$ IF NO ERROR
021460 000240 NOP ;RETURN HERE IF ERROR
021462 104000 EMT ;ERROR # DEFINED BY CMPBUF SUBROUTINE
021464 000137 021470 JMP 350$ ;GO TO 350$ IF ERROR
1354 021470 310$:
1355
1356 021470 350$:
1357 021470 000401 BR 370$
1358
1359 021472 000000 360$: .WORD 0
1360
1361 021474 370$:
1362
1363

```

```

*****
*TEST 14 WRITE, WRITE CHECK W/ HEAD SWITCHING
*****
TST14:

```

```

021474 000004 SCOPE ;SCOPE CALL
021476 000240 NOP ;START OF TEST
021500 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
021504 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
021510 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
021514 012737 000014 001226 MOV #14,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
1364
1365
1366 ;*****
;LOOP #1 FORMAT,WRITE,WRITE CHECK

```

```

1367
1368
1369 021522
1370 021522 012737 000000 001444
1371 021530 112737 000000 001417
1372 021536 112737 000037 001416
1373 021544 012737 010000 001442
1374 021552 012737 106314 001414
1375 021560 012737 176774 001412
1376 021566 012737 000063 001410
1377
1378
    021574 004737 041416
    021600 000405
    021602 104401 070230
    021606 104000
    021610 000137 022702
1379 021614
1380 021614 123727 001416 000037
1381 021622 001345
1382 021624 012737 071666 001174
1383 021632 012737 000001 001176
1384 021640 004737 043344
1385
1386
1387 021644 004737 040472
    021650 154130
    021652 000404
    021654 000240
    021656 104000
    021660 000137 022702
1388 021664
1389
1390
1391 021664 012702 001551
1392 021670 112722 000034
1393 021674 112722 000006
1394 021700 112722 000004
1395 021704 112722 000002
1396 021710 112722 000032
1397 021714 112722 000000
1398 021720 112712 000200
1399
1400
1401 021724 004737 044500
    021730 000404
    021732 000240
    021734 104000
    021736 000137 022702
1402 021742

```

```

;LOAD PARAMETERS AND GENERATE DATA BUFFER
10$:
    MOV     #0,RMDCO           ;CYLINDER = 0
    MOV     #0,RMDAO+1        ;TRACK = 0
    MOV     #31,RMDAO         ;SECTOR = 31.
    MOV     #FMT16,RMOFO      ;16 BIT FORMAT
    MOV     #BUFONE,RMBAO     ;BUS ADDRESS
    MOV     #-258.*2,RMWCO    ;WORD COUNT FOR 2 SECTORS (2'S COMP)
    MOV     #WH!GO,RMCS10     ;WRITE HEADER AND DATA COMMAND

;VERIFY THAT SECTOR IS NOT BAD
    JSR     PC,BADSCT         ;CALL BAD SECTOR MODULE
    BR      20$               ;GO TO 20$ IF NO ERROR
    TYPE    ,SCTMSG           ;TYPE BAD SECTOR MESSAGE
    EMT     ;ERROR # DEFINED BY BADSCT SUBROUTINE
    JMP     350$              ;GO TO 350$ IF ERROR

20$:
    CMPB    RMDAO,#31.        ;IS LAST TRACK ASSIGNED ?
    BNE     15$               ;BR IF NO
    MOV     #ONES,$TMPO       ;STARTING ADDRESS OF PATTERN
    MOV     #1,$TMP1          ;RANGE OF PATTERN
    JSR     PC,GENBUF         ;GO GENERATE BUFFER FOR FORMAT

;PREPARE THE DEVICE FOR FORMAT OPERATION
    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      30$               ;GO TO 30$ IF NO ERROR
    NOP     ;RETURN HERE IF ERROR
    EMT     ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    JMP     350$              ;GO TO 350$ IF ERROR

30$:
;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
    MOV     #PUTINX,R2        ;R2 = BYTE ENTRY POSITION
    MOV     #RMDC,(R2)+
    MOV     #RMDA,(R2)+
    MOV     #RMBA,(R2)+
    MOV     #RMWC,(R2)+
    MOV     #RMOF,(R2)+
    MOV     #RMCS1,(R2)+
    MOV     #200,(R2)        ;TERMINATE TABLE

;FORMAT THE DRIVE
    JSR     PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    BR      40$             ;GO TO 40$ IF NO ERROR
    NOP     ;RETURN HERE IF ERROR
    EMT     ;ERROR # DEFINED BY PUT SUBROUTINE
    JMP     350$            ;GO TO 350$ IF ERROR

40$:

```

```

1403
1404      021742 004737 044144      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR     PC,GETSTS           ;GO TO GETSTS SUBROUTINE
1405
1406      021746 004737 045042      ;WAIT FOR COMMAND TO COMPLETE
      JSR     PC,TIMOUT          ;GO TO TIMOUT SUBROUTINE
1407
1408      ;GO READ STATUS FOR FORMAT OPERATION
1409      021752 004737 044230      JSR     PC,GET                 ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021756 000404                    BR     50$                     ;GO TO 50$ IF NO ERROR
      021760 000240                    NOP                                ;RETURN HERE IF ERROR
      021762 104000                    EMT                                ;ERROR # DEFINED BY GET SUBROUTINE
      021764 000137 022702          JMP     350$                   ;GO TO 350$ IF ERROR
1410      021770      50$:
1411
1412      ;VERIFY NO ERRORS DURING FORMAT
1413      021770 004737 057742      JSR     PC,DTASTS             ;GO VERIFY RESULTS OF DATA TRANSFER
      021774 000405                    BR     60$                     ;GO TO 60$ IF NO ERROR
      021776 000240                    NOP                                ;RETURN HERE IF ERROR
      022000 104000                    EMT                                ;ERROR # DEFINED BY DTASTS SUBROUTINE
      022002 004736                    JSR     PC,@(SP)+              ;GO BACK FOR MORE ERROR CHECKS
      022004 000137 022702          JMP     350$                   ;GO TO 350$ IF ERROR
1414      022010      60$:
1415
1416      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
1417      022010 012737 022062 001124  MOV     #70$,$LPERR
1418
1419      ;REGENERATE BUFFER
1420      022016 012737 177000 001412  MOV     #-256.*2,RMWCO         ;CHANGE WORD COUNT
1421      022024 012737 106314 001414  MOV     #BUFONE,RMBAO         ;CHANGE BUS ADDRESS
1422      022032 012737 071730 001174  MOV     #ZEROS,$TMPO          ;STARTING ADDRESS
1423      022040 012737 000001 001176  MOV     #1,$TMP1              ;RANGE
1424      022046 012737 000060 001410  MOV     #WD,RMCS10            ;WRITE DATA
1425      022054 004737 043344      JSR     PC,GENBUF             ;GENERATE BUFFER
1426      022060 000410                    BR     80$                     ;SKIP TO WRITE OPERATION
1427
1428      ;*****
1429      ;LOOP #2      WRITE,WRITE CHECK
1430
1431      022062      70$:
1432
1433      ;PREPARE DEVICE FOR WRITE OPERATION
1434      022062 004737 040472      JSR     PC,TSTPRP             ;PREPARE DEVICE FOR TEST
      022066 154130                    .WORD 154130                   ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      022070 000404                    BR     80$                     ;GO TO 80$ IF NO ERROR
      022072 000240                    NOP                                ;RETURN HERE IF ERROR
      022074 104000                    EMT                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      022076 000137 022702          JMP     350$                   ;GO TO 350$ IF ERROR
1435      022102      80$:
1436      022102 012737 000005 001410  MOV     #SEEK!GO,RMCS10        ;LOAD SEEK COMMAND
    
```

```

1437
1438 022110 004737 044500          JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      022114 000404                BR     90$             ;GO TO 90$ IF NO ERROR
      022116 000240                NOP                    ;RETURN HERE IF ERROR
      022120 104000                EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      022122 000137 022702          JMP    350$           ;GO TO 350$ IF ERROR
1439 022126                90$:
1440
1441                ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      022126 004737 044144          JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1442
1443                ;WAIT FOR COMMAND TO COMPLETE
      022132 004737 045042          JSR    PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
1444
1445                ;GO GET REGISTER STATUS
1446 022136 004737 044230          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      022142 000404                BR     100$           ;GO TO 100$ IF NO ERROR
      022144 000240                NOP                    ;RETURN HERE IF ERROR
      022146 104000                EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      022150 000137 022702          JMP    350$           ;GO TO 350$ IF ERROR
1447 022154                100$:
1448
1449                ;VERIFY RESULTS OF SEEK COMMAND
1450 022154 004737 052344          JSR    PC,SEKSTS      ;GO VERIFY RESULTS OF SEEK OPERATION
      022160 000405                BR     110$           ;GO TO 110$ IF NO ERROR
      022162 000240                NOP                    ;RETURN HERE IF ERROR
      022164 104000                EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      022166 004736                JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      022170 000137 022702          JMP    350$           ;GO TO 350$ IF ERROR
1451 022174                110$:
1452
1453                ;WRITE DATA TO THE DRIVE
1454 022174                120$:
1455 022174 012737 000061 001410    MOV    #WD!GO, RMCS10 ;WRITE DATA COMMAND
1456 022202 012702 001551                MOV    #PUTINX,R2     ;LOAD PUT REGISTER INDEX TABLE
1457 022206 112722 000006                MOVB   #RMDA,(R2)+
1458 022212 112722 000034                MOVB   #RMDC,(R2)+
1459 022216 112722 000032                MOVB   #RMOF,(R2)+
1460 022222 112722 000004                MOVB   #RMBA,(R2)+
1461 022226 112722 000002                MOVB   #RMWC,(R2)+
1462 022232 112722 000000                MOVB   #RMCS1,(R2)+
1463 022236 112722 000200                MOVB   #200,(R2)+    ;TERMINATE TABLE
1464
1465 022242 004737 044500          JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      022246 000404                BR     130$           ;GO TO 130$ IF NO ERROR
      022250 000240                NOP                    ;RETURN HERE IF ERROR
      022252 104000                EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      022254 000137 022702          JMP    350$           ;GO TO 350$ IF ERROR
1466 022260                130$:
1467
1468                ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1469 022260 004737 044144          JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1470
1471                ;WAIT FOR COMMAND TO COMPLETE
      022264 004737 045042          JSR    PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
1472                ;GO READ STATUS FOR WRITE COMMAND

```

```

1473 022270 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      022274 000404              BR     140$        ;GO TO 140$ IF NO ERROR
      022276 000240              NOP                    ;RETURN HERE IF ERROR
      022300 104000              EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      022302 000137 022702      JMP     350$        ;GO TO 350$ IF ERROR
1474 022306                      140$:
1475
1476
1477 022306 004737 045226      ;CHECK FOR ERRORS DURING WRITE OPERATION
      022312 000405              JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      022314 000240              BR     150$        ;GO TO 150$ IF NO ERROR
      022316 104000              NOP                    ;RETURN HERE IF ERROR
      022320 004736              EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      022322 000137 022702      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      JMP     350$        ;GO TO 350$ IF ERROR
1478 022326                      150$:
1479 022326 004737 057742      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      022332 000405              BR     160$        ;GO TO 160$ IF NO ERROR
      022334 000240              NOP                    ;RETURN HERE IF ERROR
      022336 104000              EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      022340 004736              JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      022342 000137 022702      JMP     350$        ;GO TO 350$ IF ERROR
1480 022346                      160$:
1481 022346 004737 046060      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      022352 000405              BR     180$        ;GO TO 180$ IF NO ERROR
      022354 000240              NOP                    ;RETURN HERE IF ERROR
      022356 104000              EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      022360 004736              JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      022362 000137 022702      JMP     350$        ;GO TO 350$ IF ERROR
1482 022366                      180$:
1483
1484
1485 022366 012737 022376 001124 ;CHANGE LOOP ADDRESSES
1486 022374 000410              MOV    #190$,$LPERR
1487
1488
1489
1490
1491 022376                      ;*****
1492
1493
1494 022376 004737 040472      ;LOOP #3      WRITE CHECK
      022402 154130              ;PREPARE DEVICE FOR READ OPERATION
      JSR    PC,TSTPRP ;PREPARE DEVICE FOR TEST
      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      BR     200$        ;GO TO 200$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      JMP     350$        ;GO TO 350$ IF ERROR
1495 022416                      200$:
1496 022416 012737 000005 001410 MOV    #SEEK!GO,RMCS10 ;LOAD SEEK COMMAND
1497
1498 022424 004737 044500      JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE

```

```

022430 000404 BR 210$ ;GO TO 210$ IF NO ERROR
022432 000240 NOP ;RETURN HERE IF ERROR
022434 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
022436 000137 022702 JMP 350$ ;GO TO 350$ IF ERROR
1499 022442 210$:
1500
1501 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
022442 004737 044144 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
1502
1503 ;WAIT FOR COMMAND TO COMPLETE
022446 004737 045042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1504
1505 ;GO READ REGISTER STATUS
1506 022452 004737 044230 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
022456 000404 BR 220$ ;GO TO 220$ IF NO ERROR
022460 000240 NOP ;RETURN HERE IF ERROR
022462 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
022464 000137 022702 JMP 350$ ;GO TO 350$ IF ERROR
1507 022470 220$:
1508
1509 ;GO VERIFY RESULTS OF SEEK
1510 022470 004737 052344 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
022474 000405 BR 230$ ;GO TO 230$ IF NO ERROR
022476 000240 NOP ;RETURN HERE IF ERROR
022500 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
022502 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022504 000137 022702 JMP 350$ ;GO TO 350$ IF ERROR
1511 022510 230$:
1512
1513 ;READ DATA FROM DEVICE
1514 022510 240$:
1515 022510 012737 000051 001410 MOV #WCD!GO,RMCS10 ;READ DATA COMMAND
1516 022516 012702 001551 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
1517 022522 112722 000006 MOVB #RMDA,(R2)+
1518 022526 112722 000032 MOVB #RMOF,(R2)+
1519 022532 112722 000034 MOVB #RMDC,(R2)+
1520 022536 112722 000004 MOVB #RMBA,(R2)+
1521 022542 112722 000002 MOVB #RMWC,(R2)+
1522 022546 112722 000000 MOVB #RMCS1,(R2)+
1523 022552 112712 000200 MOVB #200,(R2)
1524
1525 022556 004737 044500 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
022562 000404 BR 250$ ;GO TO 250$ IF NO ERROR
022564 000240 NOP ;RETURN HERE IF ERROR
022566 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
022570 000137 022702 JMP 350$ ;GO TO 350$ IF ERROR
1526 022574 250$:
1527
1528 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
022574 004737 044144 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
1529
1530 ;WAIT FOR COMMAND TO COMPLETE
022600 004737 045042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1531
1532 ;GO READ STATUS FOR READ OPERATION
1533 022604 004737 044230 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
022610 000404 BR 260$ ;GO TO 260$ IF NO ERROR
  
```

```

022612 000240      NOP      ;RETURN HERE IF ERROR
022614 104000      EMT      ;ERROR # DEFINED BY GET SUBROUTINE
022616 000137 022702  JMP      350$      ;GO TO 350$ IF ERROR
1534 022622      260$:
1535
1536      ;CHECK FOR ERRORS DURING READ OPERATION
1537 022622 004737 045226  JSR      PC,PRIERR  ;GO CHECK FOR PRIMARY ERRORS
022626 000405      BR      270$      ;GO TO 270$ IF NO ERROR
022630 000240      NOP      ;RETURN HERE IF ERROR
022632 104000      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
022634 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022636 000137 022702  JMP      350$      ;GO TO 350$ IF ERROR
1538 022642      270$:
1539 022642 004737 057742  JSR      PC,DTASTS  ;GO VERIFY RESULTS OF DATA TRANSFER
022646 000405      BR      280$      ;GO TO 280$ IF NO ERROR
022650 000240      NOP      ;RETURN HERE IF ERROR
022652 104000      EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
022654 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022656 000137 022702  JMP      350$      ;GO TO 350$ IF ERROR
1540 022662      280$:
1541 022662 004737 046060  JSR      PC,SECERR  ;GO CHECK FOR SECONDARY ERRORS
022666 000405      BR      300$      ;GO TO 300$ IF NO ERROR
022670 000240      NOP      ;RETURN HERE IF ERROR
022672 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
022674 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022676 000137 022702  JMP      350$      ;GO TO 350$ IF ERROR
1542 022702      300$:
1543
1544 022702      350$:
1545
1546
;*****
;*TEST 15      WRITE, WRITE CHECK W/ MID-TRANSFER SEEK
;*****
TST15:
022702      SCOPE      ;SCOPE CALL
022704 000240      NOP      ;START OF TEST
022706 012706 001100  MOV      #STACK,SP  ;INITIALIZE STACK POINTER
022712 013700 001276  MOV      $BASE,R0   ;R0 = UNIBUS ADDRESS
022716 013701 001464  MOV      TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
022722 012737 000015 001226  MOV      #15,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1547
1548
1549      ;*****
1550      ;LOOP #1      FORMAT,WRITE,WRITE CHECK
1551
1552      ;LOAD PARAMETERS AND GENERATE DATA BUFFER
1553 022730      10$:
1554 022730 012737 000000 001444  MOV      #0,RMDCO   ;CYLINDER = 0
1555 022736 013737 001332 001416  MOV      LSTRK,RMDAO ;SET LAST TRACK AND
1556 022744 112737 000037 001416  MOV      #31.,RMDAO ;LAST SECTOR
1557 022752 012737 010000 001442  MOV      #FMT16,RMOFO ;16 BIT FORMAT
1558 022760 012737 106314 001414  MUV     #BUFONE,RMBAO ;BUS ADDRESS
1559 022766 012737 176774 001412  MOV      #-258.*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2' COMP)
1560 022774 012737 000063 001410  MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
1561
023002 004737 041416      ;VERIFY THAT SECTOR IS NOT BAD
023006 000405      JSR      PC,BADSCT  ;CALL BAD SECTOR MODULE
BR      20$      ;GO TO 20$ IF NO ERROR
  
```



```

023010 104401 070230          TYPE      ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
023014 104000          EMT              ;ERROR # DEFINED BY BADSCT SUBROUTINE
023016 000137 023724          JMP          350$          ;GO TO 350$ IF ERROR
1562 023022          20$:
1563 023022 123737 001417 001333  CMPB      RMDAO+1,LSTRK+1      ;IS LAST TRACK ASSIGNED ?
1564 023030 001342          BNE          15$              ;BR IF NO
1565 023032 012737 071730 001174  MOV       #ZEROS,$TMP0        ;STARTING ADDRESS OF PATTERN
1566 023040 012737 000001 001176  MOV       #1,$TMP1           ;RANGE OF PATTERN
1567 023046 004737 043344          JSR        PC,GENBUF         ;GO GENERATE BUFFER FOR FORMAT
1568
1569
1570 023052 004737 040472          ;PREPARE THE DEVICE FOR FORMAT OPERATION
      023056 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
023060 000404          BR          25$              ;GO TO 25$ IF NO ERROR
023062 000240          NOP
023064 104000          EMT
023066 000137 023724          JMP          350$          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                          ;GO TO 350$ IF ERROR
1571 023072          25$:
1572
1573          ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
1574 023072 012702 001551          MOV       #PUTINX,R2          ;R2 = BYTE ENTRY POSITION
1575 023076 112722 000034          MOVB     #RMDC,(R2)+
1576 023102 112722 000006          MOVB     #RMDA,(R2)+
1577 023106 112722 000004          MOVB     #RMEA,(R2)+
1578 023112 112722 000002          MOVB     #RMWC,(R2)+
1579 023116 112722 000032          MOVB     #RMOF,(R2)+
1580 023122 112722 000000          MOVB     #RMCS1,(R2)+
1581 023126 112712 000200          MOVB     #200,(R2)          ;TERMINATE TABLE
1582 023132          30$:
1583 023132 004737 044500          JSR        PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      023136 000404          BR          40$              ;GO TO 40$ IF NO ERROR
      023140 000240          NOP
      023142 104000          EMT
      023144 000137 023724          JMP          350$          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY PUT SUBROUTINE
                          ;GO TO 350$ IF ERROR
1584 023150          40$:
1585
1586          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      023150 004737 044144          JSR        PC,GETSTS        ;GO TO GETSTS SUBROUTINE
1587
1588          ;WAIT FOR COMMAND TO COMPLETE
      023154 004737 045042          JSR        PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
1589
1590          ;GO READ STATUS FOR FORMAT OPERATION
1591 023160 004737 044230          JSR        PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      023164 000404          BR          50$              ;GO TO 50$ IF NO ERROR
      023166 000240          NOP
      023170 104000          EMT
      023172 000137 023724          JMP          350$          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY GET SUBROUTINE
                          ;GO TO 350$ IF ERROR
1592 023176          50$:
1593

```

```

1594
1595 023176 004737 057742      ;VERIFY NO ERRORS DURING FORMAT
      023202 000405           JSR   PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      023204 000240           BR    60$           ;GO TO 60$ IF NO ERROR
      023206 104000           NOP                    ;RETURN HERE IF ERROR
      023210 004736           EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      023212 000137 023724     JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      023216                   JMP   350$          ;GO TO 350$ IF ERROR
1596 023216                   60$:
1597
1598
1599 023216 012737 023270 001124 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
      023216                   MOV   #70$,$LPERR
1600
1601
1602 023224 012737 000060 001410 ;REGENERATE BUFFER
      023224                   MOV   #WD,RMCS10      ;WRITE DATA
      023226 012737 177000 001412  MOV   #-256.*2,RMWC0 ;CHANGE WORD COUNT
      023228 012737 106314 001414  MOV   #BUFONE,RMBAO  ;CHANGE BUS ADDRSS
      023230 012737 071666 001174  MOV   #ONES,$TMPO    ;PATTERN ADDRESS
      023232 012737 000001 001176  MOV   #1,$TMP1       ;PATTERN RANGE
      023234 004737 043344           JSR   PC,GENBUF
      023236 000410           BR    80$           ;SKIP TO WRITE OPERATION
1609
1610
1611
1612
1613 023270
1614
1615
1616 023270 004737 040472      ;PREPARE DEVICE FOR WRITE OPERATION
      023274 154130           JSR   PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      .WORD 154130           .WORD ;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 80$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 350$ IF ERROR
      023276 000404           BR    80$
      023300 000240           NOP
      023302 104000           EMT
      023304 000137 023724     JMP   350$
1617 023310                   80$:
1618
1619
1620 023310                   ;WRITE DATA TO THE DRIVE
      023310                   120$:
      023310 012737 000061 001410  MOV   #WD!GO,RMCS10 ;WRITE DATA COMMAND
      023312 012702 001551           MOV   #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
      023314 112722 000006           MOVB  #RMDA,(R2)+
      023316 112722 000034           MOVB  #RMDC,(R2)+
      023318 112722 000032           MOVB  #RMOF,(R2)+
      023320 112722 000004           MOVB  #RMB A,(R2)+
      023322 112722 000002           MOVB  #RMWC,(R2)+
      023324 112722 000000           MOVB  #RMCS1,(R2)+
      023326 112722 000200           MOVB  #200,(R2)+
      ;TERMINATE TABLE
1629 023352 112722 000200
1630
1631 023356 004737 044500      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      023362 000404           JSR   PC,PUT        ;GO TO 130$ IF NO ERROR
      023364 000240           BR    130$         ;RETURN HERE IF ERROR
      NOP
  
```

```

023366 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
023370 000137 023724  JMP          350$          ;GO TO 350$ IF ERROR
1632 023374          130$:
1633
1634          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
023374 004737 044144  JSR          PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1635
1636          ;WAIT FOR COMMAND TO COMPLETE
023400 004737 045042  JSR          PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
1637
1638          ;GO READ STATUS FOR WRITE COMMAND
1639 023404 004737 044230  JSR          PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
023410 000404          BR          140$          ;GO TO 140$ IF NO ERROR
023412 000240          NOP          ;RETURN HERE IF ERROR
023414 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
023416 000137 023724  JMP          350$          ;GO TO 350$ IF ERROR
1640 023422          140$:
1641
1642          ;CHECK FOR ERRORS DURING WRITE OPERATION
1643 023422 004737 045226  JSR          PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
023426 000405          BR          150$          ;GO TO 150$ IF NO ERROR
023430 000240          NOP          ;RETURN HERE IF ERROR
023432 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
023434 004736          JSR          PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
023436 000137 023724  JMP          350$          ;GO TO 350$ IF ERROR
1644 023442          150$:
1645 023442 004737 057742  JSR          PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
023446 000405          BR          160$          ;GO TO 160$ IF NO ERROR
023450 000240          NOP          ;RETURN HERE IF ERROR
023452 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
023454 004736          JSR          PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
023456 000137 023724  JMP          350$          ;GO TO 350$ IF ERROR
1646 023462          160$:
1647 023462 004737 046060  JSR          PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
023466 000405          BR          180$          ;GO TO 180$ IF NO ERROR
023470 000240          NOP          ;RETURN HERE IF ERROR
023472 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
023474 004736          JSR          PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
023476 000137 023724  JMP          350$          ;GO TO 350$ IF ERROR
1648 023502          180$:
1649
1650          ;CHANGE LOOP ADDRESSES
1651 023502 012737 023512 001124  MOV          #190$,$LPERR
1652 023510 000410          BR          200$          ;SKIP TO NEXT OPERATION
1653
1654          ;*****
1655          ;LOOP #3          WRITE CHECK
1656
1657 023512          190$:
1658
1659          ;PREPARE DEVICE FOR READ OPERATION
1660 023512 004737 040472  JSR          PC,TSTPRP      ;PREPARE DEVICE FOR TEST
023516 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 200$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 350$ IF ERROR
023520 000404 BR 200$
023522 000240 NOP
023524 104000 EMT
023526 000137 023724 JMP 350$
1661 023532 200$:
1662
1663 ;READ DATA FROM DEVICE
1664 023532 240$:
1665 023532 012737 000051 001410 MOV #WCD!GO, RMCS10 ;READ DATA COMMAND
1666 023540 012702 001551 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
1667 023544 112722 000006 MOVB #RMDA, (R2)+
1668 023550 112722 000032 MOVB #RMOF, (R2)+
1669 023554 112722 000034 MOVB #RMDC, (R2)+
1670 023560 112722 000004 MOVB #RMBA, (R2)+
1671 023564 112722 000002 MOVB #RMWC, (R2)+
1672 023570 112722 000000 MOVB #RMCS1, (R2)+
1673 023574 112712 000200 MOVB #200, (R2)
1674
1675 023600 004737 044500 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
023604 000404 BR 250$ ;GO TO 250$ IF NO ERROR
023606 000240 NOP ;RETURN HERE IF ERROR
023610 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
023612 000137 023724 JMP 350$ ;GO TO 350$ IF ERROR
1676 023616 250$:
1677
1678 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1679 023616 004737 044144 JSR PC, GETSTS ;GO TO GETSTS SUBROUTINE
1680
1681 ;WAIT FOR COMMAND TO COMPLETE
1682 023622 004737 045042 JSR PC, TIMEOUT ;GO TO TIMEOUT SUBROUTINE
1683
1683 ;GO READ STATUS FOR READ OPERATION
023626 004737 044230 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
023632 000404 BR 260$ ;GO TO 260$ IF NO ERROR
023634 000240 NOP ;RETURN HERE IF ERROR
023636 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
023640 000137 023724 JMP 350$ ;GO TO 350$ IF ERROR
1684 023644 260$:
1685
1686 ;CHECK FOR ERRORS DURING READ OPERATION
1687 023644 004737 045226 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
023650 000405 BR 270$ ;GO TO 270$ IF NO ERROR
023652 000240 NOP ;RETURN HERE IF ERROR
023654 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
023656 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
023660 000137 023724 JMP 350$ ;GO TO 350$ IF ERROR
1688 023664 270$:
1689 023664 004737 057742 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
023670 000405 BR 280$ ;GO TO 280$ IF NO ERROR
023672 000240 NOP ;RETURN HERE IF ERROR
023674 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
023676 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
023700 000137 023724 JMP 350$ ;GO TO 350$ IF ERROR
1690 023704 280$:
  
```

```

1691 023704 004737 046060      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      023710 000405          BR        300$         ;GO TO 300$ IF NO ERROR
      023712 000240          NOP          ;RETURN HERE IF ERROR
      023714 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
      023716 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      023720 000137 023724      JMP        350$         ;GO TO 350$ IF ERROR
1692 023724      300$:
1693
1694 023724      350$:
1695
1696
      ;*****
      ;*TEST 16      WRITE, READ W/ HCE ERROR
      ;*****
      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

      ;*****
      ;LOOP #1      FORMAT,WRITE,READ
      MOV      #0,R4      ;R4 = 1ST HEADER WORD
      MOV      #1,R3      ;R3 = HCE BIT
10$:
15$:
      ;PREPARE THE DEVICE FOR FORMAT OPERATION
      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        20$      ;GO TO 20$ IF NO ERROR
      NOP          ;RETURN HERE IF ERROR
      EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      JMP        370$     ;GO TO 370$ IF ERROR

      ;LOAD PARAMETERS AND GENERATE DATA BUFFER
20$:
      MOV      #0,RMDCO   ;CYLINDER = 0
      MOV      #0,RMDAO   ;TRACK = 0, SECTOR = 0
      MOV      #BUFONE,RMBAO ;BUS ADDRESS
      MOV      #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
      MOV      #FMT16,RMOFO ;16 BIT FORMAT
      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

      ;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT  ;CALL BAD SECTOR MODULE
      BR        25$      ;GO TO 25$ IF NO ERROR
      TYPE     ,SCTMSG    ;TYPE BAD SECTOR MESSAGE
      EMT          ;ERROR # DEFINED BY BADSCT SUBROUTINE
  
```

```

1718 024062 000137 025420          JMP      350$          ;GO TO 350$ IF ERROR
1719 024066 012737 071730 001174 25$:   MOV      #ZEROS,$TMP0  ;STARTING ADDRESS OF PATTERN
1720 024074 012737 000001 001176  MOV      #1,$TMP1     ;RANGE OF PATTERN
1721 024102 004737 043344          JSR      PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
1722
1723          ;CHANGE HEADER WORD TO FORCE HCE DURING WRITE & READ
1724 024106 030364 106314          BIT      R3,BUFONE(R4) ;SET OR RESET FOR HCE??
1725 024112 001403 26$      BEQ      26$
1726 024114 040364 106314          BIC      R3,BUFONE(R4) ;RESET BIT FOR HCE
1727 024120 000402 27$      BR       27$
1728 024122 050364 106314 26$:   BIS      R3,BUFONE(R4) ;SET FOR HCE
1729
1730          ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
1731 024126 27$:   MOV      #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
1732 024126 012702 001551          MOVB     #RMDC,(R2)+
1733 024132 112722 000034          MOVB     #RMDA,(R2)+
1734 024136 112722 000006          MOVB     #RMBA,(R2)+
1735 024142 112722 000004          MOVB     #RMWC,(R2)+
1736 024146 112722 000002          MOVB     #RMOF,(R2)+
1737 024152 112722 000032          MOVB     #RMCS1,(R2)+
1738 024156 112722 000000          MOVB     #200,(R2)   ;TERMINATE TABLE
1739 024162 112712 000200
1740 024166 30$:   JSR      PC,PUT       ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
1741 024166 004737 044500          BR       40$         ;GO TO 40$ IF NO ERROR
1742 024172 000404          NOP      ;RETURN HERE IF ERROR
1743 024174 000240          EMT     ;ERROR # DEFINED BY PUT SUBROUTINE
1744 024176 104000          JMP      350$       ;GO TO 350$ IF ERROR
1745 024200 000137 025420 40$:
1746 024204          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1747 024204 004737 044144          JSR      PC,GETSTS   ;GO TO GETSTS SUBROUTINE
1748
1749          ;WAIT FOR COMMAND TO COMPLETE
1750 024210 004737 045042          JSR      PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
1751
1752          ;GO READ STATUS FOR FORMAT OPERATION
1753 024214 004737 044230          JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
1754 024220 000404          BR       50$         ;GO TO 50$ IF NO ERROR
1755 024222 000240          NOP      ;RETURN HERE IF ERROR
1756 024224 104000          EMT     ;ERROR # DEFINED BY GET SUBROUTINE
1757 024226 000137 025420          JMP      350$       ;GO TO 350$ IF ERROR
1758 024232 50$:
1759          ;VERIFY NO ERRORS DURING FORMAT
1760 024232 004737 057742          JSR      PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
1761 024236 000405          BR       60$         ;GO TO 60$ IF NO ERROR
1762 024240 000240          NOP      ;RETURN HERE IF ERROR
1763 024242 104000          EMT     ;ERROR # DEFINED BY DTASTS SUBROUTINE
1764 024244 004736          JSR      PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
1765 024246 000137 025420          JMP      350$       ;GO TO 350$ IF ERROR
1766 024252 60$:
1767          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
1768 024252 012737 024262 001124  MOV      #70$,$LPERR
1769 024260 000410          BR       80$         ;SKIP TO WRITE OPERATION

```



```

1794
1795 024430 004737 045226      ;CHECK FOR ERRORS DURING WRITE OPERATION
      024434 000405      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      024436 000240      BR     145$          ;GO TO 145$ IF NO ERROR
      024440 104000      NOP                    ;RETURN HERE IF ERROR
      024442 004736      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      024444 000137 025420    JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      JMP    350$          ;GO TO 350$ IF ERROR

1796
1797
1798 024450      ;DECODE THE TYPE OF ERROR PRODUCED BY THE BIT POSITION OF R3
1799 024450 005704      145$:
1800 024452 001006      TST    R4            ;IS THIS THE FIRST HEADER WORD ?
      BNE    146$          ;NO !!

1801
1802 024454 032703 010000    BIT    #FMT16,R3     ;SHOULD FER BE SET ?
1803 024460 001037      BNE    155$          ;YES !!
1804 024462 032703 140000    BIT    #MSE!USE,R3  ;SHOULD BSE BE SET ?
1805 024466 001061      BNE    165$          ;YES !!
1806
1807
1808 024470      ;VERIFY THAT A HEADER COMPARE ERROR WAS DETECTED
1809 024470 032737 000200 001350 146$:
1810 024476 001102      BIT    #HCE,RMER1I  ;WAS HCE DETECTED ??
      BNE    175$          ;YES !!

1811
1812 024500 004737 057742      JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      024504 000405      BR     150$          ;GO TO 150$ IF NO ERROR
      024506 000240      NOP                    ;RETURN HERE IF ERROR
      024510 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      024512 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      024514 000137 025420    JMP    350$          ;GO TO 350$ IF ERROR

1813 024520
1814 024520 013737 001350 001142 150$:
1815 024526 012737 000200 001140    MOV    RMER1I,$BDDAT ;RECEIVED STATUS
1816 024534 010437 001174      MOV    #HCE,$GDDAT  ;EXPECTED STATUS
1817 024540 001002      MOV    R4,$TMP0     ;R4 = HEADER WORD NUMBER AND
      BNE    151$          ;ADJUST NUMBER.
1818 024542 005237 001174      INC    $TMP0
1819 024546 010337 001176 151$:
1820 024552 104344      MOV    R3,$TMP1     ;R3 = BIT POSIITON
      EMT    344
1821 024554 000137 025420    JMP    350$

1822
1823
1824 024560      ;VERIFY THAT A FORMAT ERROR WAS DETECTED
1825 024560 032737 000020 001350 155$:
1826 024566 001046      BIT    #FER,RMER1I  ;WAS FER DETECTED ??
      BNE    175$          ;YES !!

1827
1828 024570 004737 057742      JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      024574 000405      BR     160$          ;GO TO 160$ IF NO ERROR
      024576 000240      NOP                    ;RETURN HERE IF ERROR
      024600 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      024602 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      024604 000137 025420    JMP    350$          ;GO TO 350$ IF ERROR

1829 024610
1830 024610 012737 000020 001140 160$:
1831 024616 013737 001350 001142    MOV    #FER,$GDDAT  ;EXPECTED STATUS
      MOV    RMER1I,$BDDAT ;RECEIVED STATUS
1832 024624 104343      EMT    343
1833 024626 000137 025420    JMP    350$

1834
1835      ;VERIFY THAT A BAD SECTOR ERROR WAS DETECTED

```



```

1836 024632
1837 024632 032737 100000 001376 165$: BIT #BSE,RMER2I ;WAS BSE DETECTED ??
1838 024640 001021 BNE 175$ ;YES !!
1839
1840 024642 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
024646 000405 BR 170$ ;GO TO 170$ IF NO ERROR
024650 000240 NOP ;RETURN HERE IF ERROR
024652 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
024654 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
024656 000137 025420 JMP 350$ ;GO TO 350$ IF ERROR
1841 024662
1842 024662 013737 001376 001142 170$: MOV RMER2I,$BDDAT ;RECEIVED STATUS
1843 024670 012737 100000 001140 MOV #BSE,$GDDAT ;EXPECTED STAIUS
1844 024676 104345 EMT 345
1845 024700 000137 025420 JMP 350$
1846
1847 ;CHECK FOR OTHER ERRORS
1848 024704 175$:
1849 024704 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
024710 000405 BR 180$ ;GO TO 180$ IF NO ERROR
024712 000240 NOP ;RETURN HERE IF ERROR
024714 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
024716 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
024720 000137 025420 JMP 350$ ;GO TO 350$ IF ERROR
1850
1851
1852 ;CHANGE LOOP ADDRESSES
1853 024724 012737 024732 001124 MOV #190$,$LPERR
1854
1855 ;*****
1856 ;LOOP #3 READ
1857
1858 024732 190$:
1859
1860 ;PREPARE DEVICE FOR READ OPERATION
1861 024732 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
024736 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
024740 000404 BR 200$ ;GO TO 200$ IF NO ERROR
024742 000240 NOP ;RETURN HERE IF ERROR
024744 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
024746 000137 025420 JMP 350$ ;GO TO 350$ IF ERROR
1862 024752 200$:
1863
1864 ;READ DATA FROM DEVICE
1865 024752 240$:
1866 024752 012737 107324 001414 MOV #BUFTWO+4,RMBAO ;CHANGE BUS ADDRESS
1867 024760 012737 177400 001412 MOV #-256.,RMWCO ;CHANGE WORD COUNT
1868 024766 012737 000071 001410 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
1869 024774 012702 001551 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
1870 025000 112722 000006 MOV#B #RMDA,(R2)+
  
```

```

1871 025004 112722 000032      MOVB  #RMOF,(R2)+
1872 025010 112722 000034      MOVB  #RMDC,(R2)+
1873 025014 112722 000004      MOVB  #RMBA,(R2)+
1874 025020 112722 000002      MOVB  #RMWC,(R2)+
1875 025024 112722 000000      MOVB  #RMCS1,(R2)+
1876 025030 112712 000200      MOVB  #200,(R2)
1877
1878 025034 004737 044500      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      025040 000404          BR    250$           ;GO TO 250$ IF NO ERROR
      025042 000240          NOP                    ;RETURN HERE IF ERROR
      025044 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      025046 000137 025420      JMP   350$           ;GO TO 350$ IF ERROR
1879 025052      250$:
1880
1881      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      025052 004737 044144      JSR   PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1882
1883      ;WAIT FOR COMMAND TO COMPLETE
      025056 004737 045042      JSR   PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
1884
1885      ;GO READ STATUS FOR READ OPERATION
1886 025062 004737 044230      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      025066 000404          BR    260$           ;GO TO 260$ IF NO ERROR
      025070 000240          NOP                    ;RETURN HERE IF ERROR
      025072 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      025074 000137 025420      JMP   350$           ;GO TO 350$ IF ERROR
1887 025100      260$:
1888
1889      ;CHECK FOR ERRORS DURING READ OPERATION
1890 025100 004737 045226      JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      025104 000405          BR    265$           ;GO TO 265$ IF NO ERROR
      025106 000240          NOP                    ;RETURN HERE IF ERROR
      025110 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      025112 004736          JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      025114 000137 025420      JMP   350$           ;GO TO 350$ IF ERROR
1891 025120      265$:
1892 025120 005704          TST   R4
1893 025122 001006          BNE   266$           ;IS THIS THE FIRST HEADER WORD ?
1894
1895 025124 032703 010000          BIT   #FMT16,R3      ;SHOULD FER BE SET ?
1896 025130 001037          BNE   275$           ;YES !!
1897 025132 032703 140000          BIT   #MSE!USE,R3   ;SHOULD BSE BE SET ?
1898 025136 001061          BNE   285$           ;YES !!
1899
1900      ;VERIFY THAT A HEADER COMPARE ERROR WAS DETECTED
1901 025140      266$:
1902 025140 032737 000200 001350      BIT   #HCE,RMER1I    ;WAS HCE DETECTED ??
1903 025146 001102          BNE                    ;YES !!
1904
1905 025150 004737 057742      JSR   PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      025154 000405          BR    270$           ;GO TO 270$ IF NO ERROR
      025156 000240          NOP                    ;RETURN HERE IF ERROR
      025160 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      025162 004736          JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      025164 000137 025420      JMP   350$           ;GO TO 350$ IF ERROR
1906 025170      270$:
1907 025170 013737 001350 001142      MOV   RMER1I,$BDDAT ;RECEIVED STATUS
  
```



```

1950 025410 012703 000001          MOV    #1,R3          ;SETUP FOR HCE
1951 025414          310$:    JMP    15$
1952 025414 000137 023762          350$:    BR    370$          ;EXIT IF ERROR
1953 025420          ;*****
1954 025420 000464          ;*REFORMAT SECTOR THAT WAS WRITTEN WITH BAD HEADER
1955          ;*****
1956          355$:
1957          MOV    #-2,RMWCO          ;ONLY TWO HEAD WORDS
1958 025422          MOV    #FMT16,RMOFO        ;ALWAYS IN 16 BITS MODE
1959 025422 012737 177776 001412    MOV    #BUFONE,RMBAO       ;BUFFER ADDRESS,REFORMAT THE SECTOR
1960 025430 012737 010000 001442    MOV    #WH,RMCS10         ;WRITE HEAD AND DATA COMMAND
1961 025436 012737 106314 001414    JSR    PC,GENBUF          ;SET UP THE BUFFER
1962 025444 012737 000062 001410
1963 025452 004737 043344
1964          JSR    PC,TSTPRP        ;PREPARE DEVICE FOR TEST
1965 025456 004737 040472          .WORD 054130            ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 360$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 360$ IF ERROR
                                025464 000404          BR    360$
                                025466 000240          NOP
                                025470 104000          EMT
                                025472 000137 025476          JMP    360$
1966 025476          360$:
1967 025476 012737 000063 001410    MOV    #WH!GO,RMCS10       ;FORMAT THE SECTOR
1968 025504 012702 001551          MOV    #PUTINX,R2          ;SET UP THE REGS
1969 025510 112722 000006          MOVB   #RMDA,(R2)+
1970 025514 112722 000032          MOVB   #RMOF,(R2)+
1971 025520 112722 000034          MOVB   #RMDC,(R2)+
1972 025524 112722 000004          MOVB   #RMBA,(R2)+
1973 025530 112722 000002          MOVB   #RMWC,(R2)+
1974 025534 112722 000000          MOVB   #RMCS1,(R2)+
1975 025540 112722 000200          MOVB   #200,(R2)+
1976 025544 004737 044500          JSR    PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                025550 000404          BR    365$              ;GO TO 365$ IF NO ERROR
                                025552 000240          NOP                    ;RETURN HERE IF ERROR
                                025554 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
                                025556 000137 025572          JMP    370$            ;GO TO 370$ IF ERROR
1977 025562 000240          365$:    NOP
1978 025564 004737 045042          JSR    PC,TIMOUT          ;WAIT FOR FINISH
1979 025570 000240          370$:    NOP
1980 025572
1981
1982          ;*****
          ;*TEST 17 WRITE, READ W/ HCI
          ;*****
          TST17:
          025572          SCOPE          ;SCOPE CALL
          025572 000004          NOP              ;START OF TEST
          025574 000240          MOV    #STACK,SP      ;INITIALIZE STACK POINTER
          025576 012706 001100          MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
          025602 013700 001276          MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
          025606 013701 001464          MOV    #17,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
          025612 012737 000017 001226

```

1983
 1984
 1985
 1986
 1987 025620 012704 000000
 1988 025624 012703 000001
 1989 025630
 1990
 1991
 1992 025630 004737 040472
 025634 154130
 025636 000404
 025640 000240
 025642 104000
 025644 000137 027176
 1993
 1994
 1995 025650
 1996 025650 012737 000000 001444
 1997 025656 012737 000000 001416
 1998 025664 012737 106314 001414
 1999 025672 012737 177376 001412
 2000 025700 012737 010000 001442
 2001 025706 012737 000063 001410
 2002
 2003
 025714 004737 041416
 025720 000405
 025722 104401 070230
 025726 104000
 025730 000137 027024
 2004 025734
 2005 025734 012737 071666 001174
 2006 025742 012737 000001 001176
 2007 025750 004737 043344
 2008
 2009
 2010 025754 030364 106314
 2011 025760 001403
 2012 025762 040364 106314
 2013 025766 000402
 2014 025770 050364 106314
 2015
 2016
 2017 025774
 2018 025774 012702 001551
 2019 026000 112722 000034
 2020 026004 112722 000006
 2021 026010 112722 000004
 2022 026014 112722 000002

```

:*****
:LOOP #1      .FORMAT,WRITE,READ
10$:  MOV      #0,R4      ;HEADER WORD
      MOV      #1,R3      ;HCE BIT
15$:
;PREPARE THE DEVICE FOR FORMAT OPERATION
      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 20$ IF NO ERROR
                          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                          ;GO TO 370$ IF ERROR
      BR       20$
      NOP
      EMT
      JMP      370$
;LOAD PARAMETERS AND GENERATE DATA BUFFER
20$:  MOV      #0,RMDCO     ;CYLINDER = 0
      MOV      #0,RMDAO     ;TRACK = 0, SECTOR = 0
      MOV      #BUFONE,RMBAO ;BUS ADDRESS
      MOV      #-258,RMWCO  ;2 + 256. WORDS (2' COMP)
      MOV      #FMT16,RMOFO ;16 BIT FORMAT
      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT    ;CALL BAD SECTOR MODULE
      BR       25$         ;GO TO 25$ IF NO ERROR
      TYPE     ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
      EMT
      JMP      350$        ;ERROR # DEFINED BY BADSCT SUBROUTINE
                          ;GO TO 350$ IF ERROR
25$:  MOV      #ONES,$TMP0   ;STARTING ADDRESS OF PATTERN
      MOV      #1,$TMP1     ;RANGE OF PATTERN
      JSR      PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
;CHANGE HEADER WORD TO FORCE ERROR DURING WRITE
      BIT      R3,BUFONE(R4) ;SET OR RESET BIT??
      BEQ     27$
      BIC     R3,BUFONE(R4) ;RESET BIT
      BR      28$
27$:  BIS      R3,BUFONE(R4) ;SET BIT
;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
28$:  MOV      #PUTINX,R2    ;R2 = BYTE ENTRY POSITION
      MOVB   #RMDC,(R2)+
      MOVB   #RMDA,(R2)+
      MOVB   #RMBA,(R2)+
      MOVB   #RMWC,(R2)+
  
```

```

2023 026020 112722 000032          MOVB  #RMOF,(R2)+
2024 026024 112722 000000          MOVB  #RMCS1,(R2)+
2025 026030 112712 000200          MOVB  #200,(R2)          ;TERMINATE TABLE
2026 026034
2027
2028
2029 026034 004737 044500          ;FORMAT THE DRIVE
          JSR  PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          BR   40$            ;GO TO 40$ IF NO ERROR
          NOP                    ;RETURN HERE IF ERROR
          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
          JMP  350$           ;GO TO 350$ IF ERROR
2030 026052
2031
2032          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
          JSR  PC,GETSTS       ;GO TO GETSTS SUBROUTINE
2033 026052 004737 044144
2034          ;WAIT FOR COMMAND TO COMPLETE
          JSR  PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
2035 026056 004737 045042
2036          ;GO READ STATUS FOR FORMAT OPERATION
2037 026062 004737 044230          JSR  PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
          BR   50$            ;GO TO 50$ IF NO ERROR
          NOP                    ;RETURN HERE IF ERROR
          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
          JMP  350$           ;GO TO 350$ IF ERROR
2038 026074 000137 027024
2039 026074 000137 027024
2040 026100 004737 057742          ;VERIFY NO ERRORS DURING FORMAT
          JSR  PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
          BR   60$            ;GO TO 60$ IF NO ERROR
          NOP                    ;RETURN HERE IF ERROR
          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
          JSR  PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
          JMP  350$           ;GO TO 350$ IF ERROR
2041 026104 000405
2042 026106 000240
2043 026110 104000
2044 026112 004736
2045 026114 000137 027024
2046 026120 012737 026130 001124          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
          MOV  #70$,$LPERR
          BR   80$            ;SKIP TO WRITE OPERATION
2047
2048          ;*****
2049          ;LOOP #2          WRITE,READ
2050
2051 026130
2052
2053          ;PREPARE DEVICE FOR WRITE OPERATION
2054 026130 004737 040472          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   80$            ;GO TO 80$ IF NO ERROR
          NOP                    ;RETURN HERE IF ERROR
          026136 000404
          026140 000240
  
```

```

026142 104000          EMT
026144 000137 027024  JMP      350$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
2055 026150          80$:
2056
2057          ;WRITE DATA TO THE DRIVE
2058 026150          120$:
2059 026150 012737 012000 001442  MOV      #FMT16!HCI,RMOFO      ;INHIBIT HEADER COMPARE
2060 026156 012737 000061 001410  MOV      #WD!GO,RMCS10        ;WRITE DATA COMMAND
2061 026164 012737 106320 001414  MOV      #BUFONE+4,RMBAO      ;LOAD STARTING BUFFER ADDRESS
2062 026172 012702 001551      MOV      #PUTINX,R2           ;LOAD PUT REGISTER INDEX TABLE
2063 026176 112722 000006      MOV      #RMDA,(R2)+
2064 026202 112722 000034      MOV      #RMDC,(R2)+
2065 026206 112722 000032      MOV      #RMOF,(R2)+
2066 026212 112722 000004      MOV      #RMB A,(R2)+
2067 026216 112722 000002      MOV      #RMCW,(R2)+
2068 026222 112722 000000      MOV      #RMCS1,(R2)+
2069 026226 112722 000200      MOV      #200,(R2)+          ;TERMINATE TABLE
2070
2071 026232 004737 044500      JSR      PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026236 000404          BR      130$                ;GO TO 130$ IF NO ERROR
      026240 000240          NOP
      026242 104000          EMT
      026244 000137 027024  JMP      350$                ;ERROR # DEFINED BY PUT SUBROUTINE
2072 026250          130$:
2073
2074          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      026250 004737 044144      JSR      PC,GETSTS          ;GO TO GETSTS SUBROUTINE
2075
2076          ;WAIT FOR COMMAND TO COMPLETE
      026254 004737 045042      JSR      PC,TIMOUT         ;GO TO TIMOUT SUBROUTINE
2077
2078          ;GO READ STATUS FOR WRITE COMMAND
2079 026260 004737 044230      JSR      PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026264 000404          BR      140$                ;GO TO 140$ IF NO ERROR
      026266 000240          NOP
      026270 104000          EMT
      026272 000137 027024  JMP      350$                ;ERROR # DEFINED BY GET SUBROUTINE
2080 026276          140$:
2081
2082          ;CHECK FOR ERRORS DURING WRITE OPERATION
2083 026276 004737 045226      JSR      PC,PRIERR         ;GO CHECK FOR PRIMARY ERRORS
      026302 000405          BR      150$                ;GO TO 150$ IF NO ERROR
      026304 000240          NOP
      026306 104000          EMT
      026310 004736          JSR      PC,@(SP)+         ;ERROR # DEFINED BY PRIERR SUBROUTINE
      026312 000137 027024  JMP      350$                ;GO BACK FOR MORE ERROR CHECKS
2084 026316 005037 001140      150$: CLR      $GDDAT          ;GO TO 350$ IF ERROR
2085
      ;EXPECTED STATUS
2086 026322 032737 000220 001350  BIT      #HCE!FER,RMER1I      ;ANY ERROR??
2087 026330 001407          BEQ      151$
2088 026332 013737 001350 001142  MOV      RMER1I,$BDDAT        ;RECEIVED STATUS
2089 026340 042737 177557 001142  BIC      #^C<HCE!FER>,$BDDAT ;CLEAR DONT CARES
2090 026346 000412          BR      152$
2091 026350          151$:
2092 026350 013737 001376 001142  MOV      RMER2I,$BDDAT        ;RECEIVED STATUS
2093 026356 032737 100000 001376  BIT      #BSE,RMER2I          ;ANY BAD SECTOR ERROR ?
2094 026364 001406          BEQ      155$                ;NO !!

```

```

2095 026366 042737 077777 001142      BIC      #^CBSE,$BDDAT      ;SAVE BSE FOR ERROR
2096 026374      152$:
2097 026374 104346      EMT      346
2098 026376 000137 027024      JMP      350$
2099 026402      155$:
2100
2101 026402      160$:
2102 026402 004737 057742      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      026406 000405      BR      170$      ;GO TO 170$ IF NO ERROR
      026410 000240      NOP
      026412 104000      EMT      ;RETURN HERE IF ERROR
      026414 004736      JSR      PC,@(SP)+      ;ERROR # DEFINED BY DTASTS SUBROUTINE
      026416 000137 027024      JMP      350$      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
2103 026422      170$:
2104 026422 004737 046060      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      026426 000405      BR      180$      ;GO TO 180$ IF NO ERROR
      026430 000240      NOP      ;RETURN HERE IF ERROR
      026432 104000      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
      026434 004736      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      026436 000137 027024      JMP      350$      ;GO TO 350$ IF ERROR
2105 026442      180$:
2106
2107      ;CHANGE LOOP ADDRESSES
2108 026442 012737 026452 001124      MOV      #190$,$LPERR
2109 026450 000410      BR      200$
2110
2111      ;*****
2112      ;LOOP #3      READ
2113
2114 026452      190$:
2115
2116      ;PREPARE DEVICE FOR READ OPERATION
2117 026452 004737 040472      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      026456 154130      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      ;GO TO 200$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 350$ IF ERROR
      026460 000404      BR      200$
      026462 000240      NOP
      026464 104000      EMT
      026466 000137 027024      JMP      350$
2118 026472      200$:
2119
2120      ;READ DATA FROM DEVICE
2121 026472      240$:
2122 026472 012737 000071 001410      MOV      #RD!GO,RMCS10      ;READ DATA COMMAND
2123 026500 012737 107324 001414      MOV      #BUF TWO+4,RMBAO      ;LOAD STARTING BUFFER ADDRESS
2124 026506 012702 001551      MOV      #PUTINX,R2      ;LOAD PUT REGISTER INDEX TABLE
2125 026512 112722 000006      MOVB     #RMDA,(R2)+
2126 026516 112722 000032      MOVB     #RMOF,(R2)+
2127 026522 112722 000034      MOVB     #RMDC,(R2)+
2128 026526 112722 000004      MOVB     #RMBA,(R2)+
2129 026532 112722 000002      MOVB     #RMWC,(R2)+
  
```



```

2130 026536 112722 000000      MOVB  #RMCS1,(R2)+
2131 026542 112712 000200      MOVB  #200,(R2)
2132
2133 026546 004737 044500      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026552 000404      BR    250$           ;GO TO 250$ IF NO ERROR
      026554 000240      NOP                    ;RETURN HERE IF ERROR
      026556 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      026560 000137 027024      JMP   350$           ;GO TO 350$ IF ERROR
2134 026564      250$:
2135
2136      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      026564 004737 044144      JSR   PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2137
2138      ;WAIT FOR COMMAND TO COMPLETE
      026570 004737 045042      JSR   PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
2139
2140      ;GO READ STATUS FOR READ OPERATION
2141 026574 004737 044230      JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026600 000404      BR    260$           ;GO TO 260$ IF NO ERROR
      026602 000240      NOP                    ;RETURN HERE IF ERROR
      026604 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      026606 000137 027024      JMP   350$           ;GO TO 350$ IF ERROR
2142 026612      260$:
2143
2144      ;CHECK FOR ERRORS DURING READ OPERATION
2145 026612 004737 045226      JSR   PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      026616 000405      BR    270$           ;GO TO 270$ IF NO ERROR
      026620 000240      NOP                    ;RETURN HERE IF ERROR
      026622 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      026624 004736      JSR   PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      026626 000137 027024      JMP   350$           ;GO TO 350$ IF ERROR
2146 026632 005037 001140      270$: CLR  $GDDAT      ;EXPECTED STATUS
2147
2148 026636 032737 000220 001350      BIT   #HCE!FER,RMER1I ;ANY ERROR ?
2149 026644 001407      BEQ   271$           ;NO!!
2150 026646 013737 001350 001142      MOV   RMER1I,$BDDAT  ;RECEIVED STATUS
2151 026654 042737 177557 001142      BIC   #^C<HCE!FER>,$BDDAT ;CLEAR DONT CARES
2152 026662 000412
2153 026664      271$:
2154 026664 013737 001376 001142      MOV   RMER2I,$BDDAT  ;RECEIVED STATUS
2155 026672 032737 100000 001376      BIT   #BSE,RMER2I    ;ANY BAD SECTOR ERROR ?
2156 026700 001406      BEQ   275$           ;NO !!
2157 026702 042737 077777 001142      BIC   #^CBSE,$BDDAT ;CLEAR DONT CARES
2158 026710      272$:
2159 026710 104346      EMT   346
2160 026712 000137 027024      JMP   350$
2161 026716      275$:
2162
2163      280$:
2164 026716 004737 057742      JSR   PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      026722 000405      BR    290$           ;GO TO 290$ IF NO ERROR
      026724 000240      NOP                    ;RETURN HERE IF ERROR
      026726 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      026730 004736      JSR   PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      026732 000137 027024      JMP   350$           ;GO TO 350$ IF ERROR
2165 026736      290$:
2166 026736 004737 046060      JSR   PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
  
```

```

026742 000405 BR 300$ ;GO TO 300$ IF NO ERROR
026744 000240 NOP ;RETURN HERE IF ERROR
026746 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
026750 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
026752 000137 027024 JMP 350$ ;GO TO 350$ IF ERROR
2167 026756 300$:
2168 026756 004737 043602 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
026762 106320 .WORD BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
026764 107324 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
026766 000404 BR 310$ ;GO TO 310$ IF NO ERROR
026770 000240 NOP ;RETURN HERE IF ERROR
026772 104000 EMT ;ERROR # DEFINED BY CMPBUF SUBROUTINE
026774 000137 027024 JMP 350$ ;GO TO 350$ IF ERROR
2169 027000 310$:
2170 027000 006303 ASL R3 ;SHIFT HCE BIT
2171 027002 001006 BNE 320$ ;CONTINUE IF NOT DONE
2172 027004 005704 TST R4 ;SECOND HEADER DONE??
2173 027006 001006 BNE 350$ ;YES!!
2174 027010 012703 000001 MOV #1,R3 ;START WITH BIT 0
2175 027014 012704 000002 MOV #2,R4 ;DO SECOND HEADER WORD
2176 027020 320$:
2177 027020 000137 025630 JMP 15$
2178 027024 350$:
2179 027024 000464 BR 370$ ;EXIT IF ERROR
2180 ;*****
2181 ;*REFORMAT SECTOR THAT WAS WRITTEN WITH BAD HEADER
2182 ;*****
2183 027026 355$:
2184 027026 012737 177776 001412 MOV #-2,RMWC0 ;WORD COUNT
2185 027034 012737 010000 001442 MOV #FMT16,RMOFO ;IN 16 BIT MODE
2186 027042 012737 106314 001414 MOV #BUFOFF,RMBAO ;BUFFER ADDRESS
2187 027050 012737 000062 001410 MOV #WH,RMCS10 ;FORMAT COMMAND
2188 027056 004737 043344 JSR PC,GENBUF ;SET UP BUFFER
2189
2190 027062 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
027066 054130 .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 360$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 360$ IF ERROR
027070 000404 BR 360$
027072 000240 NOP
027074 104000 EMT
027076 000137 027102 JMP 360$
2191 027102 360$:
2192 027102 012737 000063 001410 MOV #WH!GO,RMCS10
2193 027110 012702 001551 MOV #PUTINX,R2 ;TABLE ADDRESS
2194 027114 112722 000006 MOVB #RMDA,(R2)+
2195 027120 112722 000032 MOVB #RMOF,(R2)+
2196 027124 112722 000034 MOVB #RMDC,(R2)+
2197 027130 112722 000004 MOVB #RMBA,(R2)+
2198 027134 112722 000002 MOVB #RMWC,(R2)+
2199 027140 112722 000000 MOVB #RMCS1,(R2)+
2200 027144 112722 000200 MOVB #200,(R2)+
2201 027150 004737 044500 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE

```

```

027154 000404 BR 365$ ;GO TO 365$ IF NO ERROR
027156 000240 NOP ;RETURN HERE IF ERROR
027160 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
027162 000137 027176 JMP 370$ ;GO TO 370$ IF ERROR
2202 027166 000240 365$: NOP
2203 027170 004737 045042 JSR PC,TIMOUT
2204 027174 000240 NOP
2205 027176 370$:
2206
2207
;*****
;*TEST 20 WRITE, READ W/ IVC ERROR
;*****
TST20:
027176 000004 SCOPE ;SCOPE CALL
027200 000240 NOP ;START OF TEST
027202 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
027206 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
027212 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
027216 012737 000020 001226 MOV #20,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2208
2209
;*****
;LOOP #1 WRITE,READ
;*****
;LOAD PARAMETERS AND GENERATE DATA BUFFER
10$:
2213 027224 MOV #0,RMDCO ;CYLINDER = 0
2214 027224 012737 000000 001444 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
2215 027232 012737 000000 001416 MOV #BUFONE,RMBAO ;BUS ADDRESS
2216 027240 012737 106314 001414 MOV #-256.,RMWCO ;256. WORDS (2'S COMP)
2217 027246 012737 177400 001412 MOV #FMT16,RMOFO ;16 BIT FORMAT
2218 027254 012737 010000 001442 MOV #WD,RMCS10 ;WRITE HEADER AND DATA COMMAND
2219 027262 012737 000060 001410
2220
2221 ;VERIFY THAT SECTOR IS NOT BAD
027270 004737 041416 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
027274 000405 BR 70$ ;GO TO 70$ IF NO ERROR
027276 104401 070230 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
027302 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
027304 000137 030216 JMP 350$ ;GO TO 350$ IF ERROR
2222 027310 70$:
2223 027310 012737 001416 001174 MOV #RMDAO,$TMP0 ;USE SECTOR FOR DATA PATTERN
2224 027316 012737 000001 001176 MOV #1,$TMP1
2225 027324 004737 043344 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
2226
2227 ;PREPARE DEVICE FOR WRITE OPERATION
2228 027330 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
027334 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
027336 000404 BR 80$ ;GO TO 80$ IF NO ERROR
027340 000240 NOP ;RETURN HERE IF ERROR
027342 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
027344 000137 030216 JMP 350$ ;GO TO 350$ IF ERROR

```

```

2229 027350      80$:
2230
2231
2232 027350 112737 000024 001551 ;RESET VOLUME VALID
      027356 112737 000200 001552      MOVB #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      027364 012737 000001 001434      MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
      027372 004737 044500      MOV #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
      027376 000404      JSR PC,PUT ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      027400 000240      BR 90$ ;GO TO 90$ IF NO ERROR
      027402 104000      NOP ;RETURN HERE IF ERROR
      027404 000137 030216      EMT ;ERROR # DEFINED BY PUT SUBROUTINE
      027410      JMP 350$ ;GO TO 350$ IF ERROR

2233 027410      90$:
2234
2235      ;SETUP PARAMETERS AND EXECUTE WRITE DATA COMMAND
2236 027410      120$:
2237 027410 012737 000061 001410      MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
2238 027416 012737 000000 001434      MOV #0,RMMR10 ;RESET DIAGNOSTIC MODE
2239 027424 012702 001551      MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
2240 027430 112722 000024      MOVB #RMMR1,(R2)+
2241 027434 112722 000006      MOVB #RMDA,(R2)+
2242 027440 112722 000034      MOVB #RMDC,(R2)+
2243 027444 112722 000032      MOVB #RMOF,(R2)+
2244 027450 112722 000002      MOVB #RMWC,(R2)+
2245 027454 112722 000004      MOVB #RMBA,(R2)+
2246 027460 112722 000000      MOVB #RMCS1,(R2)+
2247 027464 112722 000200      MOVB #200,(R2)+ ;TERMINATE TABLE
2248
2249 027470 004737 044500      JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027474 000404      BR 130$ ;GO TO 130$ IF NO ERROR
      027476 000240      NOP ;RETURN HERE IF ERROR
      027500 104000      EMT ;ERROR # DEFINED BY PUT SUBROUTINE
      027502 000137 030216      JMP 350$ ;GO TO 350$ IF ERROR

2250 027506      130$:
2251
2252      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      027506 004737 044144      JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
2253
2254      ;WAIT FOR COMMAND TO COMPLETE
      027512 004737 045042      JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
2255
2256      ;GO READ STATUS FOR WRITE COMMAND
2257 027516 004737 044230      JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027522 000404      BR 140$ ;GO TO 140$ IF NO ERROR
      027524 000240      NOP ;RETURN HERE IF ERROR
      027526 104000      EMT ;ERROR # DEFINED BY GET SUBROUTINE
      027530 000137 030216      JMP 350$ ;GO TO 350$ IF ERROR

2258 027534      140$:
2259
2260      ;CHECK FOR ERRORS DURING WRITE OPERATION
2261 027534 004737 045226      JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      027540 000405      BR 150$ ;GO TO 150$ IF NO ERROR
      027542 000240      NOP ;RETURN HERE IF ERROR
      027544 104000      EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
      027546 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      027550 000137 030216      JMP 350$ ;GO TO 350$ IF ERROR

2262 027554      150$:
2263 027554 032737 010000 001376      BIT #IVC,RMER2I ;WAS "IVC" DETECTED??

```

```

2264 027562 001024          BNE      170$          ;YES!!
2265
2266 027564 004737 057742    JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      027570 000405          BR       160$          ;GO TO 160$ IF NO ERROR
      027572 000240          NOP
      027574 104000          EMT      ;RETURN HERE IF ERROR
      027576 004736          JSR      PC,@(SP)+     ;ERROR # DEFINED BY DTASTS SUBROUTINE
      027600 000137 030216    JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
2267 027604          160$:          ;GO TO 350$ IF ERROR
2268 027604 013737 001376 001142    MOV      RMER2I,$BDDAT ;RECEIVED STATUS
2269 027612 013737 001376 001140    MOV      RMER2I,$GDDAT ;EXPECTED STATUS
2270 027620 052737 010000 001140    BIS      #IVC,$GDDAT
2271 027626 104342          EMT      342
2272 027630 000137 030216    JMP      350$
2273 027634          170$:
2274 027634 004737 046060    JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      027640 000405          BR       180$          ;GO TO 180$ IF NO ERROR
      027642 000240          NOP      ;RETURN HERE IF ERROR
      027644 104000          EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
      027646 004736          JSR      PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      027650 000137 030216    JMP      350$          ;GO TO 350$ IF ERROR
2275 027654          180$:
2276
2277          ;CHANGE LOOP ADDRESSES
2278 027654 012737 027664 001124    MOV      #190$,$LPERR
2279 027662 000410          BR       200$
2280
2281          ;*****
2282          ;LOOP #2          READ
2283
2284 027664          190$:
2285
2286          ;PREPARE DEVICE FOR READ OPERATION
2287 027664 004737 040472    JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      027670 154130          .WORD   154130        ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      027672 000404          BR       200$          ;GO TO 200$ IF NO ERROR
      027674 000240          NOP      ;RETURN HERE IF ERROR
      027676 104000          EMT      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      027700 000137 030216    JMP      350$          ;GO TO 350$ IF ERROR
2288 027704          200$:
2289
2290          ;RESET VOLUME VALID
2291 027704 112737 000024 001551    MOV      #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
      027712 112737 000200 001552    MOV      #200,PUTINX+1 ;SET TERMINATOR BYTE
      027720 012737 000001 001434    MOV      #DMD,RMMR10   ;SET RMMR1 OUTPUT BUFFER = DMD
      027726 004737 044500          JSR      PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      027732 000404          BR       210$          ;GO TO 210$ IF NO ERROR
      027734 000240          NOP      ;RETURN HERE IF ERROR
      027736 104000          EMT      ;ERROR DEFINED BY PUT SUBROUTINE
      027740 000137 030216    JMP      350$          ;GO TO 350$ IF ERROR

```

```

2292 027744          210$:
2293
2294          ;READ DATA FROM DEVICE
2295 027744          240$:
2296 027744 012737 000071 001410      MOV      #RD!GO,RMCS10      ;READ DATA COMMAND
2297 027752 012737 107320 001414      MOV      #BUF TWO,RMBAO     ;LOAD STARTING BUFFER ADDRESS
2298 027760 012737 000000 001434      MOV      #0,RMMR10         ;RESET DIANOSTIC MODE
2299 027766 012702 001551              MOV      #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
2300 027772 112722 000024              MOVB     #RMMR1,(R2)+
2301 027776 112722 000006              MOVB     #RMDA,(R2)+
2302 030002 112722 000032              MOVB     #RMOF,(R2)+
2303 030006 112722 000034              MOVB     #RMDC,(R2)+
2304 030012 112722 000004              MOVB     #RMB A,(R2)+
2305 030016 112722 000002              MOVB     #RMWC,(R2)+
2306 030022 112722 000000              MOVB     #RMCS1,(R2)+
2307 030026 112712 000200              MOVB     #200,(R2)
2308
2309 030032 004737 044500              JSR      PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      030036 000404              BR       250$             ;GO TO 250$ IF NO ERROR
      030040 000240              NOP
      030042 104000              EMT
      030044 000137 030216          JMP      350$             ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 350$ IF ERROR
2310 030050          250$:
2311
2312          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
2313 030050 004737 044144              JSR      PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2314
2315          ;WAIT FOR COMMAND TO COMPLETE
2316 030054 004737 045042              JSR      PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
2317
2317 030060 004737 044230          ;GO READ STATUS FOR READ OPERATION
      030064 000404              JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      030066 000240              BR       260$             ;GO TO 260$ IF NO ERROR
      030070 104000              NOP                       ;RETURN HERE IF ERROR
      030072 000137 030216          EMT                       ;ERROR # DEFINED BY GET SUBROUTINE
      030076 000137 030216          JMP      350$             ;GO TO 350$ IF ERROR
2318 030076          260$:
2319
2320          ;CHECK FOR ERRORS DURING READ OPERATION
2321 030076 004737 045226              JSR      PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      030102 000405              BR       270$             ;GO TO 270$ IF NO ERROR
      030104 000240              NOP                       ;RETURN HERE IF ERROR
      030106 104000              EMT                       ;ERROR # DEFINED BY PRIERR SUBROUTINE
      030110 004736              JSR      PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      030112 000137 030216          JMP      350$             ;GO TO 350$ IF ERROR
2322 030116          270$:
2323 030116 032737 010000 001376      BIT      #IVC,RMER2I      ;WAS "IVC" DETECTED??
2324 030124 001024              BNE     290$             ;YES!!
2325
2326 030126 004737 057742              JSR      PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      030132 000405              BR       280$             ;GO TO 280$ IF NO ERROR
      030134 000240              NOP                       ;RETURN HERE IF ERROR
      030136 104000              EMT                       ;ERROR # DEFINED BY DTASTS SUBROUTINE
      030140 004736              JSR      PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      030142 000137 030216          JMP      350$             ;GO TO 350$ IF ERROR
2327 030146          280$:
2328 030146 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS

```

```

2329 030154 052737 010000 001140 BIS #IVC,$GDDAT
2330 030162 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
2331 030170 104342 EMT 342
2332 030172 000137 030216 JMP 350$
2333 030176 290$:
2334 030176 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    030202 000405 BR 300$ ;GO TO 300$ IF NO ERROR
    030204 000240 NOP ;RETURN HERE IF ERROR
    030206 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    030210 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    030212 000137 030216 JMP 350$ ;GO TO 350$ IF ERROR
2335 030216 300$:
2336
2337 030216 350$:
2338
2339
;*****
;*TEST 21 WRITE, READ W/ ABORT
;*****
TST21:
    030216 000004 SCOPE ;SCOPE CALL
    030220 000240 NOP ;START OF TEST
    030222 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
    030226 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
    030232 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
    030236 012737 000021 001226 MOV #21,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

;*****
;LOOP #1 WRITE,READ
;LOAD PARAMETERS AND GENERATE DATA BUFFER
2345 030244 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
2346 030252 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
2347 030260 012737 106314 001414 MOV #BUFONE,RMBAD ;BUS ADDRESS
2348 030266 012737 177400 001412 MOV #-256.,RMWCO ;256. WORDS (2'S COMP)
2349 030274 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
2350 030302 012737 000061 001410 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
2351 030310 70$:
2352
2353
2354 030310 004737 040472 ;PREPARE DEVICE FOR WRITE OPERATION
    030314 154130 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
    .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
    ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
    ;VERIFY RECALIBRATION
    ;GO TO 80$ IF NO ERROR
    ;RETURN HERE IF ERROR
    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    ;GO TO 350$ IF ERROR
    030316 000404 BR 80$
    030320 000240 NOP
    030322 104000 EMT
    030324 000137 031134 JMP 350$
2355 030330 80$:
2356
2357
2358 030330 112737 000014 001551 ;SET UNSAFE ERROR
    030336 112737 000200 001552 MOVB #RMER1,PUTINX ;SETUP PUT INDEX TABLE
    MOVB #200,PUTINX+1 ;SET TERMINATOR BYTE
    
```

```

030344 012737 040000 001424      MOV      #UNS,RMER10      ;SET RMER1 OUTPUT BUFFER = UNS
030352 004737 044500                JSR      PC,PUT           ;GO WRITE RMER1 VIA PUT SUBROUTINE
030356 000404                        BR       90$             ;GO TO 90$ IF NO ERROR
030360 000240                        NOP                       ;RETURN HERE IF ERROR
030362 104000                        EMT                      ;ERROR DEFINED BY PUT SUBROUTINE
030364 000137 031134                JMP      350$            ;GO TO 350$ IF ERROR
2359 030370      90$:
2360
2361      ;WRITE DATA TO THE DRIVE
2362 030370      120$:
2363 030370 012737 000061 001410      MOV      #WD!GO,RMCS10    ;WRITE DATA COMMAND
2364 030376 012702 001551                MOV      #PUTINX,R2      ;LOAD PUT REGISTER INDEX TABLE
2365 030402 112722 000006                MOV      #RMDA,(R2)+
2366 030406 112722 000034                MOV      #RMDC,(R2)+
2367 030412 112722 000032                MOV      #RMOF,(R2)+
2368 030416 112722 000004                MOV      #RMBA,(R2)+
2369 030422 112722 000002                MOV      #RMWC,(R2)+
2370 030426 112722 000000                MOV      #RMCS1,(R2)+
2371 030432 112722 000200                MOV      #200,(R2)+      ;TERMINATE TABLE
2372
2373 030436 004737 044500                JSR      PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030442 000404                        BR       130$            ;GO TO 130$ IF NO ERROR
030444 000240                        NOP                       ;RETURN HERE IF ERROR
030446 104000                        EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
030450 000137 031134                JMP      350$            ;GO TO 350$ IF ERROR
2374 030454      130$:
2375
2376      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
030454 004737 044144                JSR      PC,GETSTS       ;GO TO GETSTS SUBROUTINE
2377
2378      ;GO GET REGISTER STATUS
2379 030460 004737 044230                JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
030464 000404                        BR       135$            ;GO TO 135$ IF NO ERROR
030466 000240                        NOP                       ;RETURN HERE IF ERROR
030470 104000                        EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
030472 000137 031134                JMP      350$            ;GO TO 350$ IF ERROR
2380 030476      135$:
2381 030476 032737 020000 001346      BIT      #PIP,RMDSI      ;WAS COMMAND EXECUTED?
2382 030504 001412                        BEQ      136$            ;NO, GO WAIT
2383 030506 013737 001346 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
2384 030514 042737 020000 001140      BIC      #PIP,$GDDAT
2385 030522 013737 001346 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
2386 030530 104347                        EMT      347
2387 030532      136$:
2388
2389      ;WAIT FOR COMMAND TO COMPLETE
030532 004737 045042                JSR      PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
2390
2391      ;GO GET REGISTER STATUS
2392 030536 004737 044230                JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
030542 000404                        BR       140$            ;GO TO 140$ IF NO ERROR
030544 000240                        NOP                       ;RETURN HERE IF ERROR
030546 104000                        EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
030550 000137 031134                JMP      350$            ;GO TO 350$ IF ERROR
2393 030554      140$:
2394 030554 004737 045226                JSR      PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
030560 000405                        BR       150$            ;GO TO 150$ IF NO ERROR
  
```



```

030562 000240      NOP      ;RETURN HERE IF ERROR
030564 104000      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
030566 004736      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030570 000137 031134 JMP      350$    ;GO TO 350$ IF ERROR
2395 030574      150$:      JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
2396 030574 004737 046060 BR      180$    ;GO TO 180$ IF NO ERROR
030600 000405      NOP      ;RETURN HERE IF ERROR
030602 000240      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
030604 104000      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030606 004736      JMP      350$    ;GO TO 350$ IF ERROR
030610 000137 031134 180$:
030614
2397
2398
2399 ;CHANGE LOOP ADDRESSES
2400 030614 012737 030624 001124 MOV      #190$,$LPERR
2401 030622 000410      BR      200$
2402
2403 ;*****
2404 ;LOOP #2      READ
2405
2406 030624      190$:
2407
2408 ;PREPARE DEVICE FOR READ OPERATION
2409 030624 004737 040472 JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
030630 154130      .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 200$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR
030632 000404      BR      200$
030634 000240      NOP
030636 104000      EMT
030640 000137 031134 JMP      350$
2410 030644      200$:
2411
2412 ;SET UNSAFE ERROR
2413 030644 112737 000014 001551 MOV      #RMER1,PUTINX ;SETUP PUT INDEX TABLE
030652 112737 000200 001552 MOV      #200,PUTINX+1 ;SET TERMINATOR BYTE
030660 012737 040000 001424 MOV      #UNS,RMER10 ;SET RMER1 OUTPUT BUFFER = UNS
030666 004737 044500      JSR      PC,PUT ;GO WRITE RMER1 VIA PUT SUBROUTINE
030672 000404      BR      210$    ;GO TO 210$ IF NO ERROR
030674 000240      NOP      ;RETURN HERE IF ERROR
030676 104000      EMT      ;ERROR DEFINED BY PUT SUBROUTINE
030700 000137 031134 JMP      350$    ;GO TO 350$ IF ERROR
2414 030704      210$:
2415
2416 ;READ DATA FROM DEVICE
2417 030704      240$:
2418 030704 012737 000071 001410 MOV      #RD!GO,RMCS10 ;READ DATA COMMAND
2419 030712 012702 001551      MOV      #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
2420 030716 112722 000006      MOV      #RMDA,(R2)+
2421 030722 112722 000032      MOV      #RMOF,(R2)+
2422 030726 112722 000034      MOV      #RMDC,(R2)+
2423 030732 112722 000004      MOV      #RMBA,(R2)+

```

```

2424 030736 112722 000002      MOVB  #RMWC,(R2)+
2425 030742 112722 000000      MOVB  #RMCS1,(R2)+
2426 030746 112712 000200      MOVB  #200,(R2)
2427
2428 030752 004737 044500      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      030756 000404      BR    250$        ;GO TO 250$ IF NO ERROR
      030760 000240      NOP                    ;RETURN HERE IF ERROR
      030762 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      030764 000137 031134      JMP   350$        ;GO TO 350$ IF ERROR
2429 030770      250$:
2430
2431      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      030770 004737 044144      JSR   PC,GETSTS   ;GO TO GETSTS SUBROUTINE
2432
2433      ;SEE IF DEVICE STARTED COMMAND
2434 030774 004737 044230      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031000 000404      BR    255$        ;GO TO 255$ IF NO ERROR
      031002 000240      NOP                    ;RETURN HERE IF ERROR
      031004 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      031006 000137 031134      JMP   350$        ;GO TO 350$ IF ERROR
2435 031012      255$:
2436 031012 032737 020000 001346      BIT   #PIP,RMSDI   ;WAS COMMAND EXECUTED??
2437 031020 001414      BEQ   256$        ;NO!!
2438 031022 013737 001346 001140      MOV   RMSDI,$GDDAT ;EXPECTED STATUS
2439 031030 042737 020000 001140      BIC   #PIP,$GDDAT
2440 031036 013737 001346 001142      MOV   RMSDI,$BDDAT ;RECEIVED STATUS
2441 031044 104347      EMT 347
2442 031046 000137 031134      JMP   350$
2443 031052      256$:
2444
2445      ;WAIT FOR COMMAND TO COMPLETE
      031052 004737 045042      JSR   PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
2446
2447      ;GO READ STATUS FOR READ OPERATION
2448 031056 004737 044230      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031062 000404      BR    260$        ;GO TO 260$ IF NO ERROR
      031064 000240      NOP                    ;RETURN HERE IF ERROR
      031066 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      031070 000137 031134      JMP   350$        ;GO TO 350$ IF ERROR
2449 031074      260$:
2450 031074 004737 045226      JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      031100 000405      BR    270$        ;GO TO 270$ IF NO ERROR
      031102 000240      NOP                    ;RETURN HERE IF ERROR
      031104 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      031106 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      031110 000137 031134      JMP   350$        ;GO TO 350$ IF ERROR
2451 031114      270$:
2452 031114 004737 046060      JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      031120 000405      BR    300$        ;GO TO 300$ IF NO ERROR
      031122 000240      NOP                    ;RETURN HERE IF ERROR
      031124 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      031126 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      031130 000137 031134      JMP   350$        ;GO TO 350$ IF ERROR
2453 031134      300$:
2454
2455 031134      350$:
2456
  
```

2457

```

:*****
:*TEST 22 WRITE, READ EACH CURRENT LEVEL
:*****
TST22:

```

031134
031134 000004
031136 000240
031140 012706 001100
031144 013700 001276
031150 013701 001464
031154 012737 000022 001226
2458 031162
2459
2460
2461
2462
2463
2464 031162 004737 040472
031166 054130

```

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

```

10\$:

```

:*****
:LOOP #1 FORMAT,WRITE,READ

```

```

;PREPARE THE DEVICE FOR FORMAT OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
.WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
;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 20$ ;GO TO 20$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR

```

2465 031202
2466
2467
2468 031202 012737 000000 001444
2469 031210 012737 000000 001416
2470 031216 012737 106314 001414
2471 031224 012737 177376 001412
2472 031232 012737 010000 001442
2473 031240 012737 000063 001410
2474
2475

```

;LOAD PARAMETERS AND GENERATE DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #BUFONE,RMBAO ;BUS ADDRESS
MOV #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

```

```

;VERIFY THAT SECTOR IS NOT BAD
JSR PC,BADSCT ;CALL BAD SECTOR MODULE
BR 26$ ;GO TO 26$ IF NO ERROR
TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR

```

2476 031266
2477 031266 012737 071730 001174
2478 031274 012737 000001 001176
2479 031302 004737 043344
2480
2481

```

26$:
MOV #ZEROS,$TMP0 ;STARTING ADDRESS OF PATTERN
MOV #1,$TMP1 ;RANGE OF PATTERN
JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT

```

```

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
MOVB #RMDC,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+

```

2482 031306 012702 001551
2483 031312 112722 000034
2484 031316 112722 000006
2485 031322 112722 000004
2486 031326 112722 000002
2487 031332 112722 000032
2488 031336 112722 000000

```

2489 031342 112712 000200          MQVB  #200,(R2)          ;TERMINATE TABLE
2490 031346          30$:
2491 031346 004737 044500          JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      031352 000404          BR    40$            ;GO TO 40$ IF NO ERROR
      031354 000240          NOP                    ;RETURN HERE IF ERROR
      031356 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      031360 000137 032172          JMP   350$           ;GO TO 350$ IF ERROR
2492 031364          40$:
2493
2494          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      031364 004737 044144          JSR   PC,GETSTS       ;GO TO GETSTS SUBROUTINE
2495
2496          ;WAIT FOR COMMAND TO COMPLETE
      031370 004737 045042          JSR   PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
2497
2498          ;GO READ STATUS FOR FORMAT OPERATION
2499 031374 004737 044230          JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031400 000404          BR    50$            ;GO TO 50$ IF NO ERROR
      031402 000240          NOP                    ;RETURN HERE IF ERROR
      031404 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      031406 000137 032172          JMP   350$           ;GO TO 350$ IF ERROR
2500 031412          50$:
2501
2502          ;VERIFY NO ERRORS DURING FORMAT
2503 031412 004737 057742          JSR   PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
      031416 000405          BR    60$            ;GO TO 60$ IF NO ERROR
      031420 000240          NOP                    ;RETURN HERE IF ERROR
      031422 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      031424 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      031426 000137 032172          JMP   350$           ;GO TO 350$ IF ERROR
2504 031432          60$:
2505
2506          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
2507 031432 012737 031442 001124          MOV   #70$,$LPERR
2508 031440 000410          BR    80$            ;SKIP TO WRITE OPERATION
2509
2510          ;:*****
2511          ;LOOP #2          WRITE,READ
2512
2513 031442          70$:
2514
2515          ;PREPARE DEVICE FOR WRITE OPERATION
2516          JSR   PC,TSTPRP ;PREPARE DEVICE FOR TEST
2517 031442 004737 040472          .WORD 054130 ;TASK DESCRIPTOR AS FOLLOWS:
      031446 054130          ;CLEAR CONTROLLER & SELECT DEVICE
      031450 000404          BR    80$            ;VERIFY CONTROLLER CLEAR OPERATION
      031452 000240          NOP                    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      031454 104000          EMT                    ;VERIFY PACK ACKNOWLEDGE
      031456 000137 032172          JMP   350$           ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
2518 031462          80$:          ;VERIFY RECALIBRATION
2519          ;GO TO 80$ IF NO ERROR
      031452 000240          NOP                    ;RETURN HERE IF ERROR
      031454 104000          EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      031456 000137 032172          JMP   350$           ;GO TO 350$ IF ERROR

```

```

2520 031462      120$:
2521
2522
2523 031462 012737 177400 001412 ;WRITE DATA TO THE DRIVE
2524 031470 012737 106320 001414   MOV #-256.,RMWCO           ;256. WORDS (2'S COMP)
2525 031476 012737 000061 001410   MOV #BUFONE+4,RMBAO       ;CHANGE ADDRESS
2526 031504 012702 001551           MOV #WD!GO,RMCS10        ;WRITE DATA COMMAND
2527 031510 112722 000006           MOV #PUTINX,R2           ;LOAD PUT REGISTER INDEX TABLE
2528 031514 112722 000034           MOV #RMDA,(R2)+
2529 031520 112722 000032           MOV #RMDC,(R2)+
2530 031524 112722 000004           MOV #RMOF,(R2)+
2531 031530 112722 000002           MOV #RMBA,(R2)+
2532 031534 112722 000000           MOV #RMWC,(R2)+
2533 031540 112722 000200           MOV #RMCS1,(R2)+
2534
2535 031544 004737 044500           MOV #200,(R2)+           ;TERMINATE TABLE
2536 031550 000404           JSR PC,PUT               ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
2537 031552 000240           BR 130$                 ;GO TO 130$ IF NO ERROR
2538 031554 104000           NOP                      ;RETURN HERE IF ERROR
2539 031556 000137 032172           EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
2540
2541 031558 000137 032172           JMP 350$                ;GO TO 350$ IF ERROR
2542
2543 031562 004737 044144           ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
2544 031564 004737 044144           JSR PC,GETSTS           ;GO TO GETSTS SUBROUTINE
2545
2546 031566 004737 045042           ;WAIT FOR COMMAND TO COMPLETE
2547 031568 004737 045042           JSR PC,TIMOUT          ;GO TO TIMOUT SUBROUTINE
2548
2549 031572 004737 044230           ;GO READ STATUS FOR WRITE COMMAND
2550 031574 004737 044230           JSR PC,GET             ;GO READ REGISTER(S) WITH GET SUBROUTINE
2551 031576 000404           BR 140$                 ;GO TO 140$ IF NO ERROR
2552 031600 000240           NOP                      ;RETURN HERE IF ERROR
2553 031602 104000           EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
2554 031604 000137 032172           JMP 350$                ;GO TO 350$ IF ERROR
2555
2556 031610 004737 045226           140$:
2557 031612 004737 045226           ;CHECK FOR ERRORS DURING WRITE OPERATION
2558 031614 000405           JSR PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
2559 031616 000240           BR 150$                 ;GO TO 150$ IF NO ERROR
2560 031620 104000           NOP                      ;RETURN HERE IF ERROR
2561 031622 004736           EMT                      ;ERROR # DEFINED BY PRIERR SUBROUTINE
2562 031624 000137 032172           JSR PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
2563
2564 031630 004737 057742           150$:
2565 031632 000405           JSR PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
2566 031634 000240           BR 160$                 ;GO TO 160$ IF NO ERROR
2567 031636 000137 032172           NOP                      ;RETURN HERE IF ERROR
2568 031640 104000           EMT                      ;ERROR # DEFINED BY DTASTS SUBROUTINE
2569 031642 004736           JSR PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
2570 031644 000137 032172           JMP 350$                ;GO TO 350$ IF ERROR
2571
2572 031650 004737 046060           160$:
2573 031652 000405           JSR PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
2574 031654 000240           BR 180$                 ;GO TO 180$ IF NO ERROR
2575 031656 000137 032172           NOP                      ;RETURN HERE IF ERROR
2576 031660 104000           EMT                      ;ERROR # DEFINED BY SECERR SUBROUTINE
    
```

```

031662 004736      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
031664 000137 032172  JMP    350$                ;GO TO 350$ IF ERROR
2554 031670      180$:
2555
2556      ;CHANGE LOOP ADDRESSES
2557 031670 012737 031700 001124  MOV    #190$,$LPERR
2558 031676 000410      BR     200$                ;SKIP TO NEXT OPERATION
2559
2560      ;:*****
2561      ;LOOP #3      READ
2562
2563 031700      190$:
2564
2565      ;PREPARE DEVICE FOR READ OPERATION
2566 031700 004737 040472  JSR    PC,TSTPRP          ;PREPARE DEVICE FOR TEST
031704 054130      .WORD 054130            ;TASK DESCRIPTOR AS FOLLOWS:
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 200$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 350$ IF ERROR
                                BR     200$
031706 000404      BR     200$
031710 000240      NOP
031712 104000      EMT
031714 000137 032172  JMP    350$
2567 031720      200$:
2568
2569 031720      240$:
2570
2571      ;READ DATA FROM DEVICE
2572 031720 012737 107324 001414  MOV    #BUFTWO+4,RMBA0    ;CHANGE ADDRESS
2573 031726 012737 177400 001412  MOV    #-256.,RMWCO      ;256. WORDS (2'S COMP)
2574 031734 012737 000071 001410  MOV    #RD!GO,RMCS10     ;READ DATA COMMAND
2575 031742 012702 001551      MOV    #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
2576 031746 112722 000006  MOVB   #RMDA,(R2)+
2577 031752 112722 000032  MOVB   #RMOF,(R2)+
2578 031756 112722 000034  MOVB   #RMDC,(R2)+
2579 031762 112722 000004  MOVB   #RMBA,(R2)+
2580 031766 112722 000002  MOVB   #RMWC,(R2)+
2581 031772 112722 000000  MOVB   #RMCS1,(R2)+
2582 031776 112712 000200  MOVB   #200,(R2)
2583
2584 032002 004737 044500      JSR    PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
032006 000404      BR     250$            ;GO TO 250$ IF NO ERROR
032010 000240      NOP                    ;RETURN HERE IF ERROR
032012 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
032014 000137 032172  JMP    350$            ;GO TO 350$ IF ERROR
2585 032020      250$:
2586
2587      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
032020 004737 044144  JSR    PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2588
2589      ;WAIT FOR COMMAND TO COMPLETE
032024 004737 045042  JSR    PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
2590
2591      ;GO READ STATUS FOR READ OPERATION
  
```

```

2592 032030 004737 044230      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      032034 000404          BR      260$      ;GO TO 260$ IF NO ERROR
      032036 000240          NOP          ;RETURN HERE IF ERROR
      032040 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
      032042 000137 032172    JMP     350$      ;GO TO 350$ IF ERROR
2593 032046          260$:
2594
2595          ;CHECK FOR ERRORS DURING READ OPERATION
2596 032046 004737 045226      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      032052 000405          BR      270$      ;GO TO 270$ IF NO ERROR
      032054 000240          NOP          ;RETURN HERE IF ERROR
      032056 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
      032060 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      032062 000137 032172    JMP     350$      ;GO TO 350$ IF ERROR
2597 032066          270$:
2598 032066 004737 057742      JSR    PC,DTASTS  ;GO VERIFY RESULTS OF DATA TRANSFER
      032072 000405          BR      280$      ;GO TO 280$ IF NO ERROR
      032074 000240          NOP          ;RETURN HERE IF ERROR
      032076 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
      032100 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      032102 000137 032172    JMP     350$      ;GO TO 350$ IF ERROR
2599 032106          280$:
2600
2601 032106          290$:
2602 032106 004737 046060      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      032112 000405          BR      300$      ;GO TO 300$ IF NO ERROR
      032114 000240          NOP          ;RETURN HERE IF ERROR
      032116 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
      032120 004736          JSR    PC,@(SP)+  ;GO BACK FOR MORE ERROR CHECKS
      032122 000137 032172    JMP     350$      ;GO TO 350$ IF ERROR
2603 032126          300$:
2604 032126 004737 043602      JSR    PC,CMPBUF   ;GO COMPARE WRITE, READ DATA BUFFERS
      032132 106320          .WORD   BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
      032134 107324          .WORD   BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
      032136 000404          BR      310$      ;GO TO 310$ IF NO ERROR
      032140 000240          NOP          ;RETURN HERE IF ERROR
      032142 104000          EMT          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      032144 000137 032172    JMP     350$      ;GO TO 350$ IF ERROR
2605 032150          310$:
2606 032150 062737 000200 001444  ADD    #128.,RMDCO ;ADVANCE TO NEXT THRESHOLD
2607 032156 023727 001444 001400  CMP    RMDCO,#768. ;DONE??
2608 032164 101002          BHI    350$      ;YES!!
2609 032166 000137 031210    JMP     25$      ;DO NEXT CURRENT LEVEL
2610 032172          350$:
2611
2612

```

```

*****
;*TEST 23      WRITE, WRITE CHECK W/ CURRENT LEVEL SWITCHING
*****
TST23:

```

```

032172 000004          SCOPE          ;SCOPE CALL
032174 000240          NOP          ;START OF TEST
032176 012706 001100    MOV    #STACK,SP ;INITIALIZE STACK POINTER
032202 013700 001276    MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
032206 013701 001464    MOV    TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
032212 012737 000023 001226  MOV    #23,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
2613 032220          10$:
2614

```

```

2615
2616
2617
2618 032220 004737 040472
      032224 054130
      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      .WORD 054130          ;TASK DESCRIPTOR AS FOLLOWS:
                               ;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 20$ IF NO ERROR
                               ;RETURN HERE IF ERROR
                               ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                               ;GO TO 350$ IF ERROR

      032226 000404
      032230 000240
      032232 104000
      032234 000137 033176
      032240
2619 20$:
2620
2621 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
2622 032240 012737 000577 001444
      MOV    #383.,RMDCO      ;CYLINDER = 383.
2623 032246 013737 001332 001416
      MOV    LSTRK,RMDAO      ;SET LAST TRACK AND
2624 032254 112737 000037 001416
      MOV    #31.,RMDAO       ;LAST SECTOR
2625 032262 012737 106314 001414
      MOV    #BUFONE,RMBAO     ;BUS ADDRESS
2626 032270 012737 176774 001412
      MOV    #-258.*2,RMWCO    ;2 SECTORS (2'S COMP)
2627 032276 012737 010000 001442
      MOV    #FMT16,RMOFO     ;16 BIT FORMAT
2628 032304 012737 000063 001410
      MOV    #WH!GO,RMCS10     ;WRITE HEADER AND DATA COMMAND
2629
2630 ;VERIFY THAT SECTOR IS NOT BAD
      JSR    PC,BADSCT        ;CALL BAD SECTOR MODULE
      BR     25$              ;GO TO 25$ IF NO ERROR
      TYPE  ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
      EMT
      JMP    350$            ;ERROR # DEFINED BY BADSCT SUBROUTINE
                               ;GO TO 350$ IF ERROR
2631 25$:
2632 032332 012737 071666 001174
      MOV    #ONES,$TMP0       ;STARTING ADDRESS OF PATTERN
2633 032340 012737 000001 001176
      MOV    #1,$TMP1          ;RANGE OF PATTERN
2634 032346 004737 043344
      JSR    PC,GENBUF         ;GO GENERATE BUFFER FOR FORMAT
2635
2636 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
      MOV    #PUTINX,R2        ;R2 = BYTE ENTRY POSITION
2637 032352 012702 001551
      MOV    #RMDC,(R2)+
2638 032356 112722 000034
      MOV    #RMDA,(R2)+
2639 032362 112722 000006
      MOV    #RMBA,(R2)+
2640 032366 112722 000004
      MOV    #RMWC,(R2)+
2641 032372 112722 000002
      MOV    #RMWC,(R2)+
2642 032376 112722 000032
      MOV    #RMOF,(R2)+
2643 032402 112722 000000
      MOV    #RMCS1,(R2)+
2644 032406 112712 000200
      MOV    #200,(R2)        ;TERMINATE TABLE
2645 30$:
2646 032412 004737 044500
      JSR    PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR     40$             ;GO TO 40$ IF NO ERROR
      NOP
      EMT
      JMP    350$            ;RETURN HERE IF ERROR
                               ;ERROR # DEFINED BY PUT SUBROUTINE
                               ;GO TO 350$ IF ERROR
2647 40$:
2648
2649 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR    PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2650

```



```

2651      032434 004737 045042      ;WAIT FOR COMMAND TO COMPLETE
                JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
2652
2653      ;GO READ STATUS FOR FORMAT OPERATION
2654      032440 004737 044230      JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
                BR      50$            ;GO TO 50$ IF NO ERROR
                NOP                     ;RETURN HERE IF ERROR
                EMT                     ;ERROR # DEFINED BY GET SUBROUTINE
                JMP      350$          ;GO TO 350$ IF ERROR
2655      032452 000137 033176      50$:
2656
2657      ;VERIFY NO ERRORS DURING FORMAT
2658      032456 004737 057742      JSR      PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
                BR      60$            ;GO TO 60$ IF NO ERROR
                NOP                     ;RETURN HERE IF ERROR
                EMT                     ;ERROR # DEFINED BY DTASTS SUBROUTINE
                JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                JMP      350$          ;GO TO 350$ IF ERROR
2659      032476      60$:
2660
2661      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
2662      032476 012737 032506 001124  MOV      #70$,$LPERR
2663      032504 000410      BR      80$            ;SKIP TO WRITE OPERATION
2664
2665      ;*****
2666      ;LOOP #2      WRITE,READ
2667
2668      032506      70$:
2669
2670
2671      ;PREPARE DEVICE FOR WRITE OPERATION
2672      032506 004737 040472      JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
                .WORD 054130          ;TASK DESCRIPTOR AS FOLLOWS:
                ;CLEAR CONTROLLER & SELECT DEVICE
                ;VERIFY CONTROLLER CLEAR OPERATION
                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                ;VERIFY PACK ACKNOWLEDGE
                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                ;VERIFY RECALIBRATION
                ;GO TO 80$ IF NO ERROR
                ;RETURN HERE IF ERROR
                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                ;GO TO 350$ IF ERROR
                BR      80$
                NOP
                EMT
                JMP      350$
2673      032514 000404      BR      80$
2674      032516 000240      NOP
2675      032520 104000      EMT
2676      032522 000137 033176      JMP      350$
2677      032526      80$:
2678      ;WRITE DATA TO THE DRIVE
                MOV      #-256.*2,RMWCO ;2 SECTORS (2'S COMP)
                MOV      #BUFONE,RMBAO  ;CHANGE ADDRESS
                MOV      #WD!GO,RMCS10  ;WRITE DATA COMMAND
                MOV      #ONES,$TMP0    ;STARTING ADDRESS OF PATTERN
                MOV      #1,$TMP1       ;RANGE OF PATTERN
                JSR      PC,GENBUF       ;GO GENERATE BUFFER FOR FORMAT
                MOV      #PUTINX,R2     ;LOAD PUT REGISTER INDEX TABLE
                MOV      #RMDA,(R2)+
                MOV      #RMDC,(R2)+
2679      032526 012737 177000 001412  MOV      #-256.*2,RMWCO
2680      032534 012737 106314 001414  MOV      #BUFONE,RMBAO
2681      032542 012737 000061 001410  MOV      #WD!GO,RMCS10
2682      032550 012737 071666 001174  MOV      #ONES,$TMP0
2683      032556 012737 000001 001176  MOV      #1,$TMP1
2684      032564 004737 043344      JSR      PC,GENBUF
2685      032570 012702 001551      MOV      #PUTINX,R2
2686      032574 112722 000006      MOV      #RMDA,(R2)+
                032600 112722 000034      MOV      #RMDC,(R2)+
  
```

```

2687 032604 112722 000032      MOVB  #RMOF,(R2)+
2688 032610 112722 000004      MOVB  #RMBA,(R2)+
2689 032614 112722 000002      MOVB  #RMWC,(R2)+
2690 032620 112722 000000      MOVB  #RMCS1,(R2)+
2691 032624 112722 000200      MOVB  #200,(R2)+
2692                                     ;TERMINATE TABLE
2693 032630 004737 044500      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      032634 000404          BR    130$           ;GO TO 130$ IF NO ERROR
      032636 000240          NOP                    ;RETURN HERE IF ERROR
      032640 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      032642 000137 033176      JMP   350$           ;GO TO 350$ IF ERROR
2694 032646
2695
2696                                     ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      032646 004737 044144      JSR   PC,GETSTS       ;GO TO GETSTS SUBROUTINE
2697
2698                                     ;WAIT FOR COMMAND TO COMPLETE
      032652 004737 045042      JSR   PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
2699
2700                                     ;GO READ STATUS FOR WRITE COMMAND
2701 032656 004737 044230      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      032662 000404          BR    140$           ;GO TO 140$ IF NO ERROR
      032664 000240          NOP                    ;RETURN HERE IF ERROR
      032666 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      032670 000137 033176      JMP   350$           ;GO TO 350$ IF ERROR
2702 032674
2703
2704                                     ;CHECK FOR ERRORS DURING WRITE OPERATION
2705 032674 004737 045226      JSR   PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      032700 000405          BR    150$           ;GO TO 150$ IF NO ERROR
      032702 000240          NOP                    ;RETURN HERE IF ERROR
      032704 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      032706 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      032710 000137 033176      JMP   350$           ;GO TO 350$ IF ERROR
2706 032714
2707 032714 004737 057742      JSR   PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
      032720 000405          BR    160$           ;GO TO 160$ IF NO ERROR
      032722 000240          NOP                    ;RETURN HERE IF ERROR
      032724 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      032726 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      032730 000137 033176      JMP   350$           ;GO TO 350$ IF ERROR
2708 032734
2709
2710 032734
2711 032734 004737 046060      JSR   PC,SECERR       ;GO CHECK FOR SECONDARY ERRORS
      032740 000405          BR    180$           ;GO TO 180$ IF NO ERROR
      032742 000240          NOP                    ;RETURN HERE IF ERROR
      032744 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      032746 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      032750 000137 033176      JMP   350$           ;GO TO 350$ IF ERROR
2712 032754
2713
2714                                     ;CHANGE LOOP ADDRESSES
2715 032754 012737 032764 001124  MOV   #190$,$LPERR
2716 032762 000410          BR    200$
2717                                     ;SKIP TO NEXT OPERATION
2718
;:*****
  
```

```

2719          ;LOOP #3          READ
2720
2721 032764    190$:
2722
2723          ;PREPARE DEVICE FOR READ OPERATION
2724 032764    004737 040472    JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      032770    054130          .WORD   054130    ;TASK DESCRIPTOR AS FOLLOWS:
                                          ;CLEAR CONTROLLER & SELECT DEVICE
                                          ;VERIFY CONTROLLER CLEAR OPERATION
                                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                          ;VERIFY PACK ACKNOWLEDGE
                                          ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                          ;VERIFY RECALIBRATION
      032772    000404          BR      200$    ;GO TO 200$ IF NO ERROR
      032774    000240          NOP
      032776    104000          EMT
      033000    000137 033176    JMP     350$    ;RETURN HERE IF ERROR
                                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                          ;GO TO 350$ IF ERROR
2725 033004    200$:
2726
2727 033004    240$:
2728
2729          ;READ DATA FROM DEVICE
2730 033004    012737 000051 001410  MOV    #WCD!GO, RMCS10    ;READ DATA COMMAND
2731 033012    012702 001551          MOV    #PUTINX, R2        ;LOAD PUT REGISTER INDEX TABLE
2732 033016    112722 000006          MOVB  #RMDA, (R2)+
2733 033022    112722 000032          MOVB  #RMOF, (R2)+
2734 033026    112722 000034          MOVB  #RMDC, (R2)+
2735 033032    112722 000004          MOVB  #RMBA, (R2)+
2736 033036    112722 000002          MOVB  #RMWC, (R2)+
2737 033042    112722 000000          MOVB  #RMCS1, (R2)+
2738 033046    112712 000200          MOVB  #200, (R2)
2739
2740 033052    004737 044500          JSR    PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      033056    000404          BR      250$            ;GO TO 250$ IF NO ERROR
      033060    000240          NOP                    ;RETURN HERE IF ERROR
      033062    104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      033064    000137 033176    JMP     350$            ;GO TO 350$ IF ERROR
2741 033070    250$:
2742
2743          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      033070    004737 044144    JSR    PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2744
2745          ;WAIT FOR COMMAND TO COMPLETE
      033074    004737 045042    JSR    PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
2746
2747          ;GO READ STATUS FOR READ OPERATION
2748 033100    004737 044230          JSR    PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      033104    000404          BR      260$            ;GO TO 260$ IF NO ERROR
      033106    000240          NOP                    ;RETURN HERE IF ERROR
      033110    104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      033112    000137 033176    JMP     350$            ;GO TO 350$ IF ERROR
2749 033116    260$:
2750
2751          ;CHECK FOR ERRORS DURING READ OPERATION
2752 033116    004737 045226          JSR    PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      033122    000405          BR      270$            ;GO TO 270$ IF NO ERROR
      033124    000240          NOP                    ;RETURN HERE IF ERROR
  
```

```

033126 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
033130 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033132 000137 033176 JMP 350$ ;GO TO 350$ IF ERROR
2753 033136 270$:
2754 033136 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
033142 000405 BR 280$ ;GO TO 280$ IF NO ERROR
033144 000240 NOP ;RETURN HERE IF ERROR
033146 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
033150 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033152 000137 033176 JMP 350$ ;GO TO 350$ IF ERROR
2755 033156 280$:
2756
2757 033156 290$:
2758 033156 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
033162 000405 BR 300$ ;GO TO 300$ IF NO ERROR
033164 000240 NOP ;RETURN HERE IF ERROR
033166 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
033170 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
033172 000137 033176 JMP 350$ ;GO TO 350$ IF ERROR
2759 033176 300$:
2760
2761 033176 350$:
2762
2763
;*****
;*TEST 24 WRITE, READ EARLY PEAK SHIFT
;*****
TST24:
033176 000004 SCOPE ;SCOPE CALL
033200 000240 NOP ;START OF TEST
033202 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
033206 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
033212 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
033216 012737 000024 001226 MOV #24,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2764
2765
2766 ;*****
2767 ;LOOP #1 FORMAT,WRITE,READ
2768 033224 10$:
2769
2770 ;PREPARE THE DEVICE FOR FORMAT OPERATION
2771 033224 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
033230 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
033232 000404 BR 20$ ;GO TO 20$ IF NO ERROR
033234 000240 NOP ;RETURN HERE IF ERROR
033236 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
033240 000137 034204 JMP 350$ ;GO TO 350$ IF ERROR
2772 033244 20$:
2773
2774 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
2775 033244 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0

```

```

2776 033252 012737 000000 001416      MOV      #0,RMDAO          ;TRACK = 0, SECTOR = 0
2777 033260 012737 106314 001414      MOV      #BUFONE,RMBAO    ;BUS ADDRESS
2778 033266 012737 177376 001412      MOV      #-258.,RMWCO     ;2 + 256. WORDS (2'S COMP)
2779 033274 012737 010000 001442      MOV      #FMT16,RMOFO     ;16 BIT FORMAT
2780 033302 012737 000063 001410      MOV      #WH!GO,RMCS10    ;WRITE HEADER AND DATA COMMAND
2781
2782                                     ;VERIFY THAT SECTOR IS NOT BAD
      033310 004737 041416      JSR      PC,BADSCT        ;CALL BAD SECTOR MODULE
      033314 000405      BR       25$              ;GO TO 25$ IF NO ERROR
      033316 104401 070230      TYPE    ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
      033322 104000      EMT                                           ;ERROR # DEFINED BY BADSCT SUBROUTINE
      033324 000137 034204      JMP      350$            ;GO TO 350$ IF ERROR
2783 033330      25$:
2784 033330 012737 072142 001174      MOV      #EARLY,$TMP0     ;STARTING ADDRESS OF PATTERN
2785 033336 012737 000001 001176      MOV      #1,$TMP1        ;RANGE OF PATTERN
2786 033344 004737 043344      JSR      PC,GENBUF        ;GO GENERATE BUFFER FOR FORMAT
2787
2788                                     ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
2789 033350 012702 001551      MOV      #PUTINX,R2       ;R2 = BYTE ENTRY POSITION
2790 033354 112722 000034      MOVB    #RMDC,(R2)+
2791 033360 112722 000006      MOVB    #RMDA,(R2)+
2792 033364 112722 000004      MOVB    #RMBA,(R2)+
2793 033370 112722 000002      MOVB    #RMWC,(R2)+
2794 033374 112722 000032      MOVB    #RMOF,(R2)+
2795 033400 112722 000000      MOVB    #RMCS1,(R2)+
2796 033404 112712 000200      MOVB    #200,(R2)        ;TERMINATE TABLE
2797 033410      30$:
2798 033410 004737 044500      JSR      PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      033414 000404      BR       40$              ;GO TO 40$ IF NO ERROR
      033416 000240      NOP                                           ;RETURN HERE IF ERROR
      033420 104000      EMT                                           ;ERROR # DEFINED BY PUT SUBROUTINE
      033422 000137 034204      JMP      350$            ;GO TO 350$ IF ERROR
2799 033426      40$:
2800
2801                                     ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      033426 004737 044144      JSR      PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2802
2803                                     ;WAIT FOR COMMAND TO COMPLETE
      033432 004737 045042      JSR      PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
2804
2805                                     ;GO READ STATUS FOR FORMAT OPERATION
2806 033436 004737 044230      JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      033442 000404      BR       50$              ;GO TO 50$ IF NO ERROR
      033444 000240      NOP                                           ;RETURN HERE IF ERROR
      033446 104000      EMT                                           ;ERROR # DEFINED BY GET SUBROUTINE
      033450 000137 034204      JMP      350$            ;GO TO 350$ IF ERROR
2807 033454      50$:
2808
2809                                     ;VERIFY NO ERRORS DURING FORMAT
2810 033454 004737 057742      JSR      PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      033460 000405      BR       60$              ;GO TO 60$ IF NO ERROR
      033462 000240      NOP                                           ;RETURN HERE IF ERROR
      033464 104000      EMT                                           ;ERROR # DEFINED BY DTASTS SUBROUTINE
      033466 004736      JSR      PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      033470 000137 034204      JMP      350$            ;GO TO 350$ IF ERROR
2811 033474      60$:
2812

```



```

2849 033646 000137 034204          JMP      350$          ;GO TO 350$ IF ERROR
2850 033652          140$:
2851          ;CHECK FOR ERRORS DURING WRITE OPERATION
2852 033652 004737 045226          JSR      PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
          033656 000405          BR       150$        ;GO TO 150$ IF NO ERROR
          033660 000240          NOP
          033662 104000          EMT      ;RETURN HERE IF ERROR
          033664 004736          JSR      PC,@(SP)+   ;ERROR # DEFINED BY PRIERR SUBROUTINE
          033666 000137 034204          JMP      350$        ;GO BACK FOR MORE ERROR CHECKS
          ;GO TO 350$ IF ERROR
2853 033672          150$:
2854 033672 004737 057742          JSR      PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
          033676 000405          BR       160$        ;GO TO 160$ IF NO ERROR
          033700 000240          NOP
          033702 104000          EMT      ;RETURN HERE IF ERROR
          033704 004736          JSR      PC,@(SP)+   ;ERROR # DEFINED BY DTASTS SUBROUTINE
          033706 000137 034204          JMP      350$        ;GO BACK FOR MORE ERROR CHECKS
          ;GO TO 350$ IF ERROR
2855 033712          160$:
2856 033712 004737 046060          JSR      PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
          033716 000405          BR       180$        ;GO TO 180$ IF NO ERROR
          033720 000240          NOP
          033722 104000          EMT      ;RETURN HERE IF ERROR
          033724 004736          JSR      PC,@(SP)+   ;ERROR # DEFINED BY SECERR SUBROUTINE
          033726 000137 034204          JMP      350$        ;GO BACK FOR MORE ERROR CHECKS
          ;GO TO 350$ IF ERROR
2857 033732          180$:
2858
2859          ;CHANGE LOOP ADDRESSES
2860 033732 012737 033742 001124          MOV      #190$,$LPERR
2861 033740 000410          BR       200$
2862
2863          ;*****
2864          ;LOOP #3      READ
2865
2866 033742          190$:
2867
2868          ;PREPARE DEVICE FOR READ OPERATION
2869 033742 004737 040472          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
          033746 154130          .WORD   154130      ;TASK DESCRIPTOR AS FOLLOWS:
          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
          ;CLEAR CONTROLLER & SELECT DEVICE
          ;VERIFY CONTROLLER CLEAR OPERATION
          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
          ;VERIFY PACK ACKNOWLEDGE
          ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
          ;VERIFY RECALIBRATION
          BR       200$          ;GO TO 200$ IF NO ERROR
          NOP
          EMT      ;RETURN HERE IF ERROR
          JMP      350$        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          ;GO TO 350$ IF ERROR
          033750 000404          BR       200$
          033752 000240          NOP
          033754 104000          EMT
          033756 000137 034204          JMP      350$
2870 033762          200$:
2871
2872          ;READ DATA FROM DEVICE
2873 033762          240$:
2874 033762 012737 000071 001410          MOV      #RD!GO,RMCS10 ;READ DATA COMMAND
2875 033770 012737 107324 001414          MOV      #BUFTWO+4,RMBAO ;CHANGE BUFFER
2876 033776 012702 001551          MOV      #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
2877 034002 112722 000006          MOVB    #RMDA,(R2)+
    
```

```

2878 034006 112722 000032      MOVB    #RMOF,(R2)+
2879 034012 112722 000034      MOVB    #RMDC,(R2)+
2880 034016 112722 000004      MOVB    #RMBA,(R2)+
2881 034022 112722 000002      MOVB    #RMWC,(R2)+
2882 034026 112722 000000      MOVB    #RMCS1,(R2)+
2883 034032 112712 000200      MOVB    #200,(R2)
2884
2885 034036 004737 044500      JSR     PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      034042 000404      BR      250$          ;GO TO 250$ IF NO ERROR
      034044 000240      NOP
      034046 104000      EMT
      034050 000137 034204      JMP     350$          ;ERROR # DEFINED BY PUT SUBROUTINE
                          ;GO TO 350$ IF ERROR
2886 034054
2887
2888                          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      034054 004737 044144      JSR     PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2889
2890                          ;WAIT FOR COMMAND TO COMPLETE
      034060 004737 045042      JSR     PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
2891
2892                          ;GO READ STATUS FOR READ OPERATION
2893 034064 004737 044230      JSR     PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      034070 000404      BR      260$          ;GO TO 260$ IF NO ERROR
      034072 000240      NOP
      034074 104000      EMT
      034076 000137 034204      JMP     350$          ;ERROR # DEFINED BY GET SUBROUTINE
                          ;GO TO 350$ IF ERROR
2894 034102
2895
2896                          ;CHECK FOR ERRORS DURING READ OPERATION
2897 034102 004737 045226      JSR     PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      034106 000405      BR      270$          ;GO TO 270$ IF NO ERROR
      034110 000240      NOP
      034112 104000      EMT
      034114 004736      JSR     PC,@(SP)+    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      034116 000137 034204      JMP     350$          ;GO BACK FOR MORE ERROR CHECKS
                          ;GO TO 350$ IF ERROR
2898 034122
2899 034122 004737 057742      JSR     PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      034126 000405      BR      280$          ;GO TO 280$ IF NO ERROR
      034130 000240      NOP
      034132 104000      EMT
      034134 004736      JSR     PC,@(SP)+    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      034136 000137 034204      JMP     350$          ;GO BACK FOR MORE ERROR CHECKS
                          ;GO TO 350$ IF ERROR
2900 034142
2901 034142 004737 046060      JSR     PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      034146 000405      BR      300$          ;GO TO 300$ IF NO ERROR
      034150 000240      NOP
      034152 104000      EMT
      034154 004736      JSR     PC,@(SP)+    ;ERROR # DEFINED BY SECERR SUBROUTINE
      034156 000137 034204      JMP     350$          ;GO BACK FOR MORE ERROR CHECKS
                          ;GO TO 350$ IF ERROR
2902 034162
2903 034162 004737 043602      JSR     PC,CMPBUF     ;GO COMPARE WRITE, READ DATA BUFFERS
      034166 106320      .WORD  BUFOFF+4      ;STARTING ADDRESS OF WRITE BUFFER
      034170 107324      .WORD  BUFTWO+4     ;STARTING ADDRESS OF READ BUFFER
      034172 000404      BR      310$          ;GO TO 310$ IF NO ERROR
      034174 000240      NOP
      034176 104000      EMT
      034200 000137 034204      JMP     350$          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
                          ;GO TO 350$ IF ERROR

```


2904 034204
 2905
 2906 034204
 2907
 2908

310\$:

350\$:

 ;*TEST 25 WRITE, READ MIXED PEAK SHIFT

 TST25:

034204
 034204 000004
 034206 000240
 034210 012706 001100
 034214 013700 001276
 034220 013701 001464
 034224 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,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

2909
 2910
 2911
 2912
 2913 034232
 2914
 2915

 ;LOOP #1 FORMAT,WRITE,READ

10\$:

2916 034232 004737 040472
 034236 154130

;PREPARE THE DEVICE FOR FORMAT OPERATION
 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 20\$ IF NO ERROR
 ;RETURN HERE IF ERROR
 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
 ;GO TO 350\$ IF ERROR

034240 000404
 034242 000240
 034244 104000
 034246 000137 035212
 2917 034252

BR 20\$
 NOP
 EMT
 JMP 350\$

2918
 2919
 2920 034252 012737 000000 001444
 2921 034260 012737 000000 001416
 2922 034266 012737 106314 001414
 2923 034274 012737 177376 001412
 2924 034302 012737 010000 001442
 2925 034310 012737 000063 001410
 2926
 2927

;LOAD PARAMETERS AND GENERATE DATA BUFFER
 MOV #0,RMDCO ;CYLINDER = 0
 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
 MOV #BUFONE,RMBAO ;BUS ADDRESS
 MOV #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
 MOV #FMT16,RMOFO ;16 BIT FORMAT
 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

;VERIFY THAT SECTOR IS NOT BAD
 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
 BR 25\$;GO TO 25\$ IF NO ERROR
 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
 JMP 350\$;GO TO 350\$ IF ERROR

2928 034336
 2929 034336 012737 071626 001174
 2930 034344 012737 000200 001176
 2931 034352 004737 043344
 2932
 2933

25\$:
 MOV #MIXED,\$TMP0 ;STARTING ADDRESS OF PATTERN
 MOV #128.,\$TMP1 ;RANGE OF PATTERN
 JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT

2934 034356 012702 001551

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
 MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION

```

2935 034362 112722 000034      MOVB  #RMDC,(R2)+
2936 034366 112722 000006      MOVB  #RMDA,(R2)+
2937 034372 112722 000004      MOVB  #RMBA,(R2)+
2938 034376 112722 000002      MOVB  #RMWC,(R2)+
2939 034402 112722 000032      MOVB  #RMOF,(R2)+
2940 034406 112722 000000      MOVB  #RMCS1,(R2)+
2941 034412 112712 000200      MOVB  #200,(R2)          ;TERMINATE TABLE
2942 034416
2943 034416 004737 044500      30$: JSR  PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR  40$              ;GO TO 40$ IF NO ERROR
      NOP                  ;RETURN HERE IF ERROR
      EMT                  ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP 350$             ;GO TO 350$ IF ERROR
      034422 000404
      034424 000240
      034426 104000
      034430 000137 035212
2944 034434
2945
2946      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR  PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2947 034434 004737 044144
2948      ;WAIT FOR COMMAND TO COMPLETE
      JSR  PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
2949 034440 004737 045042
2950      ;GO READ STATUS FOR FORMAT OPERATION
2951 034444 004737 044230      JSR  PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR  50$              ;GO TO 50$ IF NO ERROR
      NOP                  ;RETURN HERE IF ERROR
      EMT                  ;ERROR # DEFINED BY GET SUBROUTINE
      JMP 350$             ;GO TO 350$ IF ERROR
      034450 000404
      034452 000240
      034454 104000
      034456 000137 035212
2952 034462
2953
2954      ;VERIFY NO ERRORS DURING FORMAT
2955 034462 004737 057742      JSR  PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      BR  60$              ;GO TO 60$ IF NO ERROR
      NOP                  ;RETURN HERE IF ERROR
      EMT                  ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR  PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      JMP 350$             ;GO TO 350$ IF ERROR
      034466 000405
      034470 000240
      034472 104000
      034474 004736
      034476 000137 035212
2956 034502
2957
2958      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
2959 034502 012737 034512 001124  MOV  #70$,$LPERR
2960 034510 000410      BR  80$              ;SKIP TO WRITE OPERATION
2961
2962      ;*****
2963      ;LOOP #2      WRITE,READ
2964
2965 034512
2966
2967      70$:
2968 034512 004737 040472      ;PREPARE DEVICE FOR WRITE OPERATION
      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
    
```

```

034520 000404          BR      80$          ;GO TO 80$ IF NO ERROR
034522 000240          NOP          ;RETURN HERE IF ERROR
034524 104000          EMT          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
034526 000137 035212  JMP      350$         ;GO TO 350$ IF ERROR
2969 034532          80$:
2970
2971          ;WRITE DATA TO THE DRIVE
2972 034532          120$:
2973 034532 012737 177400 001412  MOV     #-256.,RMWCO ;CHANGE WORD COUNT
2974 034540 012737 106320 001414  MOV     #BUFONE+4,RMBAO ;CHANGE ADDRESS
2975 034546 012737 000061 001410  MOV     #WD!GO,RMCS10 ;WRITE DATA COMMAND
2976 034554 012702 001551          MOV     #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
2977 034560 112722 000006  MOVB   #RMDA,(R2)+
2978 034564 112722 000034  MOVB   #RMDC,(R2)+
2979 034570 112722 000032  MOVB   #RMOF,(R2)+
2980 034574 112722 000004  MOVB   #RMBA,(R2)+
2981 034600 112722 000002  MOVB   #RMWC,(R2)+
2982 034604 112722 000000  MOVB   #RMCS1,(R2)+
2983 034610 112722 000200  MOVB   #200,(R2)+ ;TERMINATE TABLE
2984
2985 034614 004737 044500  JSR     PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
034620 000404          BR      130$         ;GO TO 130$ IF NO ERROR
034622 000240          NOP          ;RETURN HERE IF ERROR
034624 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
034626 000137 035212  JMP      350$         ;GO TO 350$ IF ERROR
2986 034632          130$:
2987
2988          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
034632 004737 044144  JSR     PC,GETSTS ;GO TO GETSTS SUBROUTINE
2989
2990          ;WAIT FOR COMMAND TO COMPLETE
034636 004737 045042  JSR     PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
2991
2992          ;GO READ STATUS FOR WRITE COMMAND
2993 034642 004737 044230  JSR     PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
034646 000404          BR      140$         ;GO TO 140$ IF NO ERROR
034650 000240          NOP          ;RETURN HERE IF ERROR
034652 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
034654 000137 035212  JMP      350$         ;GO TO 350$ IF ERROR
2994 034660          140$:
2995
2996          ;CHECK FOR ERRORS DURING WRITE OPERATION
2997 034660 004737 045226  JSR     PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
034664 000405          BR      150$         ;GO TO 150$ IF NO ERROR
034666 000240          NOP          ;RETURN HERE IF ERROR
034670 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
034672 004736  JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034674 000137 035212  JMP      350$         ;GO TO 350$ IF ERROR
2998 034700          150$:
2999 034700 004737 057742  JSR     PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
034704 000405          BR      160$         ;GO TO 160$ IF NO ERROR
034706 000240          NOP          ;RETURN HERE IF ERROR
034710 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
034712 004736  JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034714 000137 035212  JMP      350$         ;GO TO 350$ IF ERROR
3000 034720          160$:
3001 034720 004737 046060  JSR     PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
  
```

```

034724 000405 BR 180$ ;GO TO 180$ IF NO ERROR
034726 000240 NOP ;RETURN HERE IF ERROR
034730 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
034732 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
034734 000137 035212 JMP 350$ ;GO TO 350$ IF ERROR
3002 034740 180$:
3003
3004 ;CHANGE LOOP ADDRESSES
3005 034740 012737 034750 001124 MOV #190$,$LPERR
3006 034746 000410 BR 200$ ;SKIP TO NEXT OPERATION
3007
3008 ;*****
3009 ;LOOP #3 READ
3010
3011 034750 190$:
3012 ;PREPARE DEVICE FOR READ OPERATION
3013
3014 034750 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
034754 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 200$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR
3015 034756 000404 BR 200$
3016 034760 000240 NOP
3017 034762 104000 EMT
3018 034764 000137 035212 JMP 350$
3019 034770 200$:
3020 ;READ DATA FROM DEVICE
3021 240$:
3022 MOV #BUFTWO+4,RMBAO ;CHANGE BUFFER
3023 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
3024 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
3025 MOVB #RMDA,(R2)+
3026 MOVB #RMOF,(R2)+
3027 MOVB #RMDC,(R2)+
3028 MOVB #RMBA,(R2)+
3029 MOVB #RMWC,(R2)+
3030 MOVB #RMCS1,(R2)+
3031 MOVB #200,(R2)
3032 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
3033 BR 250$ ;GO TO 250$ IF NO ERROR
3034 NOP ;RETURN HERE IF ERROR
3035 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
3036 JMP 350$ ;GO TO 350$ IF ERROR
3037 250$:
3038 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
3039 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
3040
3041 ;WAIT FOR COMMAND TO COMPLETE
3042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
    
```

```

3036
3037
3038 035072 004737 044230      ;GO READ STATUS FOR READ OPERATION
      JSR    PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR    260$             ;GO TO 260$ IF NO ERROR
      NOP
      EMT
      JMP    350$             ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 350$ IF ERROR
3039 035110 035212      260$:
3040
3041
3042 035110 004737 045226      ;CHECK FOR ERRORS DURING READ OPERATION
      JSR    PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      BR    270$             ;GO TO 270$ IF NO ERROR
      NOP
      EMT
      JSR    PC,@(SP)+        ;RETURN HERE IF ERROR
      JMP    350$             ;ERROR # DEFINED BY PRIERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
3043 035130 035212      270$:
3044 035130 004737 057742      JSR    PC,DTASTS            ;GO VERIFY RESULTS OF DATA TRANSFER
      BR    280$             ;GO TO 280$ IF NO ERROR
      NOP
      EMT
      JSR    PC,@(SP)+        ;RETURN HERE IF ERROR
      JMP    350$             ;ERROR # DEFINED BY DTASTS SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
3045 035150 035212      280$:
3046 035150 004737 046060      JSR    PC,SECERR           ;GO CHECK FOR SECONDARY ERRORS
      BR    300$             ;GO TO 300$ IF NO ERROR
      NOP
      EMT
      JSR    PC,@(SP)+        ;RETURN HERE IF ERROR
      JMP    350$             ;ERROR # DEFINED BY SECERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
3047 035170 035212      300$:
3048 035170 004737 043602      JSR    PC,CMPBUF           ;GO COMPARE WRITE, READ DATA BUFFERS
      .WORD  BUFOFF+4        ;STARTING ADDRESS OF WRITE BUFFER
      .WORD  BUFTWO+4        ;STARTING ADDRESS OF READ BUFFER
      BR    310$             ;GO TO 310$ IF NO ERROR
      NOP
      EMT
      JMP    350$             ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      ;GO TO 350$ IF ERROR
3049 035212 035212      310$:
3050
3051 035212 035212      350$:
3052
3053
      ;*****
      ;*TEST 26 WRITE, READ EACH TRACK
      ;*****
      TST26:
      SCOPE                   ;SCOPE CALL
      NOP                     ;START OF TEST
      MOV    #STACK,SP        ;INITIALIZE STACK POINTER
      MOV    $BASE,R0         ;R0 = UNIBUS ADDRESS
      MOV    TSTQUE,R1        ;(R1) = DEVICE BEING TESTED
      MOV    #26,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
3054
3055
3056
3057
3058 035240 035240      ;*****
      ;LOOP #1 FORMAT,WRITE,READ
      10$:
  
```

3059
3060
3061 035240 004737 040472
035244 154130

;PREPARE THE DEVICE FOR FORMAT OPERATION

JSR PC,TSTPRP
.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 20\$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350\$ IF ERROR

035246 000404
035250 000240
035252 104000
035254 000137 036240
3062 035260

BR 20\$
NOP
EMT
JMP 350\$

20\$:

3063
3064
3065 035260 012737 000000 001444
3066 035266 012737 000000 001416
3067 035274
3068 035274 012737 106314 001414
3069 035302 012737 177376 001412
3070 035310 012737 010000 001442
3071 035316 012737 000063 001410
3072
3073

;LOAD PARAMETERS AND GENERATE DATA BUFFER

MOV #0,RMDCO
MOV #0,RMDAO

;CYLINDER = 0
;TRACK = 0, SECTOR = 0

25\$:

MOV #BUFONE,RMBAO
MOV #-258.,RMWCO
MOV #FMT16,RMOFO
MOV #WH!GO,RMCS10

;BUS ADDRESS
;2 + 256. WORDS (2'S COMP)
;16 BIT FORMAT
;WRITE HEADER AND DATA COMMAND

035324 004737 041416
035330 000405
035332 104401 070230
035336 104000
035340 000137 036240
3074 035344

;VERIFY THAT SECTOR IS NOT BAD

JSR PC,BADSCT
BR 26\$
TYPE ,SCTMSG
EMT
JMP 350\$

;CALL BAD SECTOR MODULE
;GO TO 26\$ IF NO ERROR
;TYPE BAD SECTOR MESSAGE
;ERROR # DEFINED BY BADSCT SUBROUTINE
;GO TO 350\$ IF ERROR

26\$:

3075 035344 012737 071626 001174
3076 035352 012737 000200 001176
3077 035360 004737 043344
3078
3079

MOV #MIXED,\$TMP0
MOV #128.,\$TMP1
JSR PC,GENBUF

;STARTING ADDRESS OF PATTERN
;RANGE OF PATTERN
;GO GENERATE BUFFER FOR FORMAT

035364 012702 001551
035370 112722 000034
035374 112722 000006
035400 112722 000004
035404 112722 000002
035410 112722 000032
035414 112722 000000
035420 112712 000200
035424
3089
3090

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION

MOV #PUTINX,R2
MOVB #RMDC,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)

;R2 = BYTE ENTRY POSITION

30\$:

;TERMINATE TABLE

035424 004737 044500
035430 000404
035432 000240
035434 104000
035436 000137 036240
3092 035442
3093
3094

;FORMAT THE DRIVE

JSR PC,PUT
BR 40\$
NOP
EMT
JMP 350\$

;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
;GO TO 40\$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY PUT SUBROUTINE
;GO TO 350\$ IF ERROR

40\$:

;SETUP GET INDEX TABLE TO READ ALL REGISTERS

```

3095 035442 004737 044144 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
3096 ;WAIT FOR COMMAND TO COMPLETE
035446 004737 045042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
3097 ;GO READ STATUS FOR FORMAT OPERATION
3098
3099 035452 004737 044230 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
035456 000404 BR 50$ ;GO TO 50$ IF NO ERROR
035460 000240 NOP ;RETURN HERE IF ERROR
035462 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
035464 000137 036240 JMP 350$ ;GO TO 350$ IF ERROR
3100 035470 50$:
3101
3102 ;VERIFY NO ERRORS DURING FORMAT
3103 035470 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
035474 000405 BR 60$ ;GO TO 60$ IF NO ERROR
035476 000240 NOP ;RETURN HERE IF ERROR
035500 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
035502 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
035504 000137 036240 JMP 350$ ;GO TO 350$ IF ERROR
3104 035510 60$:
3105
3106 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
3107 035510 012737 035520 001124 MOV #70$,$LPERR
3108 035516 000410 BR 80$ ;SKIP TO WRITE OPERATION
3109
3110 ;*****
3111 ;LOOP #2 WRITE,READ
3112
3113 035520 70$:
3114
3115 ;PREPARE DEVICE FOR WRITE OPERATION
3116 035520 004737 040472 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
035524 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 80$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR
035526 000404 BR 80$
035530 000240 NOP
035532 104000 EMT
035534 000137 036240 JMP 350$
3117 035540 80$:
3118
3119 ;WRITE DATA TO THE DRIVE
3120 035540 120$:
3121 035540 012737 106320 001414 MOV #BUFONE+4,RMBAO ;CHANGE BUS ADDRESS
3122 035546 012737 177400 001412 MOV #-256.,RMWCO ;CHANGE WORD COUNT
3123 035554 012737 000061 001410 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
3124 035562 012702 001551 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
3125 035566 112722 000006 MOVB #RMDA,(R2)+
3126 035572 112722 000034 MOVB #RMDC,(R2)+
3127 035576 112722 000032 MOVB #RMOF,(R2)+
3128 035602 112722 000004 MOVB #RMBA,(R2)+

```

```

3129 035606 112722 000002      MOVB  #RMWC,(R2)+
3130 035612 112722 000000      MOVB  #RMCS1,(R2)+
3131 035616 112722 000200      MOVB  #200,(R2)+
3132
3133 035622 004737 044500      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      035626 000404      BR    130$       ;GO TO 130$ IF NO ERROR
      035630 000240      NOP
      035632 104000      EMT             ;RETURN HERE IF ERROR
      035634 000137 036240      JMP   350$       ;ERROR # DEFINED BY PUT SUBROUTINE
3134 035640      130$:           ;GO TO 350$ IF ERROR
3135
3136      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      035640 004737 044144      JSR   PC,GETSTS  ;GO TO GETSTS SUBROUTINE
3137
3138      ;WAIT FOR COMMAND TO COMPLETE
      035644 004737 045042      JSR   PC,TIMOUT  ;GO TO TIMOUT SUBROUTINE
3139
3140      ;GO READ STATUS FOR WRITE COMMAND
3141 035650 004737 044230      JSR   PC,GET     ;GO READ REGISTER(S) WITH GET SUBROUTINE
      035654 000404      BR    140$       ;GO TO 140$ IF NO ERROR
      035656 000240      NOP             ;RETURN HERE IF ERROR
      035660 104000      EMT             ;ERROR # DEFINED BY GET SUBROUTINE
      035662 000137 036240      JMP   350$       ;GO TO 350$ IF ERROR
3142 035666
3143      140$:
3144      ;CHECK FOR ERRORS DURING WRITE OPERATION
3145 035666 004737 045226      JSR   PC,PRIERR  ;GO CHECK FOR PRIMARY ERRORS
      035672 000405      BR    150$       ;GO TO 150$ IF NO ERROR
      035674 000240      NOP             ;RETURN HERE IF ERROR
      035676 104000      EMT             ;ERROR # DEFINED BY PRIERR SUBROUTINE
      035700 004736      JSR   PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035702 000137 036240      JMP   350$       ;GO TO 350$ IF ERROR
3146 035706      150$:
3147 035706 004737 057742      JSR   PC,DTASTS  ;GO VERIFY RESULTS OF DATA TRANSFER
      035712 000405      BR    160$       ;GO TO 160$ IF NO ERROR
      035714 000240      NOP             ;RETURN HERE IF ERROR
      035716 104000      EMT             ;ERROR # DEFINED BY DTASTS SUBROUTINE
      035720 004736      JSR   PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035722 000137 036240      JMP   350$       ;GO TO 350$ IF ERROR
3148 035726      160$:
3149 035726 004737 046060      JSR   PC,SECERR  ;GO CHECK FOR SECONDARY ERRORS
      035732 000405      BR    180$       ;GO TO 180$ IF NO ERROR
      035734 000240      NOP             ;RETURN HERE IF ERROR
      035736 104000      EMT             ;ERROR # DEFINED BY SECERR SUBROUTINE
      035740 004736      JSR   PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      035742 000137 036240      JMP   350$       ;GO TO 350$ IF ERROR
3150 035746      180$:
3151
3152      ;CHANGE LOOP ADDRESSES
3153 035746 012737 035756 001124      MOV   #190$,$LPERR
3154 035754 000410      BR    200$
3155
3156      ;*****
3157      ;LOOP #3      READ
3158
3159 035756      190$:
3160

```



```

3161                                     ;PREPARE DEVICE FOR READ OPERATION
3162 035756 004737 040472                JSR   PC,TSTPRP                ;PREPARE DEVICE FOR TEST
      035762 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
      035764 000404                        BR    200$                      ;GO TO 200$ IF NO ERROR
      035766 000240                        NOP                                     ;RETURN HERE IF ERROR
      035770 104000                        EMT                                     ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      035772 000137 036240                JMP   350$                      ;GO TO 350$ IF ERROR
3163 035776
3164
3165                                     ;READ DATA FROM DEVICE
3166 035776 240$:
3167 035776 012737 107324 001414         MOV   #BUFTWO+4,RMBAO          ;CHANGE BUS ADDRESS
3168 036004 012737 000071 001410         MOV   #RD!GO,RMCS10           ;READ DATA COMMAND
3169 036012 012702 001551                 MOV   #PUTINX,R2              ;LOAD PUT REGISTER INDEX TABLE
3170 036016 112722 000006                 MOVB  #RMDA,(R2)+
3171 036022 112722 000032                 MOVB  #RMOF,(R2)+
3172 036026 112722 000034                 MOVB  #RMDC,(R2)+
3173 036032 112722 000004                 MOVB  #RMBA,(R2)+
3174 036036 112722 000002                 MOVB  #RMWC,(R2)+
3175 036042 112722 000000                 MOVB  #RMCS1,(R2)+
3176 036046 112712 000200                 MOVB  #200,(R2)
3177
3178 036052 004737 044500                JSR   PC,PUT                   ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      036056 000404                        BR    250$                      ;GO TO 250$ IF NO ERROR
      036060 000240                        NOP                                     ;RETURN HERE IF ERROR
      036062 104000                        EMT                                     ;ERROR # DEFINED BY PUT SUBROUTINE
      036064 000137 036240                JMP   350$                      ;GO TO 350$ IF ERROR
3179 036070
3180
3181                                     ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
3182 036070 004737 044144                JSR   PC,GETSTS                ;GO TO GETSTS SUBROUTINE
3183
3184                                     ;WAIT FOR COMMAND TO COMPLETE
3185 036074 004737 045042                JSR   PC,TIMOUT                ;GO TO TIMOUT SUBROUTINE
3186
3187                                     ;GO READ STATUS FOR READ OPERATION
3188 036100 004737 044230                JSR   PC,GET                   ;GO READ REGISTER(S) WITH GET SUBROUTINE
      036104 000404                        BR    260$                      ;GO TO 260$ IF NO ERROR
      036106 000240                        NOP                                     ;RETURN HERE IF ERROR
      036110 104000                        EMT                                     ;ERROR # DEFINED BY GET SUBROUTINE
      036112 000137 036240                JMP   350$                      ;GO TO 350$ IF ERROR
3189 036116
3190                                     ;CHECK FOR ERRORS DURING READ OPERATION
      036122 000405                        JSR   PC,PRIERR                ;GO CHECK FOR PRIMARY ERRORS
      036124 000240                        BR    270$                      ;GO TO 270$ IF NO ERROR
      036126 104000                        NOP                                     ;RETURN HERE IF ERROR
      036130 004736                        EMT                                     ;ERROR # DEFINED BY PRIERR SUBROUTINE
      036132 000137 036240                JSR   PC,@(SP)+                ;GO BACK FOR MORE ERROR CHECKS
      JMP   350$                      ;GO TO 350$ IF ERROR

```

```

3191 036136          270$:
3192 036136 004737 057742      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      036142 000405          BR      280$          ;GO TO 280$ IF NO ERROR
      036144 000240          NOP
      036146 104000          EMT      ;RETURN HERE IF ERROR
      036150 004736          JSR    PC,@(SP)+    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      036152 000137 036240    JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
3193 036156          280$:
3194 036156 004737 046060      JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      036162 000405          BR      300$          ;GO TO 300$ IF NO ERROR
      036164 000240          NOP
      036166 104000          EMT      ;RETURN HERE IF ERROR
      036170 004736          JSR    PC,@(SP)+    ;ERROR # DEFINED BY SECERR SUBROUTINE
      036172 000137 036240    JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 350$ IF ERROR
3195 036176          300$:
3196 036176 004737 043602      JSR    PC,CMPBUF     ;GO COMPARE WRITE, READ DATA BUFFERS
      036202 106320          .WORD  BUFOFF+4      ;STARTING ADDRESS OF WRITE BUFFER
      036204 107324          .WORD  BUFOFF+4      ;STARTING ADDRESS OF READ BUFFER
      036206 000404          BR      310$          ;GO TO 310$ IF NO ERROR
      036210 000240          NOP
      036212 104000          EMT      ;RETURN HERE IF ERROR
      036214 000137 036240    JMP      350$          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      ;GO TO 350$ IF ERROR
3197 036220          310$:
3198 036220 105237 001417      INCB   RMDAO+1       ;ADVANCE TO NEXT TRACK
3199 036224 123737 001417 001333 CMPB   RMDAO+1,LSTRK+1 ;DONE ?
3200 036232 101002          BHI    350$          ;YES!!
3201 036234 000137 035274      JMP      25$          ;TEST NEXT TRACK
3202 036240          350$:
3203
3204

```

```

*****
;*TEST 27 READ, WRITE CHECK MULTIPLE SECTORS IN OFFSET MODE
*****
TST27:

```

```

036240          SCOPE          ;SCOPE CALL
036240 000004          NOP          ;START OF TEST
036242 000240          MOV    #STACK,SP    ;INITIALIZE STACK POINTER
036244 012706 001100      MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
036250 013700 001276      MOV    TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
036254 013701 001464      MOV    #27,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
036260 012737 000027 001226

```

```

*****
;LOOP #1 FORMAT,READ OFFSET,WRITE CHECK OFFSET IN BOTH DIRECTIONS
10$:

```

```

;PREPARE DEVICE FOR FORMAT OPERATION
3205
3206
3207
3208 036266
3209
3210
3211 036266 004737 040472      JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      036272 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 20$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      BR      20$
      NOP
      EMT
036274 000404
036276 000240
036300 104000

```



```

036526 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
036530 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
036532 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR

3251
3252 ;MOV LOOP ADDRESSES TO NEXT OPERATION
3253 036536 60$:
3254 036536 012737 036560 001124 MOV #70$, $LPERR ;ERROR LOOP ADDRESS
3255
3256 ;LOCATE BUFFER ADDRESS AND WORD COUNT
3257 036544 012737 177000 001412 MOV #-256.*2, RMWCO ;TWO SECTORS
3258 036552 012737 106314 001414 MOV #BUFONE, RMBAO ;BUFFER ADDRESS
3259
3260 ;*****
3261 036560 ;LOOP 2 READ AND WRITE CHECK IN OFFSET MODE
3262 70$:
3263 036560 004737 040472 ;PREPARE DEVICE FOR READ OFFSET OPEATION
036564 154130 JSR PC, TSTPRP ;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 80$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR

036566 000404 BR 80$
036570 000240 NOP
036572 104000 EMT
036574 000137 037422 JMP 350$

3264 036600 80$:
3265 036600 120$:
3266 ;READ OFFSET
3267 036600 012702 001551 MOV #PUTINX, R2 ;LOAD THE TABLE
3268 036604 112722 000006 MOVB #RMDA, (R2)+
3269 036610 112722 000034 MOVB #RMDC, (R2)+
3270 036614 112722 000032 MOVB #RMOF, (R2)+
3271 036620 112722 000004 MOVB #RMBA, (R2)+
3272 036624 112722 000002 MOVB #RMWC, (R2)+
3273 036630 112722 000200 MOVB #200, (R2)+ ;TABLE TERMINATOR
3274
3275 036634 004737 044500 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
036640 000404 BR 125$ ;GO TO 125$ IF NO ERROR
036642 000240 NOP ;RETURN HERE IF ERROR
036644 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
036646 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR
3276 036652 012737 000015 001410 125$: MOV #OFFSET!GO, RMCS10 ;OFFSET COMMAND
3277 036660 012702 001551 MOV #PUTINX, R2
3278 036664 112722 000000 MOVB #RMCS1, (R2)+
3279 036670 112722 000200 MOVB #200, (R2)+

3281 036674 004737 044500 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
036700 000404 BR 130$ ;GO TO 130$ IF NO ERROR
036702 000240 NOP ;RETURN HERE IF ERROR
036704 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
036706 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR

3282 036712 130$:
3283
3284 ;WAIT FOR COMMAND TO COMPLETE

```

```

3285 036712 004737 045042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
3286 036716 012737 000071 001410 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
3287 036724 012702 001551 MOV #PUTINX,R2 ;TABLE INDEX
3288 036730 112722 000000 MOVB #RMCS1,(R2)+
3289 036734 112712 000200 MOVB #200,(R2) ;TABLE TERMINATOR
3290
3291 036740 004737 044500 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
036744 000404 BR 135$ ;GO TO 135$ IF NO ERROR
036746 000240 NOP ;RETURN HERE IF ERROR
036750 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
036752 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR
3292 036756 135$:
3293
3294 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
036756 004737 044144 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
3295
3296 ;WAIT FOR COMMAND TO COMPLETE
036762 004737 045042 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
3297
3298 ;READ STATUS FOR READ OFFSET OPERATION
3299 036766 004737 044230 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
036772 000404 BR 140$ ;GO TO 140$ IF NO ERROR
036774 000240 NOP ;RETURN HERE IF ERROR
036776 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
037000 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR
3300
3301 ;CHECK ERROR DURING READ OFFSET
3302 037004 140$:
3303 037004 004737 045226 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
037010 000405 BR 150$ ;GO TO 150$ IF NO ERROR
037012 000240 NOP ;RETURN HERE IF ERROR
037014 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
037016 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
037020 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR
3304 037024 150$:
3305 037024 004737 057742 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
037030 000405 BR 160$ ;GO TO 160$ IF NO ERROR
037032 000240 NOP ;RETURN HERE IF ERROR
037034 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
037036 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
037040 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR
3306 037044 160$:
3307 037044 004737 046060 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
037050 000405 BR 180$ ;GO TO 180$ IF NO ERROR
037052 000240 NOP ;RETURN HERE IF ERROR
037054 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
037056 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
037060 000137 037422 JMP 350$ ;GO TO 350$ IF ERROR
3308 037064 180$:
3309
3310 ;CHANGE LOOP ADDRESS
3311 037064 012737 037074 001124 MOV #190$,$LPERR
3312 037072 000410 BR 200$ ;TO NEXT OPERATION
3313
3314
3315 ;*****
;LOOP 3 WRITE CHECK OFFSET

```

```

3316 037074          190$:
3317
3318
3319 037074 004737 040472      ;PREPARE DEVICE FOR READ OPERATION
      037100 154130          JSR   PC,TSTPRP      ;PREPARE DEVICE FOR TEST
                                      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                      ;CLEAR CONTROLLER & SELECT DEVICE
                                      ;VERIFY CONTROLLER CLEAR OPERATION
                                      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                      ;VERIFY PACK ACKNOWLEDGE
                                      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                      ;VERIFY RECALIBRATION
                                      ;GO TO 200$ IF NO ERROR
                                      ;RETURN HERE IF ERROR
                                      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                      ;GO TO 350$ IF ERROR
      037102 000404          BR    200$
      037104 000240          NOP
      037106 104000          EMT
      037110 000137 037422    JMP   350$
3320 037114          200$:
3321
3322      ;WRITE CHECK DATA IN OFFSET MODE
3323 037114          240$:
3324 037114 012702 001551      MOV   #PUTINX,R2
3325 037120 112722 000006      MOVB  #RMDA,(R2)+
3326 037124 112722 000032      MOVB  #RMOF,(R2)+
3327 037130 112722 000034      MOVB  #RMDC,(R2)+
3328 037134 112722 000004      MOVB  #RMBA,(R2)+
3329 037140 112722 000002      MOVB  #RMWC,(R2)+
3330 037144 112712 000200      MOVB  #200,(R2)      ;TABLE TERMINATOR
3331
3332 037150 004737 044500      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      037154 000404          BR    245$          ;GO TO 245$ IF NO ERROR
      037156 000240          NOP                ;RETURN HERE IF ERROR
      037160 104000          EMT                ;ERROR # DEFINED BY PUT SUBROUTINE
      037162 000137 037422    JMP   350$          ;GO TO 350$ IF ERROR
3333
3334      ;SETUP OFFSET MODE FIRST
3335 037166          245$:
3336 037166 012737 000015 001410  MOV   #OFFSET!GO,RMCS10 ;OUTPUT BUFFER
3337 037174 012702 001551      MOV   #PUTINX,R2
3338 037200 112722 000000      MOVB  #RMCS1,(R2)+
3339 037204 112712 000200      MOVB  #200,(R2)
3340
3341 037210 004737 044500      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      037214 000404          BR    250$          ;GO TO 250$ IF NO ERROR
      037216 000240          NOP                ;RETURN HERE IF ERROR
      037220 104000          EMT                ;ERROR # DEFINED BY PUT SUBROUTINE
      037222 000137 037422    JMP   350$          ;GO TO 350$ IF ERROR
3342 037226          250$:
3343
3344      ;WAIT FOR COMMAND TO COMPLETE
      037226 004737 045042    JSR   PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
3345
3346      ;EXECUTE WRITE CHECK DATA COMMAND
3347 037232 012737 000051 001410  MOV   #WCD!GO,RMCS10
3348 037240 012702 001551      MOV   #PUTINX,R2
3349 037244 112722 000000      MOVB  #RMCS1,(R2)+
3350 037250 112712 000200      MOVB  #200,(R2)
3351
  
```

```

3352 037254 004737 044500      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      037260 000404      BR    255$        ;GO TO 255$ IF NO ERROR
      037262 000240      NOP                    ;RETURN HERE IF ERROR
      037264 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      037266 000137 037422      JMP   350$        ;GO TO 350$ IF ERROR
3353 037272      255$:
3354
3355      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      037272 004737 044144      JSR   PC,GETSTS   ;GO TO GETSTS SUBROUTINE
3356
3357      ;WAIT FOR COMMAND TO COMPLETE
      037276 004737 045042      JSR   PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
3358
3359      ;READ STATUS FOR WRITE CHECK OFFSET OPERATION
3360 037302 004737 044230      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      037306 000404      BR    260$        ;GO TO 260$ IF NO ERROR
      037310 000240      NOP                    ;RETURN HERE IF ERROR
      037312 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      037314 000137 037422      JMP   350$        ;GO TO 350$ IF ERROR
3361 037320      260$:
3362
3363      ;CHECK ERROR DURING WRITE CHECK OFFSET OPERATION
3364 037320 004737 045226      JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      037324 000405      BR    270$        ;GO TO 270$ IF NO ERROR
      037326 000240      NOP                    ;RETURN HERE IF ERROR
      037330 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      037332 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      037334 000137 037422      JMP   350$        ;GO TO 350$ IF ERROR
3365 037340      270$:
3366 037340 004737 057742      JSR   PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      037344 000405      BR    280$        ;GO TO 280$ IF NO ERROR
      037346 000240      NOP                    ;RETURN HERE IF ERROR
      037350 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      037352 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      037354 000137 037422      JMP   350$        ;GO TO 350$ IF ERROR
3367 037360      280$:
3368 037360 004737 046060      JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      037364 000405      BR    300$        ;GO TO 300$ IF NO ERROR
      037366 000240      NOP                    ;RETURN HERE IF ERROR
      037370 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      037372 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      037374 000137 037422      JMP   350$        ;GO TO 350$ IF ERROR
3369 037400      300$:
3370 037400 032737 000200 001442      BIT   #OFD,RMOFO  ;IN FORWARD DIRECTION OF OFFSET ?
3371 037406 001005      BNE   350$        ;BRANCH IF SO
3372 037410 052737 000200 001442      BIS   #OFD,RMOFO  ;SET TO FORWARD DIRECTION
3373 037416 000137 036536      JMP   60$         ;CHANGE OFFSET DIRECTION TEST AGAIN
3374 037422      350$:
3375
  
```

```

1      .SBTTL  END OF SUB-PASS ROUTINE
2
3      ;THIS IS THE END OF SUB-PASS ROUTINE. THIS ROUTINE IS USED TO
4      ;TERMINATE THE OPERATION OF THE CURRENT DEVICE UNDER TEST AND
5      ;SELECT THE NEXT DEVICE FOR TEST. IF THERE ARE NO MORE DEVICES
6      ;TO TEST, EXIT IS MADE TO '$EOP' ROUTINE. OTHERWISE, RETURN
7      ;IS MADE TO 'READY' ROUTINE.
8
9 037422 000004      $EOSP:  SCOPE
10 037424 000240      NOP
11 037426 013700 001464      MOV      TSTQUE,R0      ;GET POINTER TO TSTQUE
12 037432 062700 000002      ADD      #2,R0      ;ADJUST POINTER TO NEXT DEVICE
13 037436 010037 001464      MOV      R0,TSTQUE      ;SAVE POINTER TO TSTQUE
14 037442 005710      TST      (R0)      ;ANY MORE DEVICES FOR TEST ?
15 037444 001402      BEQ      1$      ;BR IF NO
16 037446 000137 064076      JMP      SHUT      ;JUMP TO SHUT AND CHECK FOR CONTROL C
17 037452 012737 001466 001464 1$:  MOV      #TSTQUE+2,TSTQUE ;INITIALIZE POINTER TO FIRST DEVICE IN
18                                     ;TEST QUE TABLE
19
20      .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

```

```

037460      $EOP:
037460 000240      NOP
037462 005037 001116      CLR      $TSTNM      ;;ZERO THE TEST NUMBER
037466 005037 001206      CLR      $TIMES      ;;ZERO THE NUMBER OF ITERATIONS
037472 005237 001230      INC      $PASS      ;;INCREMENT THE PASS NUMBER
037476 042737 100000 001230      BIC      #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
037504 005327      DEC      (PC)+      ;;LOOP?
037506 000001      $EOPCT: .WORD 1
037510 003063      BGT      $DOAGN      ;;YES
037512 012737      MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER
037514 000001      $ENDCT: .WORD 1
037516 037506      TYPE      ,65$      ;;TYPE ASCIZ STRING
037520 104401 037526      BR      64$      ;;GET OVER THE ASCIZ
037524 000407      ;;65$: .ASCIZ <12><15>/END PASS #/
037544      64$:
037544 013746 001230      MOV      $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
037550 104405      ;;TYPE PASS NUMBER
037552 104401 037560      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
037556 000421      TYPE      ,67$      ;;TYPE ASCIZ STRING
037622      BR      66$      ;;GET OVER THE ASCIZ
037622 013746 001126      ;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
037626 104405      66$:
037630 104401 001217      MOV      $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
037634 005037 001126      ;;TOTAL NUMBER OF ERRORS
037640 013700 000042      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                TYPE      , $CRLF      ;;TYPE CARRIAGE RETURN, LINE FEED
                                CLR      $ERTTL      ;;CLEAR ERROR TOTAL
                                $GET42: MOV      @#42,R0 ;;GET MONITOR ADDRESS

```


037644	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
037646	000005			RESET		::CLEAR THE WORLD
037650	004710			\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
037652	000240			NOP		::SAVE ROOM
037654	000240			NOP		::FOR
037656	000240			NOP		::ACT11
037660				\$DOAGN:		
037660	000137			JMP	@(PC)+	::RETURN
037662	064076			\$RTNAD: .WORD	SHUT	
037664	377	377	000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
				.EVEN		

```

1      .SBTTL SUBROUTINES
2      :*****
3      .SBTTL ERROR TYPEOUT ROUTINE
4
5      :*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
6      :*REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
7      :*
8      :*      .UNIT NUMBER, TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER ARE
9      :*PRINTED ON THE FIRST LINE;
10     :*      .ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
11     :*ONE OR MORE SUCCEEDING LINES;
12     :*      .PAIRED LINES OF ERROR HEADERS AND ERROR DATA
13     :*ARE PRINTED AFTER THE ERROR MESSAGE.
14
15     037670
16     037670 104414
17     037672 032777 020000 141254
18     037700 001402
19     037702 000137 040420
20
21     037706 104401 001217
22     037712 104401 040434
23     037716 013746 001234
24
25     037722 104403
26     037724 003
27     037725 000
28     037726 005037 040424
29     037732 013737 001226 040424
30     037740 104401 040442
31     037744 013746 040424
32
33     037750 104403
34     037752 003
35     037753 000
36     037754 005037 040426
37     037760 113737 001130 040426
38     037766 001406
39     037770 104401 040452
40     037774 013746 040426
41
42     040000 104403
43     040002 003
44     040003 000
45     040004 104401 040461
46     040010 013746 001132
47
48     040014 104403
49     040016 006
50     040017 001
51
52     040020 005737 040426
53     040024 001575
54     040026 104401 001217
55     040032 105037 040432
56     040036 105037 040433
57     040042 013700 040426
58
59     SAVREG
60     BIT #SW13,@SWR ;INHIBIT TYPEOUTS??
61     BEQ 1$ ;NO!!
62     JMP 21$ ;YES!!
63
64     1$:
65     TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
66     TYPE , $CRLF
67     TYPE , ERTY00 ;TYPE 'UNT#'
68     MOV $UNIT,-(SP) ;;SAVE $UNIT FOR TYPEOUT
69     ;;TYPE UNIT NUMBER
70     TYPOS ;GO TYPE--OCTAL ASCII
71     .BYTE 3 ;TYPE 3 DIGIT(S)
72     .BYTE 0 ;SUPPRESS LEADING ZEROS
73     CLR TSTNMB ;LOAD TEST NUMBER FOR
74     MOV $TESTN,TSTNMB
75     TYPE , ERTY01 ;TYPE 'TST#'
76     MOV TSTNMB,-(SP) ;;SAVE TSTNMB FOR TYPEOUT
77     ;;TYPE TEST NUMBER
78     TYPOS ;GO TYPE--OCTAL ASCII
79     .BYTE 3 ;TYPE 3 DIGIT(S)
80     .BYTE 0 ;SUPPRESS LEADING ZEROS
81     CLR ERRNMB ;LOAD ERROR NUMBER FOR
82     MOV $ITEMB,ERRNMB ;TYPEOUT
83     BEQ 2$ ;SKIP IF NO ERROR CALLED
84     TYPE , ERTY02 ;TYPE 'ERR#'
85     MOV ERRNMB,-(SP) ;;SAVE ERRNMB FOR TYPEOUT
86     ;;TYPE ERROR NUMBER
87     TYPOS ;GO TYPE--OCTAL ASCII
88     .BYTE 3 ;TYPE 3 DIGIT(S)
89     .BYTE 0 ;SUPPRESS LEADING ZEROS
90     TYPE , ERTY03 ;TYPE 'PC='
91     MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
92     ;;TYPE PROGRAM COUNTER
93     TYPOS ;GO TYPE--OCTAL ASCII
94     .BYTE 6 ;TYPE 6 DIGIT(S)
95     .BYTE 1 ;TYPE LEADING ZEROS
96
97     2$:
98     3$:
99     ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
100    TST ERRNMB ;WAS AN ERROR CALLED?
101    BEQ 21$ ;NO!!
102    TYPE , $CRLF ;YES-TYPE CRLF
103    CLRB BOTFLG ;CLEAR BOT FLAG
104    CLRB CHRCNT ;CLEAR CHARACTER COUNTER
105    MOV ERRNMB,R0 ;R0 POINTS TO FIRST OF

```

```

42 040046 006300          ASL      R0          :FOUR ENTRIES IN ERROR
43 040050 006300          ASL      R0          :TABLE
44 040052 006300          ASL      R0
45 040054 062700 001570   ADD      #SERRTB-8.,R0
46 040060 011001          MOV      (R0),R1     :R1 POINTS TO ERROR MESSAGE
47                                     :TABLE
48 040062 001507          BEQ      13$         :BRANCH IF NO ERROR MESSAGE
49                                     :TYPE THE ERROR MESSAGE
50 040064 012102 4$:     MOV      (R1)+,R2     :R2=ADDRESS OF MESSAGE STRING
51 040066 001505          BEQ      12$         :BRANCH IF END OF MESSAGE
52 040070 010237 040236   MOV      R2,11$     :LOAD ADDRESS OF STRING
53 040074 005037 040430   CLR      BOTADR     :CLEAR BOT ADDRESS
54 040100 112203 5$:     MOV      (R2)+,R3     :END OF STRING??
55 040102 001454          BEQ      10$         :YES!!
56 040104 122703 000015   CMPB    #CR,R3      :CARRIAGE RETURN??
57 040110 001003          BNE      6$         :NO!!
58 040112 105037 040433   CLRB    CHRCNT     :YES-CLEAR CHAR COUNT
59 040116 000770          BR       5$         :GET NEXT CHARACTER
60 040120 122703 000012 6$:     CMPB    #LF,R3      :LINE FEED??
61 040124 001765          BEQ      5$         :YES-GET NEXT CHARACTER
62 040126 122703 000011   CMPB    #HT,R3      :HORIZONTAL TAB??
63 040132 001007          BNE      8$         :NO!!
64 040134 105237 040433 7$:     INCB    CHRCNT     :ADJUST CHARACTER COUNT
65 040140 132737 000007 040433  BITB    #7,CHRCNT
66 040146 001372          BNE      7$
67 040150 000407          BR       9$
68 040152 105237 040433 8$:     INCB    CHRCNT     :INCREMENT CHARACTER COUNT
69 040156 122703 000040   CMPB    #' ,R3      :SPACE??
70 040162 001002          BNE      9$         :NO!!
71 040164 010237 040430   MOV      R2,BOTADR  :SAVE ADDRESS OF SPACE
72 040170 122737 000100 040433 9$:     CMPB    #64.,CHRCNT
73 040176 103340          BHIS    5$         :END OF LINE??
74 040200 013704 040430   MOV      BOTADR,R4  :NO!!
75 040204 001007          BNE      90$        :GET ADDRESS OF LAST SPACE
76 040206 104401 001217   TYPE    ,$CRLF     :BRANCH IF SPACE DETECTED
77 040212 105037 040433   CLRB    CHRCNT     :TYPE CRLF
78 040216 013702 040236   MOV      11$,R2     :CLEAR CHARACTER COUNT
79 040222 000726          BR       5$         :SET UP R2 FOR TESTING
80 040224 105044 90$:    CLRB    -(R4)      :REPLACE SPACE
81 040226 112737 177777 040432  MOV      #-1,BOTFLG :SET BOT FLAG
82 040234 104401 10$:    TYPE    :TYPE ERROR MESSAGE STRING
83 040236 000000 11$:    .WORD   :STRING ADDRESS GOES HERE
84 040240 105737 040432   TSTB   BOTFLG     :WAS STRING TRUNCATED??
85 040244 001707          BEQ      4$         :NO!!
86 040246 104401 001217   TYPE    , $CRLF    :YES-TYPE CRLF
87 040252 105037 040433   CLRB    BOTFLG     :CLEAR BOT FLAG
88 040256 105037 040433   CLRB    CHRCNT     :CLEAR CHARACTER COUNT
89 040262 013702 040430   MOV      BOTADR,R2  :SETUP R2 FOR TESTING
90 040266 010237 040236   MOV      R2,11$     :SETUP 11$ FOR TYPING
91 040272 112742 000040   MOV      #' -(R2)   :RESTORE SPACE
92 040276 105722          TSTB   (R2)+       :RESTORE R2
93 040300 000677          BR       5$         :TYPE REST OF STRING
94 040302 12$:
95 :TYPE ERROR HEADER AND ERROR DATA
96 040302 016001 000002 13$:
97 040302 001444          MOV      2(R0),R1   :R1 POINTS TO ERROR HEADER TABLE
98 040306          BEQ      21$         :BRANCH IF NO HEADER

```

```

99 040310 104401 001217          TYPE      , $CRLF          ; (ASSUME NO DATA)
100 040314 016002 000004          MOV      4(R0),R2        ; R2 POINTS TO DATA ADDRESS TABLE
101 040320 016003 000006          MOV      6(R0),R3        ; R3 POINTS TO FORMAT TABLE
102 040324 012137 040334          MOV      (R1)+,15$      ; PUT HEADER ADDRESS FOR TYPE
103 040330 001433          BEQ      21$            ; BRANCH IF END OF HEADERS
104                                     ; (ASSUME END OF DATA)
105 040332 104401          TYPE      , $CRLF          ; (ASSUME NO DATA)
106 040334 000000          .WORD    0              ; HEADER ADDRESS GOES HERE
107 040336 104401 001217          TYPE      , $CRLF          ; (ASSUME NO DATA)
108 040342 005702          TST      R2              ; DATA WITH HEADER??
109 040344 001767          BEQ      14$            ; NO!!
110 040346 012204          MOV      (R2)+,R4        ; R4 POINTS TO DATA ADDRESS
111 040350 012305          MOV      (R3)+,R5        ; R5 POINTS TO FORMAT
112 040352 105725          TSTB     (R5)+           ; WHAT KIND OF DATA??
113 040354 100407          BMI      18$            ; BINARY
114 040356 001403          BEQ      17$            ; OCTAL
115 040360 013446          MOV      @ (R4)+,-(SP)   ; DECIMAL
116 040362 104405          TYPDS
117 040364 000405          BR       19$
118 040366 013446          MOV      @ (R4)+,-(SP)
119 040370 104402          TYPOC
120 040372 000402          BR       19$
121 040374 013446          MOV      @ (R4)+,-(SP)
122 040376 104406          TYPBN
123 040400 005714          TST      (R4)            ; MORE DATA??
124 040402 001403          BEQ      20$            ; NO!!
125 040404 104401 040467          TYPE      , ERTY04       ; YES-TYPE 2 SPACES
126 040410 000760          BR       16$            ; AND CONTINUE
127 040412 104401 001217          TYPE      , $CRLF          ; TYPE ONE BLANK LINE
128 040416 000742          BR       14$            ; BEFORE NEXT HEADER
129 040420 104415          RESREG
130 040422 000207          RTS      PC
131
132 040424 000000          TSTNMB: .WORD    0      ; TEST NUMBER
133 040426 000000          ERRNMB: .WORD    0      ; ERROR NUMBER
134 040430 000000          BOTADR: .WORD    0      ; BEGINNING OF TEXT ADDRESS
135 040432 000          BOTFLG: .BYTE    0      ; BOT FLAG
136 040433 000          CHRCNT: .BYTE    0      ; CHARACTER COUNT
137
138 040434 125 116 111 ERTY00: .ASCIZ @UNIT#@
139 040442 054 040 124 ERTY01: .ASCIZ @, TEST#@
140 040452 054 040 105 ERTY02: .ASCIZ @, ERR#@
141 040461 054 040 120 ERTY03: .ASCIZ @, PC=@
142 040467 040 040 000 ERTY04: .ASCIZ @ @
143 .EVEN
144

```

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 DEFINED BY MODULE
ERROR		

:TASK/VERIFY DESCRIPTOR

BIT 15 = 1	SELECT DEVICE AND VERIFY DEVICE IS AVAILABLE

BIT 14 = 1	CLEAR CONTROLLER AND SELECT DEVICE
BIT 13	(RESERVED FOR DRIVE CLEAR)
BIT 12 = 1	PACK ACKNOWLEDGE IF VOLUME NOT VALID

BIT 11 = 1	RECALIBRATE IF POSITIONING IN PROGRESS
BIT 10	
BIT 9	

BIT 8	
BIT 7	
BIT 6 = 1	VERIFY CONTROLLER CLEAR OPERATION

BIT 5	(RESERVED FOR DRIVE CLEAR)
BIT 4 = 1	VERIFY PACK ACKNOWLEDGE
BIT 3 = 1	VERIFY RECALIBRATION

BIT 2	
BIT 1	
BIT 0	

TSTPRP:

:STORE TASK DESCRIPTOR AND CLEAR USER'S ERROR CALL

41	040472	017637	000000	041412	MOV	@(SP),500\$:STORE DESCRIPTOR	
42	040500	062716	000006		ADD	#6,(SP)	:MOVE SP TO USERS ERROR CALL	
43	040504	105076	000000		CLRB	@(SP)	:CLEAR ERROR CALL	
44	040510	162716	000004		SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN	
46	040514	004737	044144		JSR	PC,GETSTS	:SETUP TO READ ALL REGISTERS	
47	040520	004737	044230		JSR	PC,GET	:GET RMER2	
48	040524	000411			BR	15\$:BR IF NO ERROR DETECTED	
49	040526	000401			BR	10\$:GET OVER ERROR NUMBER	
50	040530	000000			.WORD	0	:ERROR DEFINED BY GET SUBROUTINE	
51	040532	062716	000004	10\$:	ADD	#4,(SP)	:XFER ERROR TO USER AND	
52	040536	113776	040530	000000	MOVB	10\$-2,@(SP)	:GET ERROR NUMBER.	
53	040544	000137	041402		JMP	400\$		
55	040550	013737	001376	041414	15\$:	MOV	RMER2I,505\$:GET RMER2 AND SAVE FOR LATER

:*****

```

58      ;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
59 040556 005737 041412      TST      500$      ;SELECT DEVICE??
60 040562 100014      BPL      30$      ;NO!!
61
62 040564 004737 052132      JSR      PC,DEVSEL      ;GO SELECT DEVICE
63 040570 000411      BR      30$      ;NO ERROR - CONTINUE
64 040572 000401      BR      20$
65 040574 000000      .WORD   0      ;ERROR NUMBER FROM DEVSEL
66 040576 062716 000004      20$:  ADD      #4,(SP)      ;TRANSFER ERROR TO USER
67 040602 113776 040574 000000      MOVB    20$-2,a(SP)
68 040610 000137 041402      JMP     400$
69
70      ;*****
71      ;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
72 040614      30$:
73 040614 032737 040000 041412      BIT     #BIT14,500$      ;CLEAR CONTROLLER??
74 040622 001451      BEQ    120$      ;NO!!
75
76 040624 004737 053604      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
77 040630 000411      BR      60$      ;CONTINUE - NO ERROR
78 040632 000401      BR      50$
79 040634 000000      40$:  .WORD   0      ;ERROR NUMBER FROM CNTCLR
80 040636 062716 000004      50$:  ADD      #4,(SP)      ;TRANSFER ERROR TO USER
81 040642 113776 040634 000000      MOVB    40$,a(SP)
82 040650 000137 041402      JMP     400$
83
84      ;*****
85      ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
86 040654      60$:
87 040654 032737 000100 041412      BIT     #BIT6,500$      ;VERIFY??
88 040662 001431      BEQ    120$      ;NO!!
89
90 040664 004737 044230      JSR      PC,GET      ;GO GET STATUS
91 040670 000411      BR      90$      ;NO ERROR GETTING STATUS
92 040672 000401      BR      80$
93 040674 000000      70$:  .WORD   0      ;ERROR FROM GETTING STATUS
94 040676 062716 000004      80$:  ADD      #4,(SP)      ;TRANSFER ERROR TO USER
95 040702 113776 040674 000000      MOVB    70$,a(SP)
96 040710 000137 041402      JMP     400$
97
98 040714 004737 053722      90$:  JSR      PC,CLRSTS      ;GO VERIFY STATUS CLEAR
99 040720 000412      BR      120$      ;NO ERROR IN CLEAR
100 040722 000401      BR      110$
101 040724 000000      100$: .WORD   0      ;ERROR IN STATUS CLEAR
102 040726 005726      110$: TST     (SP)+      ;STRIP RETURN ADDRESS TO
103 040730 062716 000004      ADD     #4,(SP)      ;SUBROUTINE AND TRANSFER
104 040734 113776 040724 000000      MOVB    100$,a(SP)      ;ERROR TO USER
105 040742 000137 041402      JMP     400$
106
107      ;*****
108      ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
109      ;NOT VALID
110 040746      120$:
111 040746 032737 010000 041412      BIT     #BIT12,500$      ;PACK ACKNOWLEDGE??
112 040754 001503      BEQ    240$      ;NO!!
113
114 040756 004737 044230      JSR      PC,GET
    
```

```

115 040762 000411          BR      150$          ;NO ERROR GETTING RMDS
116 040764 000401          BR      140$
117 040766 000000          130$: .WORD      0
118 040770 062716 000004          140$: ADD      #4,(SP)      ;TRANSFER ERROR TO USER
119 040774 113776 040766 000000  MOVB    130$,a(SP)
120 041002 000137 041402          JMP     400$
121
122 041006 032737 000100 001346 150$: BIT     #VV,RMDSI      ;IS VOLUME VALID??
123 041014 001063          BNE    240$          ;YES!!
124
125 041016 005037 001510          CLR     MEDENB        ;CLEAR MEDIA ENABLE
126 041022 012737 000023 001410  MOV     #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
127 041030 112737 000000 001551  MOVB   #RMCS1,PUTINX    ;SETUP REGISTER INDEX TABLE
128 041036 112737 000200 001552  MOVB   #200,PUTINX+1
129 041044 004737 044500          JSR    PC,PUT         ;GO WRITE COMMAND
130 041050 000410          BR     180$          ;NO ERROR LOADING REGISTER
131 041052 000401          BR     170$
132 041054 000000          160$: .WORD      0          ;ERROR FROM PUT SUB
133 041056 062716 000004          170$: ADD     #4,(SP)      ;TRANSFER ERROR TO USER
134 041062 113776 041054 000000  MOVB   160$,a(SP)
135 041070 000544          BR     400$
136
137 041072 004737 045042          180$: JSR    PC,TIMOUT    ;WAIT FOR COMMAND TO COMPLETE
138
139
140          ;:*****
141 041076 032737 000020 041412  ;VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
142 041104 001427          BIT     #BIT4,500$    ;VERIFY PACK ACKNOWLEDGE??
143          BEQ     240$          ;NO!!
144 041106 004737 044230          JSR    PC,GET         ;GO GET STATUS
145 041112 000410          BR     210$          ;NO ERROR GETTING STATUS
146 041114 000401          BR     200$
147 041116 000000          190$: .WORD      0          ;ERROR FROM GET SUB
148 041120 062716 000004          200$: ADD     #4,(SP)      ;TRANSFER ERROR TO USER
149 041124 113776 041116 000000  MOVB   190$,a(SP)
150 041132 000523          BR     400$
151
152 041134 004737 054602          210$: JSR    PC,ACKSTS    ;GO CHECK ACKNOWLEDGE
153 041140 000411          BR     240$          ;NO ERROR
154 041142 000401          BR     230$
155 041144 000000          220$: .WORD      0          ;PACK ACKNOWLEDGE ERROR
156 041146 005726          230$: TST     (SP)+      ;STRIP RETURN TO SUB AND
157 041150 062716 000004          ADD     #4,(SP)      ;TRANSFER ERROR TO USER
158 041154 113776 041144 000000  MOVB   220$,a(SP)
159 041162 000507          BR     400$
160
161          ;:*****
162          ;RECALIBRATE DRIVE IF BIT 11 IS SET IN TASK AND 'SKI' IS SET
163          ;OR 'PIP' IS ACTIVE.
164 041164          240$:
165 041164 032737 004000 041412  BIT     #BIT11,500$    ;RECALIBRATE??
166 041172 001505          BEQ     410$          ;NO!!
167
168 041174 004737 044230          JSR    PC,GET         ;GO GET RMDS
169 041200 000410          BR     270$          ;NO ERROR GETTING RMDS
170 041202 000401          BR     260$
171 041204 000000          250$: .WORD      0          ;ERROR FROM GET SUB
    
```

```

172 041206 062716 000004      260$:  ADD    #4,(SP)      ;TRANSFER ERROR TO USER
173 041212 113776 041204 000000      MOVB   250$,a(SP)
174 041220 000470      BR     400$
175
176 041222 032737 040000 041414 270$:  BIT    #SKI,505$     ;WAS SKI SET ?
177 041230 001004      BNE    280$          ;YES, GO RECALIBRATE
178 041232 032737 020000 001346      BIT    #PIP,RMDSI    ;IS PIP ACTIVE??
179 041240 001462      BEQ    410$          ;NO!!
180
181 041242 012737 000007 001410 280$:  MOV    #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
182 041250 112737 000000 001551      MOVB   #RMCS1,PUTINX ;AND REGISTER INDEX
183 041256 112737 000200 001552      MOVB   #200,PUTINX+1
184 041264 004737 044500      JSR    PC,PUT        ;GO ISSUE RECALIBRATE
185 041270 000410      BR     300$          ;NO ERROR
186 041272 000401      BR     290$
187 041274 000000      .WORD  0             ;ERROR IN REGISTER TRANSFER
188 041276 062716 000004      290$:  ADD    #4,(SP)      ;TRANSFER ERROR TO USER
189 041302 113776 041274 000000      MOVB   290$-2,a(SP)
190 041310 000434      BR     400$
191
192 041312 004737 045042      300$:  JSR    PC,TIMOUT    ;WAIT FOR COMPLETION
193
194
195      ;:*****
196 041316 032737 000010 041412 ;VERIFY RECALIBRATE IF BIT 3 SET IN TASK
197 041324 001430      BIT    #BIT3,500$    ;VERIFY RECALIBRATE??
198      BEQ    410$          ;NO!!
199 041326 004737 044230      JSR    PC,GET        ;GO GET STATUS
200 041332 000410      BR     330$          ;NO ERROR GETTING STATUS
201 041334 000401      BR     320$
202 041336 000000      310$:  .WORD  0             ;ERROR FROM GET
203 041340 062716 000004      320$:  ADD    #4,(SP)      ;TRANSFER ERROR TO USER
204 041344 113776 041336 000000      MOVB   310$,a(SP)
205 041352 000413      BR     400$
206
207 041354 004737 055376      330$:  JSR    PC,RCLSTS    ;GO CHECK RECALIBRATE
208 041360 000412      BR     410$          ;NO ERROR DURING RECALIBRATE
209 041362 000401      BR     350$
210 041364 000000      340$:  .WORD  0             ;ERROR DURING RECALIBRATE
211 041366 005726      350$:  TST    (SP)+        ;STRIP RETURN TO SUB AND
212 041370 062716 000004      ADD    #4,(SP)      ;TRANSFER ERROR TO USER
213 041374 113776 041364 000000      MOVB   340$,a(SP)
214 041402 162716 000002      400$:  SUB    #2,(SP)      ;MOVE SP BACK BEFORE ERROR
215 041406 000240      410$:  NOP
216 041410 000207      RTS    PC            ;RETURN TO USER
217
218 041412 000000      500$:  .WORD  0             ;TASK/VERIFY DESCRIPTOR
219 041414 000000      505$:  .WORD  0             ;CONTAINS RMER2
220

```


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 041416
 34 041416 062716 000006
 35 041422 105076 000000
 36 041426 162716 000006
 37
 38
 39
 40 041432 005737 001510
 41 041436 001402
 42 041440 000137 042712
 43
 44
 45
 46
 47 041444
 48 041444 010046
 49 041450 016060 001410 106314 10\$:
 50 041456 062700 000002
 51 041462 022700 000046
 52 041466 103370
 53
 54
 55
 56 041470 012737 000003 043330

```

.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

: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
      CLRB  @ (SP)        ;ERROR CALL.
      SUB    #6,(SP)

:TEST 'MEDIA ENABLE' TO DETERMINE WHETHER OR NOT THE BAD SECTOR FILES
:HAVE BEEN RECOVERED.
      TST   MEDENB        ;HAS BAD SECTOR FILES BEEN RECOVERED ?
      BEQ   5$            ;BR IF NO
      JMP   300$         ;YES, BAD SECTOR FILE IS AVAILABLE

:RECOVER THE MANUFACTURES / USERS BAD SECTOR FILE FROM CYLINDER = 822.,
:TRACK = LAST TRACK (RM02/3 = 4 AND RM05 = 18.). ALSO, SAVE THE USER'S
:PUT BUFFER
5$:
      MOV   R0,-(SP)      ;;PUSH R0 ON STACK
      CLR  R0             ;START WITH RMCS1
10$:
      MOV  PUTBUF(R0),BUFFER(R0)
      ADD  #2,R0          ;ADVANCE TO NEXT BUFFER POSITION
      CMP  #46,R0         ;END OF BUFFER
      BHIS 10$           ;NO !!

:SET RETRY COUNT AND LOAD PUT BUFFER AND REGISTER INDEX TABLE
:SETUP PARAMETERS TO READ SKIP SECTOR FILE FIRST (3 SECTORS)
      MOV  #3,500$       ;RETRY COUNT
    
```

```

57 041476 012737 001466 001444      MOV      #822.,RMDCO      ;DESIRED CYLINDER = 822.
58 041504 013737 001332 001416      MOV      LSTRK,RMDAO    ;STARTING LAST TRACK, SECTOR = 0
59 041512 012737 177376 001412      MOV      #-258.,RMWCO   ;2 + 256. WORDS (2'S COMP)
60 041520 012737 010000 001442      MOV      #FMT16,RMOFO   ;16 BIT FORMAT
61 041526 012737 110324 001414      MOV      #MFGFIL,RMBAO  ;POINT TO MANUFACTURES FILE BUFFER
62
63 041534 012700 001551      MOV      #PUTINX,R0     ;R0 POINTS TO REGISTER INDEX TABLE
64 041540 112720 000006      MOV      #RMDA,(R0)+
65 041544 112720 000034      MOV      #RMDC,(R0)+
66 041550 112720 000002      MOV      #RMWC,(R0)+
67 041554 112720 000032      MOV      #RMOF,(R0)+
68 041560 112720 000004      MOV      #RMB A,(R0)+
69 041564 112720 000000      MOV      #RMCS1,(R0)+
70 041570 112720 000200      MOV      #200,(R0)+
71 041574 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
72
73      ;SET GET INDEX TABLE FOR READING STATUS
74 041576      20$:      JSR      PC,GETSTS     ;SETUP GET INDEX REGISTER FOR STATUS
75 041576 004737 044144      JSR      PC,GET       ;GET REGISTERS
76 041602 004737 044230      BR      30$           ;BR IF NO ERROR
77 041606 000411      BR      25$           ;JUMP OVER ERROR NUMBER
78 041610 000401      .WORD   0             ;ERROR DEFINED BY GET SUB
79 041612 000000      25$:      ADD      #6,(SP)     ;XFER ERROR TO USER AND
80 041614 062716 000006      MOV      25$-2,@(SP)  ;GET ERROR NUMBER.
81 041620 113776 041612 000000      JMP      215$
82 041626 000137 042412
83
84 041632 013737 001376 043342 30$:      MOV      RMER2I,550$  ;GET RMER2 AND SAVE FOR LATER
85
86      ;CLEAR THE DEVICE USING DRIVE CLEAR COMMAND
87 041640 012737 000011 001410      MOV      #DRVCLR!GO,RMCS10 ;LOAD COMMAND IN PUT BUFFER
88 041646 004737 044500      JSR      PC,PUT       ;OUTPUT COMMAND
89 041652 000411      BR      45$           ;RETURN HERE IF NO ERROR
90 041654 000401      BR      40$           ;GET AROUND ERROR #
91 041656 000000      35$:      .WORD   0             ;ERROR # GOES HERE
92 041660 062716 000006      40$:      ADD      #6,(SP)     ;MOVE SP TO USERS ERROR CALL
93 041664 113776 041656 000000      MOV      35$,@(SP)   ;MOVE ERROR NUMBER TO USER
94 041672 000137 042412      JMP      215$
95
96 041676 004737 045042      45$:      JSR      PC,TIMOUT    ;WAIT FOR COMPLETION
97 041702 004737 044230      JSR      PC,GET       ;GO GET STATUS
98 041706 000411      BR      60$           ;RETURN HERE IF NO ERROR
99 041710 000401      BR      55$           ;GET AROUND ERROR #
100 041712 000000      50$:      .WORD   0             ;ERROR # GOES HERE
101 041714 062716 000006      55$:      ADD      #6,(SP)     ;MOVE SP TO USERS ERROR CALL
102 041720 113776 041712 000000      MOV      50$,@(SP)  ;MOVE ERROR # TO USERS ERROR CALL
103 041726 000137 042412      JMP      215$
104
105 041732 004737 057140      60$:      JSR      PC,DRVSTS    ;GO VERIFY DRIVE CLEAR COMMAND
106 041736 000412      BR      75$           ;RETURN HERE IF NO ERROR
107 041740 000401      BR      70$           ;GET AROUND ERROR
108 041742 000000      65$:      .WORD   0             ;ERROR # GOES HERE
109 041744 005726      70$:      TST      (SP)+       ;STRIP RETURN TO SUBROUTINE
110 041746 062716 000006      ADD      #6,(SP)     ;MOVE SP TO USERS ERROR CALL
111 041752 113776 041742 000000      MOV      65$,@(SP)  ;MOVE ERROR # TO USER CALL
112 041760 000137 042412      JMP      215$
113

```

```

114                                     :ISSUE A PACK ACKNOWLEDGE IF VOLUME VALID IS RESET
115 041764                               75$:
116 041764 032737 000100 001346        BIT    #VV,RMDSI    :IS VV RESET ??
117 041772 001052                                BNE    120$      :NO !!
118
119 041774 012737 000023 001410        MOV    #PACACK!GO,RMCS10 :LOAD COMMAND
120 042002 004737 044500                JSR    PC,PUT      :GO PUT COMMAND TO DRIVE
121 042006 000411                                BR     90$        :RETURN HERE IF NO OUTPUT ERROR
122 042010 000401                                BR     85$        :GET AROUND ERROR #
123 042012 000000                80$: .WORD 0      :ERROR # GOES HERE
124 042014 062716 000006                85$: ADD    #6,(SP)   :MOVE SP TO USERS ERROR CALL
125 042020 113776 042012 000000        MOVB   80$,a(SP)   :MOVE ERROR # TO ERROR CALL
126 042026 000137 042412                JMP    215$
127
128 042032 004737 045042                90$: JSR    PC,TIMOUT :WAIT FOR COMPLETION
129 042036 004737 044230                JSR    PC,GET      :GO GET STATUS FOR PACK ACK
130 042042 000411                                BR     105$       :RETURN HERE IF NO ERROR
131 042044 000401                                BR     100$       :GET AROUND ERROR #
132 042046 000000                95$: .WORD 0      :ERROR # GOES HERE
133 042050 062716 000006                100$: ADD   #6,(SP)  :MOVE SP TO USERS ERROR CALL
134 042054 113776 042046 000000        MOVB   95$,a(SP)  :MOVE ERROR # TO CALL
135 042062 000137 042412                JMP    215$
136
137 042066 004737 054602                105$: JSR   PC,ACKSTS :GO VERIFY ACKNOWLEDGE STATUS
138 042072 000412                                BR     120$       :RETURN HERE IF NO ERROR
139 042074 000401                                BR     115$       :GET AROUND ERROR #
140 042076 000000                110$: .WORD 0      :ERROR # GOES HERE
141 042100 005726                115$: TST    (SP)+    :STRIP RETURN TO SUBROUTINE
142 042102 062716 000006                ADD    #6,(SP)    :MOVE SP TO USERS ERROR CALL
143 042106 113776 042076 000000        MOVB   110$,a(SP) :MOVE ERROR # TO USERS ERROR CALL
144 042114 000137 042412                JMP    215$
145
146                                     :RECALIBRATE THE DRIVE IF 'SKI' OR 'PIP IS SET
147 042120                               120$:
148 042120 032737 040000 043342        BIT    #SKI,550$   :WAS SKI SET ?
149 042126 001004                                BNE    125$      :YES, GO RECALIBRATE
150 042130 032737 020000 001346        BIT    #PIP,RMDSI  :IS PIP SET ??
151 042136 001452                                BEQ    165$      :NO !!
152
153 042140 012737 000007 001410        125$: MOV    #RECAL!GO,RMCS10 :LOAD RECALIBRATE COMMAND
154 042146 004737 044500                JSR    PC,PUT      :PUT THE RECAL COMMAND
155 042152 000411                                BR     135$       :RETURN HERE IF NO ERROR
156 042154 000401                                BR     130$       :GET AROUND ERROR #
157 042156 000000                .WORD 0          :ERROR # GOES HERE
158 042160 062716 000006                130$: ADD   #6,(SP)  :MOVE SP TO USERS ERROR CALL
159 042164 113776 042156 000000        MOVB   130$-2,a(SP) :MOVE ERROR # TO USERS CALL
160 042172 000137 042412                JMP    215$
161
162 042176 004737 045042                135$: JSR   PC,TIMOUT :WAIT FOR RECALIBRATE TO COMPLETE
163 042202 004737 044230                JSR    PC,GET      :GO GET RECAL STATUS
164 042206 000411                                BR     150$       :RETURN HERE IF NO ERROR
165 042210 000401                                BR     145$       :GET AROUND ERROR #
166 042212 000000                140$: .WORD 0      :ERROR # GOES HERE
167 042214 062716 000006                145$: ADD   #6,(SP)  :MOVE SP TO USERS ERROR CALL
168 042220 113776 042212 000000        MOVB   140$,a(SP) :MOVE ERROR TO USERS CALL
169 042226 000137 042412                JMP    215$
170
    
```

```

171 042232 004737 055376      150$: JSR    PC,RCLSTS      ;GO VERIFY RECALIBRATE STATUS
172 042236 000412              BR      165$              ;RETURN HERE IF NO ERROR
173 042240 000401              BR      160$              ;GET AROUND ERROR #
174 042242 000000      155$: .WORD  0              ;ERROR # GOES HERE
175 042244 005726      160$: TST    (SP)+          ;STRIP RETURN TO SUBROUTINE
176 042246 062716 000006      ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
177 042252 113776 042242 000000      MOVB   155$,a(SP)      ;MOVE ERROR # TO USERS CALL
178 042260 000137 042412      JMP    215$
179
180      ;READ THE SECTOR IDENTIFIED BY RMDAO, INCLUDING HEADER AND DATA
181 042264              165$:
182 042264 012737 000073 001410      MOV    #RH!GO,RMCS10    ;LOAD READ HEADER AND DATA COMMAND
183 042272 004737 044500      JSR    PC,PUT           ;PUT COMMAND
184 042276 000411              BR      180$           ;RETURN HERE IF NO ERROR
185 042300 000401              BR      175$           ;GET AROUND ERROR #
186 042302 000000      170$: .WORD  0              ;ERROR # GOES HERE
187 042304 062716 000006      175$: ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
188 042310 113776 042302 000000      MOVB   170$,a(SP)      ;MOVE ERRGR # TO USERS ERROR CALL
189 042316 000137 042412      JMP    215$
190
191 042322 004737 045042      180$: JSR    PC,TIMOUT     ;WAIT FOR READ OPERATION TO COMPLETE
192 042326 004737 044230      JSR    PC,GET          ;GO GET STATUS FOR READ OPERATION
193 042332 000411              BR      195$           ;RETURN HERE IF NO ERROR
194 042334 000401              BR      190$           ;GET AROUND ERROR #
195 042336 000000      185$: .WORD  0              ;ERROR # GOES HERE
196 042340 062716 000006      190$: ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
197 042344 113776 042336 000000      MOVB   185$,a(SP)      ;MOVE ERROR # TO CALL
198 042352 000137 042412      JMP    215$
199
200 042356 004737 057742      195$: JSR    PC,DTASTS     ;GO VERIFY RESULTS OF READ OPERATION
201 042362 000412              BR      210$           ;RETURN HERE IF NO ERROR
202 042364 000401              BR      205$           ;GET AROUND ERROR #
203 042366 000000      200$: .WORD  0              ;ERROR # GOES HERE
204 042370 005726      205$: TST    (SP)+          ;STRIP RETURN ADDRESS TO SUBROUTINE
205 042372 062716 000006      ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
206 042376 113776 042366 000000      MOVB   200$,a(SP)      ;MOVE ERROR # TO USERS CALL
207 042404 000137 042412      JMP    215$
208
209 042410 000450      210$: BR      240$           ;NO ERRORS DETECTED
210
211      ;*****
212      ;AN ERROR HAS BEEN DETECTED IN TRYING TO READ THE BAD SECTOR FILE.
213      ;THE SECTOR WILL BE RETRIED IF POSSIBLE.
214 042412      215$:
215 042412 005337 043330      DEC    500$            ;YES, DECREMENT RETRY COUNT AND
216 042416 100030      BPL    225$            ;RETRY IF COUNT NOT NEGATIVE.
217
218      ;THE RETRY COUNT HAS EXPIRED - SEE IF THE ERROR IS MEDIA RELATED
219 042420      220$:
220 042420 013746 001376      MOV    RMER2I,-(SP)     ;GET ER2
221 042424 042726 100000      BIC    #BSE,(SP)+      ;ANY NON-MEDIA ERRORS ?
222 042430 001027      BNE    230$            ;YES, EXIT AND REPORT ERROR ON RETURN
223
224 042432 013746 001350      MOV    RMER1I,-(SP)     ;GET ER1
225 042436 042726 100720      BIC    #DCK!HCRC!HCE!FER!ECH,(SP)+ ;ARE THERE ANY NON-MEDIA ERRORS ?
226
227 042442 001022      BNE    230$            ;YES, EXIT AND REPORT ERROR ON RETURN

```

```

228
229
230      ;THE ERRORS DETECTED WHILE TRYING TO RECOVER THE BAD SECTOR FILE ARE
231      ;DUE TO THE MEDIA.  SEE IF THE BAD SECTOR FILE CAN BE RECOVERED FROM
232      ;ANOTHER AREA ON THE LAST TRACK
233 042444 062737 000002 001416      ADD      #2,RMDAO      ;ADVANCE SECTOR ADDRESS BY 2
234 042452 122737 000012 001416      CMPB     #10.,RMDAO    ;QUIT IF ALL MFG SECTORS HAVE BEEN
235 042460 001413      BEQ      230$         ;TRIED.
236 042462 122737 000040 001416      CMPB     #32.,RMDAO   ;QUIT IF ALL USER SECTORS HAVE BEEN
237 042470 001407      BEQ      230$         ;TRIED.
238
239 042472 012737 000003 043330      MOV      #3,500$     ;REINSTATE RETRY COUNT FOR THIS SECTOR
240 042500 162716 000006      225$:  SUB      #6,(SP)   ;MOVE SP BACK TO NO ERROR
241 042504 000137 041576      JMP      20$         ;RETRY THE READ OPERATION
242
243      ;THE BAD SECTOR FILE CANNOT BE READ
244 042510      230$:
245 042510 000240      NOP
246 042512 032777 020000 136434      BIT      #SW13,@SWR  ;INHIBIT MESSAGE ?
247 042520 001002      BNE     235$         ;YES
248 042522 162716 000004      SUB      #4,(SP)     ;MOVE SP TO ERROR RETURN
249 042526 000137 043324      235$:  JMP      410$        ;GO TO MODULE EXIT
250
251      ;THE SECTOR WAS RECOVERED WITHOUT ERROR - READ THE USER FILE IF
252      ;THIS IS THE MGF FILE OR ELSE DONE.
253 042532      240$:
254 042532 022737 111332 001414      CMP      #USRFIL,RMBAO ;WAS THE USER FILE READ ??
255 042540 001446      BEQ     260$         ;YES - READ IS COMPLETE
256 042542 112737 000012 001416      MOVB     #10.,RMDAO   ;READ THE USER FILE LAST TRACK, SECTOR = 10.
257 042550 012737 111332 001414      MOV      #USRFIL,RMBAO ;POINT TO USERS FILE BUFFER
258
259 042556 012737 000003 043330      MOV      #3,500$     ;RELOAD THE RETRY COUNT FOR THIS SECTOR
260 042564 000137 041576      JMP      20$         ;GO READ THE USER FILE
261
262      ;DUMMY THE BAD SECTOR FILES
263 042570      250$:
264 042570 010046      MOV      R0,-(SP)    ;;PUSH R0 ON STACK
265 042572 010146      MOV      R1,-(SP)    ;;PUSH R1 ON STACK
266 042574 012701 000374      MOV      #252.,R1    ;R1 = NUMBER OF ENTRIES IN FILES
267 042600 012700 000014      MOV      #14,R0     ;R0 = ADDRESS INDEX TO FILE STORAGE
268 042604 012760 177777 110324      255$:  MOV      #-1,MFGFIL(R0) ;ENTER ALL ONES IN MFG FILE
269 042612 012760 177777 111332      MOV      #-1,USRFIL(R0) ;ENTER ALL ONES IN USER FILE
270 042620 005720      TST     (R0)+        ;ADVANCE ADDRESS
271 042622 005301      DEC     R1           ;DECREMENT COUNT
272 042624 001367      BNE     255$        ;CONTINUE IF NOT DONE
273
274 042626 012701 000006      MOV      #6.,R1     ;CLEAR HEADER, CLEAR ID & SERIAL NUMBERS
275 042632 005000      CLR     R0
276 042634 005060 110324      257$:  CLR     MFGFIL(R0)
277 042640 005060 111332      CLR     USRFIL(R0)
278 042644 005720      TST     (R0)+        ;ADVANCE ADDRESS
279 042646 005301      DEC     R1
280 042650 001371      BNE     257$
281 042652 012601      MOV     (SP)+,R1    ;;POP STACK INTO R1
282 042654 012600      MOV     (SP)+,R0    ;;POP STACK INTO R0
;SET MEDIA ENABLE AND RESTORE THE USERS PUT BUFFER

```

```

282 042656          260$:      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
      042656 010046          CLR     R0           ;;R0 IS REGISTER INDEX
283 042660 005000          MOV     #-1,MEDENB
284 042662 012737 177777 001510 265$:      MOV     BUFFER(R0),PUTBUF(R0)
285 042670 016060 106314 001410          ADD     #2,R0        ;ADVANCE R0
286 042676 062700 000002          CMP     #46,R0      ;DONE ??
287 042702 022700 000046          BHIS   265$
288 042706 103370          MOV     (SP)+,R0    ;;POP STACK INTO R0
289 042710 012600
290
291
292
293
294
295
296
297
298 042712          ;*****
      042712 010046          ;VERIFY THAT THE DESIRED SECTOR IS NOT IN THE MFG BAD SECTOR FILE
      042714 010146          ;AND NOT IN THE USERS BAD SECTOR FILE. ASSIGN A NEW SECTOR IF THE
      042716 010246          ;SECTOR IS IN ANY OF THE FILES.
      042716 010246          ;*****
299 042720 013737 001444 001512          ;LOAD INITIAL VARIABLES AND COMPUTE THE NUMBER OF SECTORS
300 042726 013737 001416 001514          300$:      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
      042714 010146          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
      042716 010246          MOV     R2,-(SP)      ;;PUSH R2 ON STACK
301 042734 005002          MOV     RMDCO,ASNDC   ;LOAD REQUESTED CYLINDER, TRACK,
302 042736 013700 001412          MOV     RMDAO,ASNDA   ;AND SECTOR ADDRESS IN ASSIGNED STORAGE
303 042742 005400          CLR     R2           ;R2 = NUMBER OF SECTORS
304 042744 012701 000400          MOV     RMWCO,R0      ;R0 = WORD COUNT
305 042750 032737 000002 001410          NEG     R0           ;MAKE NUMBER POSITIVE
306 042756 001402          MOV     #256.,R1     ;R1 = NUMBER OF WORDS PER SECTOR
307 042760 012701 000402          BIT     #BIT1,RMCS10 ;IS THIS A HEADER AND DATA COMMAND ??
308 042764 020100          BEQ    305$         ;NO !!
309 042766 101404          MOV     #258.,R1     ;CHANGE WORDS PER SECTOR
310 042770 005700          CMP     R1,R0        ;IS THERE A FULL SECTOR ??
311 042772 001405          BLOS   310$         ;YES !!
312 042774 005202          TST    R0           ;IS R0 ZERO ??
313 042776 000403          BEQ    315$         ;YES !!
314 043000 160100          INC    R2           ;INCREMENT FOR PARTIAL SECTOR
315 043002 005202          BR     315$
316 043004 000767          SUB    R1,R0        ;SUBTRACT ONE SECTOR FROM WORD COUNT
317 043006 010237 043330          INC    R2           ;INCREMENT SECTOR COUNT
318
319          315$:      MOV     R2,500$      ;SAVE SECTOR COUNT
320
321          ;LOAD PARAMETERS AND SEARCH THE MFG/USER SECTOR FILE FOR THE
322          ;ASSIGNED SECTOR. ALSO, SEARCH THE ADJACENT SECTORS IF THE
323          ;SECTOR COUNT IS MORE THAN ONE.
323 043012 012737 110340 043340          MOV     #MFGFIL+14,540$ ;THE STARTING ADDRESS OF MFG FILE
324
325 043020 004737 043042          JSR    PC,320$      ;TO BASE ADDRESS STORAGE.
326 043024 012737 111346 043340          MOV     #USRFIL+14,540$ ;GO SEARCH FILE
327
328 043032 004737 043042          JSR    PC,320$      ;LOAD STARTING ADDRESS OF USR FILE
329 043036 000137 043302          JSR    PC,320$      ;TO BASE ADDRESS STORAGE.
330
331 043042 013737 001512 043334          JMP    400$         ;GO SEARCH FILE
332 043050 013737 001514 043336          ;DONE WITH ALL FILE SEARCHES !!
333 043056 013737 043330 043332          320$:      MOV     ASNDC,520$   ;LOAD COMPARING CYLINDER ADDRESS
334
      MOV     ASNDA,530$   ;LOAD COMPARING TRACK, SECTOR ADDRESS
      MOV     500$,510$   ;LOAD NUMBER OF SECTORS TO CONFIRM
    
```

```

335      ;SETUP FOR A BINARY SEARCH OF THE CURRENT FILE FOR THE COMPARING
336      ;CYLINDER, TRACK AND SECTOR ADDRESS
337 043064      325$:
338 043064 013700 043340      MOV      540$,R0      ;LOAD THE BASE ADDRESS IN R0
339 043070 022710 177777      330$:  CMP      #-1,(R0)      ;IS THIS FILE TERMINATOR ?
340 043074 001446      BEQ      370$      ;BR IF YES
341 043076 021037 043334      CMP      (R0),520$      ;DOES TABLE ENTRY = COMPARING CYLINDER ?
342 043102 001010      BNE      350$      ;BR IF NO
343
344      ;FILE ENTRY EQUALS COMPARING CYLINDER. SEE IF THE NEXT ENTRY EQUALS
345      ;THE COMPARING TRACK, AND SECTOR.
346 043104      345$:
347 043104 126037 000003 043337      CMPB     3(R0),530$+1      ;DOES TABLE ENTRY = COMPARING TRACK ?
348 043112 001004      BNE      350$      ;BR IF NO
349
350 043114 126037 000002 043336      CMPB     2(R0),530$      ;DOES TABLE ENTRY = COMPARING SECTOR ?
351 043122 001402      BEQ      360$      ;BR IF YES
352
353 043124 022020      350$:  CMP      (R0)+,(R0)+      ;NO, ADJUST CYLINDER POINTER IN BAD FILE
354 043126 000760      BR       330$      ;AND CONTINUE SEARCH.
355
356      ;THE COMPARING CYLINDER, TRACK AND SECTOR IS IN THE BAD SECTOR FILE.
357      ;ADVANCE THE ASSIGNED SECTOR AND START THE SEARCH ALL OVER.
358 043130      360$:
359 043130 105237 001514      INCB     ASNDA      ;INCREMENT SECTOR
360 043134 122737 000037 001514      CMPB     #31.,ASNDA      ;SECTOR OK ??
361 043142 103337      BHIS     320$      ;YES !!
362 043144 105037 001514      CLR     ASNDA      ;CLEAR SECTOR AND ADVANCE TRACK
363 043150 105237 001515      INCB     ASNDA+1
364 043154 123737 001333 001515      CMPB     LSTRK+1,ASNDA+1 ;TRACK OK ?
365 043162 103327      BHIS     320$      ;YES !!
366 043164 005037 001514      CLR     ASNDA      ;CLEAR TRACK AND SECTOR
367 043170 005237 001512      INC     ASNDC      ;INCREMENT CYLINDER
368 043174 022737 001466 001512      CMP     #822.,ASNDC      ;CYLINDER OK ??
369 043202 103317      BHIS     320$      ;YES !!
370 043204 005037 001512      CLR     ASNDC      ;START AT CYLINDER 0
371 043210 000714      BR      320$      ;SEARCH NEXT SECTOR
372
373      ;THE COMPARING SECTOR IS NOT IN THE BAD SECTOR FILES. DECREMENT THE
374      ;NUMBER OF SECTORS TO COMPARE AND SEARCH THE NEXT SECTOR IF THE NUMBER
375      ;IS NOT ZERO.
376 043212      370$:
377 043212 005337 043332      DEC     510$      ;DECREMENT NUMBER OF SECTORS TO COMPARE
378 043216 001442      BEQ     410$      ;DONE IF ZERO
379
380 043220 105237 043336      INCB     530$      ;INCREMENT THE COMPARING SECTOR
381 043224 122737 000037 043336      CMPB     #31.,530$      ;SECTOR OK ??
382 043232 103022      BHIS     375$      ;YES !!
383 043234 105037 043336      CLR     530$      ;CLEAR SECTOR
384 043240 105237 043337      INCB     530$+1      ;INCREMENT TRACK
385 043244 123737 001333 043337      CMPB     LSTRK+1,530$+1 ;TRACK OK ??
386 043252 103012      BHIS     375$      ;YES !!
387 043254 005037 043336      CLR     530$      ;CLEAR SECTOR TRACK
388 043260 005237 043334      INC     520$      ;INCREMENT CYLINDER
389 043264 022737 001466 043334      CMP     #822.,520$      ;CYLINDER OK ??
390 043272 103002      BHIS     375$      ;YES !!
391 043274 005037 043334      CLR     520$      ;START AT CYLINDER 0
    
```

```

392
393 043300 000671          375$: BR      325$          ;SEARCH NEXT SECTOR
394
395          ;ASSIGN THE SECTOR AND RETURN TO USER
396 043302          400$:
397 043302 013737 001512 001444      MOV      ASNDC,RMDCO      ;LOAD CYLINDER
398 043310 013737 001514 001416      MOV      ASNDA,RMDAO      ;LOAD TRACK AND SECTOR
399 043316 012602          MOV      (SP)+,R2        ;;POP STACK INTO R2
      043320 012601          MOV      (SP)+,R1        ;;POP STACK INTO R1
      043322 012600          MOV      (SP)+,R0        ;;POP STACK INTO R0
400
401 043324 000240          410$: NOP
402 043326 000207          RTS      PC
403
404          ;THE FOLLOWING ARE STORAGE LOCATIONS FOR THE MODULE
405
406 043330 000000          500$: .WORD 0          ;RETRY COUNT/ NUMBER OF SECTORS REQUIRED
407 043332 000000          510$: .WORD 0          ;NUMBER OF SECTORS TO COMPARE
408 043334 000000          520$: .WORD 0          ;COMPARING CYLINDER
409 043336 000000          530$: .WORD 0          ;COMPARING TRACK AND SECTOR
410 043340 000000          540$: .WORD 0          ;BASE ADDRESS OF BAD SECTOR FILE BEING SEARCHED
411 043342 000000          550$: .WORD 0          ;CONTAINS RMER2
412
    
```



```

1      .SBTTL  BUFFER GENERATOR SUBROUTINE
2
3      ;THIS SUBROUTINE GENERATES A DATA BUFFER FOR WRITE COMMANDS.  THE
4      ;BUFFER STARTS AT RMBA AND IS RMWC WORDS LONG.  THE BUFFER
5      ;CONTAINS A REPETITIVE DATA PATTERN CONSISTING OF $TMP1 WORDS
6      ;FROM THE DATA PATTERN TABLE, BEGINNING AT ADDRESS $TMP0.
7      ;HEADER INFORMATION FOR THE BUFFER IS EXTRACTED FROM RMDC,
8      ;RMDA AND RMOF.
9
10     ;R0 = ADDRESS OF DATA BUFFER
11     ;R1 = LENGTH OF DATA BUFFER
12     ;R2 = ADDRESS OF DATA PATTERN
13     ;R3 = LENGTH OF DATA PATTERN
14     ;R4 = SECTOR COUNT
15
16     ;CALL :
17     ;(1)  JSR      PC,GENBUF
18     ;(2)  ??
19
20     GENBUF :
21     043344      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
22     043346      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
23     043350      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
24     043352      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
25     043354      010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
26     043356      013700      001414      MOV      RMBAO,R0      ;LOAD DATA BUFFER ADDRESS
27     043362      013701      001412      MOV      RMWCO,R1      ;LOAD WORD COUNT
28     043366      013737      001444      043576      MOV      RMDCO,60$     ;LOAD STARTING CYLINDER ADDRESS
29     043374      013737      001416      043600      MOV      RMDAO,65$     ;LOAD STARTING TRACK,SECTOR ADDRESS
30
31     043402      032737      000002      001410      10$:      BIT      #BIT1,RMCS10    ;WRITE HEADER & DATA??
32     043410      001445      BEQ      25$          ;NO!!
33     043412      013710      043576      MOV      60$,(R0)     ;WRITE HEADER WORD #1
34     043416      052710      140000      BIS      #MSE!USE,(R0) ;SET BAD SECTOR FLAGS FOR GOOD SECTOR
35     043422      012702      000035      MOV      #29.,R2      ;R2 = MAXIMUM SECTOR ADDRESS (29.)
36
37     043426      032737      010000      001442      BIT      #FMT16,RMOFO  ;18 BIT FORMAT??
38     043434      001404      BEQ      15$          ;YES !!
39     043436      052710      010000      BIS      #FMT16,(R0)  ;SET 16 FORMAT BIT IN HEADER
40     043442      012702      000037      MOV      #31.,R2      ;CHANGE MAXIMUM SECTOR ADDRESS (31.)
41
42     043446      005201      15$:      INC      R1            ;INCREMENT WORD COUNT
43     043450      001443      BEQ      50$          ;EXIT IF DONE
44
45     043452      005720      TST      (R0)+         ;MOVE R0 TO HEADER WORD #2
46     043454      013720      043600      MOV      65$,(R0)+    ;WRITE HEADER WORD #2
47     043460      005201      INC      R1            ;INCREMENT WORD COUNT AND
48     043462      001436      BEQ      50$          ;EXIT IF DONE
49     043464      012703      043600      MOV      #65$,R3      ;ADVANCE SECTOR ADDRESS
50     043470      105213      INCB     (R3)          ;
51     043472      120213      CMPB    R2,(R3)        ;SECTOR OVERFLOW ??
52     043474      103013      BHIS    25$          ;NO !!
53     043476      105013      CLRB    (R3)          ;YES - CLEAR SECTOR ADDRESS
54     043500      105263      000001      INCB    1(R3)         ;ADVANCE TRACK ADDRESS
55     043504      123763      001333      000001      CMPB    LSTRK+1,1(R3) ;TRACK OVERFLOW ??
56     043512      103004      BHIS    25$          ;NO !!
57     043514      105063      000001      CLRB    1(R3)         ;YES - CLEAR TRACK ADDRESS

```

53	043520	105237	043576		INCB	60\$:ADVANCE CYLINDER ADDRESS
54	043524	012704	000400	25\$:	MOV	#256,R4		:LOAD SECTOR DATA COUNT
55	043530	013702	001174	30\$:	MOV	\$TMP0,R2		:LOAD PATTERN ADDRESS
56	043534	013703	001176		MOV	\$TMP1,R3		:LOAD PATTERN COUNT
57	043540	012220		40\$:	MOV	(R2)+,(R0)+		:WRITE DATA PATTERN
58	043542	005201			INC	R1		:INCREMENT WORD COUNT AND
59	043544	001405			BEQ	50\$:EXIT IF DONE
60	043546	005304			DEC	R4		:DECREMENT SECTOR COUNT
61	043550	001714			BEQ	10\$:START NEXT SECTOR IF 0
62	043552	005303			DEC	R3		:DECREMENT PATTERN COUNT
63	043554	001765			BEQ	30\$:RESTART PATTERN IF 0
64	043556	000770			BR	40\$:CONTINUE DATA PATTERN
65	043560			50\$:				
	043560	012604			MOV	(SP)+,R4		::POP STACK INTO R4
	043562	012603			MOV	(SP)+,R3		::POP STACK INTO R3
	043564	012602			MOV	(SP)+,R2		::POP STACK INTO R2
	043566	012601			MOV	(SP)+,R1		::POP STACK INTO R1
	043570	012600			MOV	(SP)+,R0		::POP STACK INTO R0
66	043572	000240			NOP			
67	043574	000207			RTS	PC		
68								
69	043576	000000		60\$:	.WORD			:CYLINDER ADDRESS STORAGE
70	043600	000000		65\$:	.WORD			:TRACK, SECTOR ADDRESS STORAGE
71								

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

```

.SBTTL COMPARE BUFFER SUBROUTINE

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

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

CMPBUF:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
CLR 150$ ;;CLEAR CORRECTION FLAG

;DETERMINE IF DATA SHOULD BE CORRECTED
BIT ECI,RMOFI ;;WAS ECC CORRECTION ALLOWED ??
BNE 80$ ;;NO !!
BIT #DCK,RMER1I ;;WAS THERE A DATA CHECK ??
BEQ 80$ ;;NO !!
BIT #ECH,RMER1I ;;IS ERROR CORRECTION HARD SET ?
BNE 80$ ;;YES !!
BIT #FMT16,RMOFI ;;IS THIS 16 BIT FORMAT ??
BEQ 80$ ;;NO !!

;CORRECT DATA USING ECC INFORMATION
MOV RMBAO,R0 ;;R0 = STARTING BUFFER ADDRESS
MOV RMECI,R1 ;;R1 = ECC POSITION
BIS #BIT15,150$ ;;SET CORRECTION FLAG

;MOVE R0 TO WORD BOUNDARY OF ERROR BURST
10$: CMP #16.,R1 ;;IS BIT POSITION > 1 WORD
BHS 20$ ;;NO !!
SUB #16.,R1 ;;SUBTRACT 1 WORDS WORTH
TST (R0)+ ;;ADVANCE BUFFER ADDRESS 1 WORD
BR 10$

20$: MOV #1,R2 ;;R2 = BIT POINTER
MOV R2,R3 ;;R3 = BIT NUMBER

;MOVE R2 TO STARTING BIT OF ERROR BURST
30$: CMP R3,R1 ;;IS R3 SAME AS R1 ??
BEQ 35$ ;;YES !!
ASL R2 ;;SHIFT BIT POINTER
INC R3 ;;INCREMENT BIT NUMBER
BR 30$

35$: MOV #11.,R3 ;;R3 = LENGTH OF ERROR BURST

;CORRECT THE ERROR BURST
40$: BIT R2,RMECI2I ;;IS THIS BIT SET IN ECC PATTERN ??
  
```

```

043602 010046
043604 010146
043606 010246
043610 010346
043612 005037 044142

043616 033737 004000 001366
043624 001063
043626 032737 100000 001350
043634 001457
043636 032737 000100 001350
043644 001053
043646 032737 010000 001366
043654 001447

043656 013700 001414
043662 013701 001400
043666 052737 100000 044142

043674 022701 000020
043700 103004
043702 162701 000020
043706 005720
043710 000771
043712 012702 000001
043716 010203

043720 020301
043722 001403
043724 006302
043726 005203
043730 000773
043732 012703 000013

043736 030237 001402
  
```

```

54 043742 001405          BEQ      60$          ;NO - DO NOT CORRECT THIS BIT
55 043744 030210          BIT      R2,(R0)      ;IS THE BIT PRESENTLY SET ??
56 043746 001402          BEQ      50$          ;NO
57 043750 040210          BIC      R2,(R0)      ;RESET THE BIT
58 043752 000401          BR       60$
59 043754 050210          50$:    BIS      R2,(R0)      ;SET THE BIT
60 043756 006302          60$:    ASL      R2          ;SHIFT TO NEXT BIT
61 043760 001003          BNE      70$
62 043762 012702 000001    MOV      #1,R2        ;CONTINUE WITH FIRST BIT OF NEXT WORD
63 043766 005720          TST      (R0)+
64 043770 005303          70$:    DEC      R3          ;END OF BURST ??
65 043772 001361          BNE      40$          ;NO !!
66
67
68 043774 017600 000010    ;COMPARE WRITE BUFFER TO READ BUFFER
69 044000 062766 000002 000010 80$:    MOV      @10(SP),R0    ;R0 = WRITE BUFFER
70 044006 017601 000010    ADD      #2,10(SP)    ;MOVE SP TO READ ADDRESS
71 044012 062766 000002 000010 MOV      @10(SP),R1    ;R1 = READ BUFFER
72 044020 013702 001336    ADD      #2,10(SP)    ;MOVE SP TO RETURN ADDRESS
73 044024 163702 001412    MOV      RMWCI,R2     ;R2 = NUMBER OF WORDS TRANSFER
74 044030 022021          SUB      RMWCO,R2
75 044032 001003          90$:    CMP      (R0)+,(R1)+  ;COMPARE DATA WORDS
76 044034 005302          BNE      100$        ;EXIT IF NOT EQUAL
77 044036 001374          DEC      R2          ;DECREMENT WORD COUNT
78 044040 000433          BNE      90$        ;CONTINUE IF NOT DONE
79
80
81 044042 014037 001140    BR       110$        ;DONE COMPARE - NO ERROR
82 044046 014137 001142    ;DATA COMPARE FAILED
83 044052 010037 001134    100$:   MOV      -(R0),%GDDAT ;STORE GOOD DATA FOR TYPEOUT
84 044056 010137 001136    MOV      -(R1),%BDDAT ;STORE BAD DATA FOR TYPEOUT
85 044062 010237 001174    MOV      R0,%GDADR    ;STORE ADDRESS OF GOOD DATA
86 044066 062766 000004 000010 MOV      R1,%BDADR    ;STORE ADDRESS OF BAD DATA
87 044074 112776 000336 000010 MOV      R2,%STMP0    ;STORE WORD COUNT OF ERROR
88
89
90 044102 032737 100000 044142 ;CHANGE ERROR NUMBER IF ECC CORRECTION FAILED
91 044110 001403          BIT      #BIT15,150$ ;WAS ECC CORRECTION USED ??
92 044112 112776 000163 000010 BEQ      105$        ;NO !!
93 044120 162766 000002 000010 MOV      #163,@10(SP) ;ECC CORRECTION FAILED
94 044126 000240          105$:   SUB      #2,10(SP)    ;MOVE SP TO RETURN IF ERROR
95 044130          NOP
96 044130 012603          110$:   MOV      (SP)+,R3     ;;POP STACK INTO R3
97 044132 012602          MOV      (SP)+,R2     ;;POP STACK INTO R2
98 044134 012601          MOV      (SP)+,R1     ;;POP STACK INTO R1
99 044136 012600          MOV      (SP)+,R0     ;;POP STACK INTO R0
000207          RTS      PC          ;RETURN TO USER
044142 000000          150$:   .WORD          ;ECC CORRECTION FLAG
  
```

```

1      .SBTTL  GET STATUS SUBROUTINE
2
3      ;THIS SUBROUTINE SETS UP THE 'GET INDEX TABLE' AND THE 'GET
4      ;BUFFER' FOR READING ALL SUBSYSTEM REGISTERS VIA THE GET SUBROUTINE
5      ;AND THEN RETURNS TO THE USER.
6
7      ;CALL:  JSR      PC,GETSTS
8
9
10     GETSTS:
11     MOV      R0,-(SP)      ;;PUSH R0 ON STACK
12     MOV      R1,-(SP)      ;;PUSH R1 ON STACK
13     MOV      R2,-(SP)      ;;PUSH R2 ON STACK
14     MOV      #GETINX,R0    ;R0 = ADDRESS OF INDEX TABLE
15     MOV      #RMEC2I+2,R1  ;R1 = ADDRESS OF GET BUFFER
16     MOV      #RMEC2,R2    ;R2 = REGISTER INDE
17     2$:     MOVB     R2,(R0)+ ;WRITE REGISTER INDEX IN TABLE
18     CLR      -(R1)        ;CLEAR CORRESPONDING LOCATION
19     3$:     SUB      #2,R2  ;DECREMENT TO NEXT INDEX
20     BMI      4$          ;BRANCH OUT IF DONE
21     CMP      #RMDB,R2    ;DONT WRITE RMDB INDEX
22     BNE     2$
23     CLR      -(R1)
24     BR      3$
25     4$:     MOVB     #200,(R0)+ ;WRITE TERMINATOR
26     MOV      (SP)+,R2    ;;POP STACK INTO R2
27     MOV      (SP)+,R1    ;;POP STACK INTO R1
28     MOV      (SP)+,R0    ;;POP STACK INTO R0
29     NOP
30     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 0 IN THE TABLE WILL CAUSE THE SUBROUTINE TO
: READ 'RMB A' 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
      CLR   @ (SP)       ;ERROR CALL
      SUB    #4,(SP)
      MOV    R0,-(SP)     ;;PUSH R0 ON STACK
      MOV    R1,-(SP)     ;;PUSH R1 ON STACK
      MOV    R2,-(SP)     ;;PUSH R2 ON STACK
      MOV    R3,-(SP)     ;;PUSH R3 ON STACK
      MOV    R4,-(SP)     ;;PUSH R4 ON STACK
      MOV    ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
      MOV    ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
      MOV    $BASE,R0
      MOV    #GETBUF,R2
      MOV    #GETINX,R4
      MOV    #5$,ERRVEC   ;SETUP FOR TIMEOUT
      MOV    #PR6,ERRVEC+2
1$:   MOV    RMCS2(R0),$TMP0 ;GET 'NED' STATUS
      MOV    RMCS1(R0),$TMP1 ;GET 'DVA' STATUS
      BIT    #DVA,$TMP1     ;DEVICE AVAILABLE??
      BNE   3$             ;YES!!
      ADD    #4,16(SP)      ;WRITE ERROR NUMBER IN USER'S
      MOVB  #112,@16(SP)   ;ERROR CALL
      BR    7$
3$:   TSTB  (R4)            ;DONE??
      BMI  9$             ;YES!!
      MOVB  (R4),R1        ;R1 = REGISTER ADDRESS
      BIC  #^CIDXMSK,R1   ;CLEAR ANY SIGN EXTENSION
    
```

31	044230	000240		
32	044232	062716	000004	
33	044236	105076	000000	
34	044242	162716	000004	
35	044246	010046		
	044250	010146		
	044252	010246		
	044254	010346		
	044256	010446		
	044260	013746	000004	
	044264	013746	000006	
36	044270	013700	001276	
37	044274	012702	001334	
38	044300	012704	001522	
39	044304	012737	044412	000004
40	044312	012737	000300	000006
41	044320	016037	000010	001174
42	044326	016037	000000	001176
43	044334	032737	004000	001176
44	044342	001007		
45	044344	062766	000004	000016
46	044352	112776	000112	000016
47	044360	000423		
48	044362	105714		
49	044364	100433		
50	044366	111401		
51	044370	042701	177700	

```

52 044374 060001          ADD      R0,R1
53 044376 112403          MOVVB   (R4)+,R3      ;R3 = STORAGE ADDRESS FOR REGISTER
54 044400 042703 177700      BIC     #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
55 044404 060203          ADD     R2,R3
56 044406 011113          MOV     (R1),(R3)    ;READ REGISTER
57 044410 000764          BR      3$
58
59 044412 022626          5$:    CMP     (SP)+,(SP)+  ;RESTORE STACK
60 044414 062766 000004 000016  ADD     #4,16(SP)     ;WRITE ERROR NUMBER IN
61 044422 112776 000007 000016  MOVVB  #7,@16(SP)    ;USER'S ERROR CALL
62 044430 162766 000002 000016  7$:    SUB     #2,16(SP)
63 044436 105714          8$:    TSTB   (R4)        ;DONE CLEARING??
64 044440 100405          BMI     9$          ;YES!!
65 044442 005003          CLR     R3          ;CLEAR REMAINING STORAGE
66 044444 112403          MOVVB  (R4)+,R3     ;LOCATIONS
67 044446 060203          ADD     R2,R3
68 044450 005013          CLR     (R3)
69 044452 000771          BR      8$
70 044454          9$:
    044454 012637 000006      MOV     (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
    044460 012637 000004      MOV     (SP)+,ERRVEC   ;;POP STACK INTO ERRVEC
    044464 012604      MOV     (SP)+,R4       ;;POP STACK INTO R4
    044466 012603      MOV     (SP)+,R3       ;;POP STACK INTO R3
    044470 012602      MOV     (SP)+,R2       ;;POP STACK INTO R2
    044472 012601      MOV     (SP)+,R1       ;;POP STACK INTO R1
    044474 012600      MOV     (SP)+,R0       ;;POP STACK INTO R0
71 044476 000207          MOV     PC
72
    
```

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 044500 000240
29 044502 010046
044504 010146
044506 010246
044510 010346
044512 010446
044514 013746 000004
044520 013746 000006
30 044524 013700 001276
31 044530 012702 001410
32 044534 012704 001551
33 044540 012737 044650 000004
34 044546 012737 000300 000006
35 044554 016037 000010 001174
36 044562 016037 000000 001176
37 044570 032737 004000 001176
38 044576 001007
39 044600 062766 000004 000016
40 044606 112776 000112 000016
41 044614 000424
42 044616 105714
43 044620 100425
44 044622 111401
45 044624 042701 177700
46 044630 060001
47 044632 111403
48 044634 042703 177700
49 044640 060203
50 044642 011311
51 044644 105724

.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    $BASE,R0
      MOV    #PUTBUF,R2
      MOV    #PUTINX,R4
      MOV    #5$,ERRVEC    ;SETUP FOR TIMEOUT
      MOV    #PR6,ERRVEC+2
      1$: MOV    RMCS2(R0),$TMP0 ;GET 'NED' STATUS
      MOV    RMCS1(R0),$TMP1 ;GET 'DVA' STATUS
      BIT    #DVA,$TMP1     ;DEVICE AVAILABLE??
      BNE    3$            ;YES!!
      ADD    #4,16(SP)      ;WRITE ERROR NUMBER IN
      MOVB   #112,@16(SP)  ;USER'S ERROR CALL
      BR    7$
      3$: TSTB   (R4)       ;DONE??
      BMI   9$            ;YES!!
      MOVB   (R4),R1       ;R1 = REGISTER ADDRESS
      BIC   #^CIDXMSK,R1  ;CLEAR ANY SIGN EXTENSION
      ADD   R0,R1
      MOVB   (R4),R3       ;R3 = STORAGE ADDRESS
      BIC   #^CIDXMSK,R3  ;CLEAR ANY SIGN EXTENSION
      ADD   R2,R3
      MOV    (R3),(R1)     ;WRITE REGISTER
      4$: TSTB   (R4)+    ;ADJUST REGISTER POINTER
    
```



```

52 044646 000763          BR      3$
53
54 044650 022626          5$:    CMP      (SP)+,(SP)+      :ADJUST STACK
55 044652 062766 000004 000016  ADD      #4,16(SP)      :WRITE ERROR NUMBER IN
56 044660 112776 000007 000016  MOVB     #7,@16(SP)     :USER'S ERROR CALL
57 044666 162766 000002 000016  7$:    SUB      #2,16(SP)
58
59 044674          9$:
   044674 012637 000006      MOV      (SP)+,ERRVEC+2      ;;POP STACK INTO ERRVEC+2
   044700 012637 000004      MOV      (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC
   044704 012604      MOV      (SP)+,R4          ;;POP STACK INTO R4
   044706 012603      MOV      (SP)+,R3          ;;POP STACK INTO R3
   044710 012602      MOV      (SP)+,R2          ;;POP STACK INTO R2
   044712 012601      MOV      (SP)+,R1          ;;POP STACK INTO R1
   044714 012600      MOV      (SP)+,R0          ;;POP STACK INTO R0
60 044716 000207      RTS      PC                :RETURN
61
  
```

```

1          .SBTTL  SIZE CLOCK SUBROUTINE
2
3          SIZCLK:
4 044720   013746   000004   MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
5 044724   013746   000006   MOV      ERRVEC+2,-(SP)   ;;PUSH ERRVEC+2 ON STACK
6 044730   012737   044766   000004   MOV      #1$,ERRVEC      ;;SET UP FOR BUS TIMEOUT
7 044736   012737   000300   000006   MOV      #PR6,ERRVEC+2
8 044744   012737   177546   001516   MOV      #177546,CLKADR   ;;LOAD ADDRESSES FOR KW11-L
9 044752   012737   000100   001520   MOV      #100,CLKVCT
10 044760  005777   134532   TST     @CLKADR           ;;TEST FOR KW11-L PRESENT
11 044764  000421           BR      3$               ;;YES - KW11-L IS PRESENT
12 044766  022626           1$:   CMP     (SP)+,(SP)+     ;;RESTORE SP
13 044770  012737   045020   000004   MOV      #2$,ERRVEC      ;;SET UP FOR BUS TIMEOUT
14 044776  012737   172540   001516   MOV      #172540,CLKADR   ;;LOAD ADDRESSES FOR KW11-P CLOCK
15 045004  012737   000104   001520   MOV      #104,CLKVCT
16 045012  005777   134500   TST     @CLKADR           ;;TEST FOR KW11-P PRESENT
17 045016  000404           BR      3$               ;;YES - KW11-P IS PRESENT
18 045020  022626           2$:   CMP     (SP)+,(SP)+     ;;RESTORE SP
19 045022  062766   000002   000004   ADD     #2,4(SP)         ;;MOVE RETURN TO ERROR
20 045030           3$:
21 045030  012637   000006   MOV     (SP)+,ERRVEC+2   ;;POP STACK INTO ERRVEC+2
22 045034  012637   000004   MOV     (SP)+,ERRVEC     ;;POP STACK INTO ERRVEC
23 045040  000207           RTS     PC               ;;RETURN TO USER
    
```

```

1      .SBTTL  TIMEOUT SUBROUTINE
2
3      ;THIS SUBROUTINE WAITS FOR RDY = 1 AND GO = 0 OR FOR A TIMEOUT
4      ;GREATER THAN APPROX. 500 MSEC AND THEN RETURNS.
5
6      ;CALL:  JSR      PC,TIMOUT
7      ;      ???
8      ;      RETURN HERE
9
10     TIMOUT:
11     MOV      R0,-(SP)      ;;PUSH R0 ON STACK
12     MOV      R1,-(SP)      ;;PUSH R1 ON STACK
13     MOV      R2,-(SP)      ;;PUSH R2 ON STACK
14     MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
15     MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
16     MOV      #4$,ERRVEC    ;SETUP FOR BUS TIMEOUT - 04 TRAP
17     MOV      #PR6,ERRVEC+2
18     MOV      $BASE,R0      ;R0=BASE ADDRESS
19     MOV      CLKADR,R1     ;R1=CLOCK ADDRESS
20     MOV      #30,R2        ;R2=NUMBER OF CLOCK CYCLES
21     1$:  CMP      R1,#172540 ;KW11-P CLOCK??
22     BNE      2$           ;NO!!
23     MOV      #1,2(R1)     ;SET COUNTER
24     2$:  MOV      #BIT2!BIT0,(R1) ;START COUNTER
25
26     3$:  MOV      RMCS1(R0),-(SP) ;GET STATUS
27     BIC      #^C<RDY!GO>,(SP)
28     CMP      #RDY,(SP)+    ;RDY=1,GO=0??
29     BEQ      5$           ;YES!!
30     BIT      #BIT7,(R1)   ;TIMER DONE??
31     BEQ      3$           ;NO!!
32     DEC      R2           ;DEC NUMBER OF CYCLES
33     BNE      1$           ;CONTINUE IF NOT DONE
34     BR       5$           ;'RDY' DID NOT SET OR 'GO' DID NOT RESET
35
36     4$:  CMP      (SP)+,(SP)+ ;WITHIN 500 MSEC AFTER THE COMMAND WAS ISSUED.
37     ADD      #4,12(SP)    ;ADJUST STACK
38     MOVB     #7,@12(SP)   ;MOVE SP TO USER'S CALL
39     SUB      #2,12(SP)    ;WRITE ERROR NUMBER
40
41     5$:  MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
42     MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
43     MOV      (SP)+,R2      ;;POP STACK INTO R2
44     MOV      (SP)+,R1      ;;POP STACK INTO R1
45     MOV      (SP)+,R0      ;;POP STACK INTO R0
46     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 045226
40
41
42 045226 062716 000004
43 045232 105076 000000
44 045236 162716 000004
45
46
47 045242 013737 001344 001142
48 045250 042737 177770 001142
49 045256 013737 001234 001140
50 045264 042737 177770 001140
51 045272 123737 001140 001142
52
53 045300 001415
54 045302 062716 000004
55 045306 112776 000001 000000
56 045314 162716 000002
57 045320 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.
;
; .($UNIT) CONTAINS THE DRIVE NUMBER
;
;THE SUBROUTINE IS CALLED AS FOLLOWS:
;
;(1) JSR PC,PRIERR          RETURN HERE IF NO ERROR
     BR   ???              RETURN HERE TO REPORT AN ERROR
     NOP                    ERROR NUMBER DEFINED BY SUB
     ERROR                  GO BACK TO SUB FOR MORE ERROR CHECKS
     JSR PC,@(SP)+         RETURN HERE IF NO MORE ERRORS
     ???
;
PRIERR:
;CLEAR USER'S ERROR CALL
     ADD #4,(SP)           ;MOVE (SP) TO ERROR CALL
     CLRB @ (SP)          ;CLEAR ERROR NUMBER
     SUB #4,(SP)          ;MOVE (SP) TO NO ERROR RETURN
;REPORT AN ERROR IF THE WRONG UNIT IS SELECTED
     MOV RMCS2I,$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
     BEQ 1$              ;YES!!
     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 045322 162716 000010          SUB      #10,(SP)          ;RESTORE (SP)
59 045326 000240          NOP
60 045330 000137 046050          JMP      10$              ;SKIP OTHER CHECKS
61 045334
62
63                                ;REPORT AN ERROR IF THE DEVICE IS NOT AVAILABLE OR IF
64                                ;THE DEVICE IS NONEXISTANT
65 045334 032737 004000 001334          BIT      #DVA,RMCS1I      ;DEVICE AVAILABLE??
66 045342 001045          BNE      5$              ;YES!!
67 045344 013737 001334 001140          MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
68 045352 052737 004000 001140          BIS      #DVA,$GDDAT
69 045360 013737 001334 001142          MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
70 045366 062716 000004          ADD      #4,(SP)
71 045372 112776 000002 000000          MOV      #2,@(SP)        ;ERROR #2
72 045400 032737 010000 001344          BIT      #NED,RMCS2I      ;WAS NED SET??
73 045406 001414          BEQ      2$              ;NO!!
74 045410 013737 001344 001140          MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
75 045416 013737 001344 001142          MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
76 045424 042737 010000 001140          BIC      #NED,$GDDAT
77 045432 112776 000003 000000          MOV      #3,@(SP)        ;YES - CHANGE ERROR NUMBER
78 045440 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
79 045444 004736          JSR      PC,@(SP)+        ;REPORT DEVICE NOT AVAILABLE
80 045446 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
81 045452 000240          NOP
82 045454 000575          BR       10$              ;SKIP OTHER CHECKS
83 045456
84
85                                ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
86 045456 032737 000200 001334          BIT      #RDY,RMCS1I      ;CONTROLLER READY??
87 045464 001030          BNE      7$              ;YES!!
88 045466 013737 001334 001140          MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
89 045474 052737 000200 001140          BIS      #RDY,$GDDAT
90 045502 042737 160001 001140          BIC      #SC!TRC!MCPE!GO,$GDDAT
91 045510 013737 001334 001142          MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
92 045516 062716 000004          ADD      #4,(SP)
93 045522 112776 000004 000000          MOV      #4,@(SP)        ;ERROR #4
94 045530 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
95 045534 004736          JSR      PC,@(SP)+        ;REPORT CONTROLLER NOT READY
96 045536 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
97 045542 000240          NOP
98 045544 000541          BR       10$              ;SKIP OTHER CHECKS
99 045546
100
101                               ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
102 045546 032737 000001 001334          BIT      #GO,RMCS1I       ;GO RESET??
103 045554 001431          BEQ      8$              ;YES!!
104 045556 032737 000200 001346          BIT      #DRY,RMDSI       ;DRIVE READY??
105 045564 001025          BNE      8$              ;YES!!
106 045566 013737 001334 001140          MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
107 045574 042737 160001 001140          BIC      #SC!TRC!MCPE!GO,$GDDAT
108 045602 013737 001334 001142          MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
109 045610 062716 000004          ADD      #4,(SP)
110 045614 112776 000005 000000          MOV      #5,@(SP)        ;ERROR #5
111 045622 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
112 045626 004736          JSR      PC,@(SP)+        ;REPORT DRIVE NOT READY
113 045630 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
114 045634 000240          NOP
    
```

```

115 045636 000504          BR      10$
116 045640          8$:
117
118          ;REPORT AN ERROR IF THE RH CONTROLLER DETECTED BAD
119          ;PARITY ON THE MASSBUS CONTROL BUS
120 045640 032737 020000 001334  BIT      #MCPE,RMCS1I  ;PARITY ERROR ??
121 045646 001425          BEQ      9$           ;NO!!
122 045650 013737 001334 001140  MOV      RMCS1I,$GDDAT ;EXPECTED STATUS
123 045656 042737 160001 001140  BIC      #SC!TRE!MCPE!GO,$GDDAT
124 045664 013737 001334 001142  MOV      RMCS1I,$BDDAT ;RECEIVED STATUS
125 045672 062716 000004          ADD      #4,(SP)       ;MOVE STACK TO USER'S ERROR
126 045676 112776 000013 000000  MOVB    #13,@(SP)     ;ERROR #13
127 045704 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
128 045710 004736          JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
129 045712 162716 000010          SUB      #10,(SP)     ;RESTORE STACK
130 045716 000240          NOP
131 045720 000453          BR      10$
132 045722          9$:
133
134          ;REPORT AN ERROR IF DETECTED A CONTROL BUS PARITY ERROR
135 045722 032737 000010 001350  BIT      #PAR,RMER1I  ;WAS THERE A PARITY ERROR??
136 045730 001451          BEQ      11$         ;NO!!
137 045732 032737 000010 001376  BIT      #DPE,RMER2I  ;WAS IT THE CONTROL BUS??
138 045740 001045          BNE     11$         ;NOT SURE!!
139 045742 032737 000010 001424  BIT      #PAR,RMER10  ;DID TEST SET PAR ??
140 045750 001413          SEQ      93$         ;NO!!
141 045752 010046          MOV      R0,-(SP)     ;:PUSH R0 ON STACK
142 045754 012700 001551          MOV      #PUTINX,R0   ;R0 POINTS TO INDEX TABLE
143 045760 122710 000014          91$:  CMPB    #RMER1,(R0)  ;SEARCH TABLE FOR RMER1
144 045764 001002          BNE     92$
145 045766 012600          MOV      (SP)+,R0     ;:POP STACK INTO R0
146 045770 000431          BR      11$         ;PAR WAS SET BY TEST
147 045772 105720          92$:  TSTB    (R0)+       ;END OF TABLE??
148 045774 100371          BPL     91$         ;NO!!
149 045776 012600          MOV      (SP)+,R0     ;:POP STACK INTO R0
150 046000 013737 001350 001140  93$:  MOV      RMER1I,$GDDAT ;EXPECTED STATUS
151 046006 042737 000010 001140  BIC      #PAR,$GDDAT
152 046014 013737 001350 001142  MOV      RMER1I,$BDDAT ;RECEIVED STATUS
153 046022 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
154 046026 112776 000050 000000  MOVB    #50,@(SP)     ;WRITE THE ERROR NUMBER
155 046034 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
156 046040 004736          JSR      PC,@(SP)+    ;REPORT THE ERROR
157 046042 162716 000010          SUB      #10,(SP)     ;MOVE SP TO NO ERROR RETURN
158 046046 000240          NOP
159 046050 062716 000010          10$:  ADD      #10,(SP)     ;RETURN TO ERROR
160 046054 000240          11$:  NOP
161 046056 000207          RTS      PC
162
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24 046060
25
26
27
28 046060 013737 001410 051714
29 046066 042737 177701 051714
30 046074 062716 000004
31 046100 105076 000000
32 046104 162716 000004
33
34
35
36
37
38 046110 032737 000200 001346
39 046116 001024
40 046120 013737 001346 001142
41 046126 042737 177577 001142
42 046134 012737 000200 001140
43 046142 062716 000004
44 046146 112776 000010 000000
45 046154 162716 000002
46 046160 004736
47 046162 162716 000010
48 046166 000240
49
50
51 046170 032737 000001 001334
52 046176 001423
53 046200 013737 001334 001142
54 046206 042737 177776 001142
55 046214 005037 001140
56 046220 062716 000004
57 046224 112776 000011 000000

```

.SBTTL SECONDARY ERROR CHECK SUBROUTINE

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

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

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

SECERR:

;*****
;STORE FUNCTION CODE AND CLEAR USER'S ERROR NUMBER
MOV RMCS10,515$ ;STORE FUNCTION CODE
BIC #^C<F0!F1!F2!F3!F4>,515$
ADD #4,(SP) ;MOVE (SP) TO ERROR CALL
.CLRB @(SP) ;CLEAR ERROR NUMBER
SUB #4,(SP) ;MOVE (SP) TO NO ERROR RETURN
;*****

;CHECK SECONDARY ERRORS COMMON TO ALL COMMANDS

;REPORT ERROR IF DRIVE IS NOT READY, I.E., IF DRY = 0
BIT #DRY,RMDSI ;DRIVE READY??
BNE 5$ ;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
5$: BIT #GO,RMCS11 ;GO BIT RESET??
BEQ 10$ ;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
    
```

```

58 046232 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
59 046236 004736                JSR      PC,@(SP)+      ;REPORT DEVICE NOT AVAILABLE
60 046240 162716 000010          SUB      #10,(SP)       ;RESTORE (SP)
61 046244 000240                NOP
62
63                                ;REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
64 046246 013737 001334 001142 10$:  MOV      RMCS1I,$BDDAT ;IS FUNCTION CODE CORRECT??
65 046254 042737 177701 001142    BIC      #^C76,$BDDAT
66 046262 013737 051714 001140    MOV      515$,$GDDAT   ;EXPECTED FUNCTION CODE
67 046270 023737 001142 001140    CMP      $BDDAT,$GDDAT
68 046276 001413                BEQ      15$           ;YES!!
69 046300 062716 000004          ADD      #4,(SP)
70 046304 112776 000012 000000    MOVB    #12,@(SP)      ;ERROR NUMBER
71 046312 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
72 046316 004736                JSR      PC,@(SP)+      ;REPORT WRONG FUNCTION CODE
73 046320 162716 000010          SUB      #10,(SP)       ;RESTORE (SP)
74 046324 000240                NOP
75 046326
76                                15$:
77                                ;REPORT AN ERROR IF COMPOSITE ERROR IS SET AND NO OTHER
78                                ;ERRORS ARE SET, OR IF COMPOSITE ERROR IS NOT SET AND
79                                ;OTHER ERRORS ARE SET
79 046326 005037 001140          CLR      $GDDAT        ;EXPECT 'ERR' = 0
80 046332 005737 001350          TST      RMER1I        ;IS RMER1 = 0??
81 046336 001003                BNE      20$           ;NO!!
82 046340 005737 001376          TST      RMER2I        ;IS RMERZ = 0??
83 046344 001403                BEQ      25$           ;YES!!
84 046346 052737 040000 001140 20$:  BIS      #ERR,$GDDAT    ;'ERR' SOULD BE SET
85 046354 013737 001346 001142 25$:  MOV      RMDSI,$BDDAT
86 046362 042737 137777 001142    BIC      #^CERR,$BDDAT
87 046370 023737 001140 001142    CMP      $GDDAT,$BDDAT ;IS 'ERR' OK??
88 046376 001412                BEQ      30$           ;YES!!
89 046400 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR
90 046404 112776 000047 000000    MOVB    #47,@(SP)      ;WRITE ERROR NUMBER
91 046412 162716 000002          SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
92 046416 004736                JSR      PC,@(SP)+      ;REPORT INVALID COMP ERROR
93 046420 162716 000010          SUB      #10,(SP)
94
95                                ;REPORT AN ERROR IF 'TRE' IS SET AND NONE OF THE BITS WHICH SET
96                                ;TRE IS SET, OR IF TRE IS NOT SET AND ONE OR MORE BITS WHICH
97                                ;SET TRE IS SET
98 046424 005037 001140 30$:  CLR      $GDDAT        ;EXPECT 'TRE' = 0
99 046430 013746 001344          MOV      RMCS2I,-(SP)  ;WAS DLT, WCE, UPE, NED, NEM
100 046434 042726 000377         BIC      #377,(SP)+    ;PGE, MXF OR MDPE SET
101 046440 001010                BNE      35$           ;YES!!
102 046442 032737 040000 001346    BIT      #ERR,RMDSI    ;WAS EXCEPTION RECEIVED??
103 046450 001407                BEQ      40$           ;NO!!
104 046452 022737 000030 051714    CMP      #SEARCH,515$ ;WAS DATA TRANSFERRED??
105 046460 103003                BHIS    40$           ;NO!!
106 046462 052737 040000 001140 35$:  BIS      #TRE,$GDDAT   ;'TRE' SHOULD BE SET
107 046470 013737 001334 001142 40$:  MOV      RMCS1I,$BDDAT ;BAD DATA FOR TYPEOUT
108 046476 042737 137777 001142    BIC      #^CTRE,$BDDAT
109 046504 023737 001140 001142    CMP      $GDDAT,$BDDAT ;IS 'TRE' OK??
110 046512 001413                BEQ      45$           ;YES!!
111 046514 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
112 046520 112776 000014 000000    MOVB    #14,@(SP)      ;WRITE ERROR NUMBER
113 046526 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
114 046532 004736                JSR      PC,@(SP)+      ;REPORT TRE ERROR
    
```



```

115 046534 162716 000010          SUB #10,(SP)      ;RESTORE (SP)
116 046540 000240                NOP
117 046542                45$:
118
119
120
121
122
123
124
125
126
127
128 046542 010046                ;GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE
129 046544 013700 051714          MOV RO,-(SP)      ;:PUSH RO ON STACK
130 046550 016037 071516 051706  MOV 515$,RO      ;:GET FUNCTION CODE
131 046556 012600                MOV FNCDTB(RO),500$ ;:STORE ENTRY
132
133
134
135 046560 013737 051706 001140  MOV 500$,SGDDAT  ;:GET EXPECTED ATA STATUS
136 046566 032737 040000 001346  BIT #ERR,RMDSI   ;:IS COMPOSITE ERROR SET ??
137 046574 001403                BEQ 50$          ;:NO !!
138 046576 052737 100000 001140  BIS #ATA,$GDDAT ;:EXPECT AN ATTENTION
139 046604 042737 077777 001140 50$: BIC #^CATA,$GDDAT ;:STRIP DONT CARES
140 046612 013737 001346 001142  MOV RMDSI,$BDDAT ;:GET RECEIVED ATA
141 046620 042737 077777 001142  BIC #^CATA,$BDDAT ;:STRIP DONT CARES
142 046626 023737 001140 001142  CMP $GDDAT,$BDDAT ;:IS ATA OK ??
143 046634 001413                BEQ 55$          ;:YES !!
144 046636 062716 000004          ADD #4,(SP)      ;:MOVE SP TO USERS ERROR CALL
145 046642 112776 000006 000000  MOVB #6,@(SP)   ;:LOAD ERROR # IN CALL
146 046650 162716 000002          SUB #2,(SP)      ;:MOVE SP TO ERROR RETURN
147 046654 004736                JSR PC,@(SP)+    ;:REPORT ERROR
148 046656 162716 000010          SUB #10,(SP)     ;:RESTORE SP
149 046662 000240                NOP
150 046664                55$:
151
152
153
154 046664 013737 051706 001140  ;:REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
155 046672 042737 177776 001140  ;:WITH FUNCTION CODE TABLE
156 046700 013737 001350 001142  MOV 500$,SGDDAT  ;:GET EXPECTED ILF
157 046706 042737 177776 001142  BIC #^CILF,$GDDAT ;:CLEAR ALL OTHER BITS
158 046714 023737 001140 001142  MOV RMER1I,$BDDAT ;:GET RECEIVED ILF
159 046722 001412                BIC #^CILF,$BDDAT ;:CLEAR ALL OTHER BITS
160 046724 062716 000004          CMP $GDDAT,$BDDAT ;:IS ILF OK ??
161 046730 112776 000254 000000  BEQ 60$          ;:YES !!
162 046736 162716 000002          ADD #4,(SP)      ;:MOVE SP TO USERS ERROR CALL
163 046742 004736                MOVB #254,@(SP)  ;:WRITE ERROR NUMBER IN CALL
164 046744 162716 000010          SUB #2,(SP)      ;:MOVE SP TO ERROR RETURN
165 046750 005037 001140          JSR PC,@(SP)+    ;:REPORT ERROR AND RETURN
166
167
168 046754 013746 051706          60$: CLR $GDDAT      ;:CLEAR EXPECTED STATUS
169 046760 052716 137777          ;:REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
170 046764 013737 001344 001142  MOV 500$,-(SP)   ;:GET WCE STATUS ENABLE
171 046772 042637 001142          BIS #^CWCE,(SP)  ;:SET ALL OTHER BITS
                                MOV RMCS2I,$BDDAT ;:RECEIVED STATUS
                                BIC (SP)+,$BDDAT ;:CLEAR WCE IF ENABLED
    
```

```

172 046776 001412          BEQ      90$          ;BRANCH IF WCE OK
173 047000 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
174 047004 112776 000026 000000    MOVVB   #26,@(SP)    ;WRITE ERROR NUMBER
175 047012 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
176 047016 004736          JSR      PC,@(SP)+    ;REPORT ERROR
177 047020 162716 000010          SUB      #10,(SP)     ;RESTORE ERROR
178 047024          90$:
179
180          ;REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
181 047024 013746 051706    MOV      500$,-(SP)   ;GET OPI STATUS ENABLE
182 047030 052716 157777    BIS      #^COPI,(SP)  ;SET ALL OTHER BITS
183 047034 013737 001350 001142    MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
184 047042 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR OPI IF ENABLED
185 047046 001412          BEQ      100$         ;BRANCH IF OPI OK
186 047050 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
187 047054 112776 000164 000000    MOVVB   #164,@(SP)   ;WRITE ERROR NUMBER IN CALL
188 047062 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
189 047066 004736          JSR      PC,@(SP)+    ;REPORT ERROR
190 047070 162716 000010          SUB      #10,(SP)     ;RESTORE SP
191 047074          100$:
192
193          ;REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
194          ;SET AND VV IS NOT RESET
195 047074 013746 051706    MOV      500$,-(SP)   ;GET IVC STATUS ENABLE
196 047100 032737 000100 001346    BIT      #VV,RMDSI    ;IS VV SET
197 047106 001402          BEQ      105$         ;NO !!
198 047110 042716 010000    BIC      #IVC,(SP)    ;YES - IVC SHOULD BE 0
199 047114 052716 167777    BIS      #^CIVC,(SP)  ;SET ALL OTHER BITS
200 047120 013737 001376 001142    MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
201 047126 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR IVC IF ENABLED
202 047132 001412          BEQ      110$         ;BRANCH IF IVC OK
203 047134 062716 000004    ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
204 047140 112776 000165 000000    MOVVB   #165,@(SP)   ;WRITE ERROR NUMBER IN CALL
205 047146 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
206 047152 004736          JSR      PC,@(SP)+    ;REPORT ERROR
207 047154 162716 000010          SUB      #10,(SP)     ;RESTORE SP TO NO ERROR
208 047160          110$:
209
210          ;BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
211          ; ALL WRITE ERRORS, I.E.,
212          ; RMER1 - WLE, WCF
213          ; RMER2 - DPE
214          ; RMCS2 - UPE.
215          ;EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
216          ;WRITE ERROR ENABLE BIT IS RESET.
217
218          ;REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
219          ;THE DRIVE IS NOT WRITE PROTECTED
220 047160 012746 177777          MOV      #-1,-(SP)    ;ASSUME WRITE ERRORS ENABLED
221 047164 032737 004000 051706    BIT      #WLE,500$    ;ARE WRITE ERRORS ENABLED ??
222 047172 001404          BEQ      115$         ;NO !!
223 047174 032737 004000 001346    BIT      #WRL,RMDSI   ;IS THE DRIVE WRITE PROTECTED ??
224 047202 001002          BNE      120$         ;YES !!
225 047204 042716 004000 115$:    BIC      #WLE,(SP)    ;RESET WLE ENABLE
226 047210 013737 001350 001142 120$:    MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
227 047216 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR WLE IF ENABLED
228 047222 001412          BEQ      125$         ;BRANCH IF WLE OK

```

```

229 047224 062716 000004          ADD    #4,(SP)          ;MOVE SP TO USERS ERROR CALL
230 047230 112776 000023 000000    MOVB   #23,@(SP)       ;WRITE ERROR NUMBER IN CALL
231 047236 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
232 047242 004736          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
233 047244 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
234 047250          125$:
235
236          ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
237 047250 012746 177777          MOV    #-1,-(SP)      ;ASSUME WRITE ERRORS ENABLED
238 047254 032737 004000 051706    BIT    #WLE,500$      ;ARE WRITE ERRORS ENABLED ??
239 047262 001002          BNE    130$           ;YES !!
240 047264 042716 000040          BIC    #WCF,(SP)      ;DISABLE WCF ERROR
241 047270 013737 001350 001142 130$:  MOV    RMER1I,$BDDAT  ;GET RECEIVED STATUS
242 047276 042637 001142          BIC    (SP)+,$BDDAT   ;RESET WCF IF ENABLED
243 047302 001412          BEQ    135$           ;BRANCH IF WCF OK
244 047304 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USERS ERROR CALL
245 047310 112776 000025 000000    MOVB   #25,@(SP)       ;WRITE ERROR NUMBER IN CALL
246 047316 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
247 047322 004736          JSR    PC,@(SP)+       ;REPORT ERROR
248 047324 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
249 047330          135$:
250
251          ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
252 047330 012746 177777          MOV    #-1,-(SP)      ;ASSUME WRITE ERRORS ARE ENABLED
253 047334 032737 004000 051706    BIT    #WLE,500$      ;ARE WRITE ERRORS ENABLED ??
254 047342 001002          BNE    140$           ;YES !!
255 047344 042716 000010          BIC    #DPE,(SP)      ;RESET DPE ENABLE
256 047350 013737 001376 001142 140$:  MOV    RMER2I,$BDDAT  ;GET RECEIVED STATUS
257 047356 042637 001142          BIC    (SP)+,$BDDAT   ;RESET DPE IF ENABLED
258 047362 001412          BEQ    145$           ;BRANCH IF DPE OK
259 047364 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USERS ERROR CALL
260 047370 112776 000040 000000    MOVB   #40,@(SP)       ;WRITE ERROR NUMBER IN CALL
261 047376 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
262 047402 004736          JSR    PC,@(SP)+       ;REPORT ERROR
263 047404 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
264 047410          145$:
265
266          ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
267 047410 012746 177777          MOV    #-1,-(SP)      ;ASSUME WRITE ERRORS ARE ENABLED
268 047414 032737 004000 051706    BIT    #WLE,500$      ;ARE WRITE ERRORS ENABLED ??
269 047422 001002          BNE    150$           ;YES !!
270 047424 042716 020000          BIC    #UPE,(SP)      ;DISABLE UPE ERROR
271 047430 013737 001344 001142 150$:  MOV    RMCS2I,$BDDAT  ;GET RECEIVED STATUS
272 047436 042637 001142          BIC    (SP)+,$BDDAT   ;RESET UPE IF ENABLED
273 047442 001412          BEQ    155$           ;BRANCH IF UPE OK
274 047444 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USERS ERROR CALL
275 047450 112776 000024 000000    MOVB   #24,@(SP)       ;WRITE ERROR NUMBER IN CALL
276 047456 162716 000002          SUB    #2,(SP)         ;MOVE SP TO ERROR RETURN
277 047462 004736          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
278 047464 162716 000010          SUB    #10,(SP)        ;MOVE SP TO NO ERROR
279 047470          155$:
280
281          ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
282 047470 013746 051706          MOV    500$,-(SP)     ;GET IAE ENABLE
283 047474 052716 175777          BIS    #^CIAE,(SP)    ;SET ALL OTHER BITS
284 047500 013737 001350 001142    MOV    RMER1I,$BDDAT  ;GET RECEIVED STATUS
285 047506 042637 001142          BIC    (SP)+,$BDDAT   ;CLEAR IAE IF ENABLED
    
```

```

286 047512 001412          BEQ      160$          ;BRANCH IF IAE IS OK
287 047514 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
288 047520 112776 000166 000000    MOVB    #166,a(SP)    ;WRITE ERROR NUMBER
289 047526 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
290 047532 004736          JSR      PC,a(SP)+     ;REPORT ERROR AND RETURN
291 047534 162716 000010    SUB      #10,(SP)     ;MOVE SP TO NO ERROR
292 047540          160$:
293
294          ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
295          ; ALL READ/WRITE ERRORS, I.E.,
296          ;
297          ; RMCS1 - TRE
298          ; RMCS2 - DLT,NEM,MXF
299          ; RMDS - LBT
300          ; RMER1 - AOE
301          ;NOTE:
302          ; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
303          ;CYLINDER REGISTER IS WRITTEN
304          ;NOTE:
305          ; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
306          ;NOTE:
307          ; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
308
309          ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
310 047540 012746 177777    MOV      #-1,-(SP)    ;ASSUME ERRORS ARE ENABLED
311 047544 032737 001000 051706    BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
312 047552 001002          BNE      165$          ;YES !!
313 047554 042716 100000    BIC      #DLT,(SP)    ;RESET DLT ENABLE
314 047560 013737 001344 001142 165$:    MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
315 047566 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
316 047572 001412          BEQ      170$          ;BRANCH IF DLT IS OK
317 047574 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
318 047600 112776 000032 000000    MOVB    #32,a(SP)    ;WRITE ERROR NUMBER IN CALL
319 047606 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
320 047612 004736          JSR      PC,a(SP)+     ;REPORT ERROR AND RETURN
321 047614 162716 000010    SUB      #10,(SP)     ;MOVE SP TO NO ERROR
322 047620          170$:
323
324          ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
325 047620 012746 177777    MOV      #-1,-(SP)    ;ASSUME ERRORS ARE ENABLED
326 047624 032737 001000 051706    BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
327 047632 001002          BNE      175$          ;YES !!
328 047634 042716 004000    BIC      #NEM,(SP)    ;DISABLE NEM
329 047640 013737 001344 001142 175$:    MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
330 047646 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
331 047652 001412          BEQ      180$          ;BRANCH IF NEM IS OK
332 047654 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
333 047660 112776 000167 000000    MOVB    #167,a(SP)    ;WRITE ERROR NUMBER IN CALL
334 047666 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
335 047672 004736          JSR      PC,a(SP)+     ;REPORT ERROR AND RETURN
336 047674 162716 000010    SUB      #10,(SP)     ;MOVE SP TO NO ERROR
337 047700          180$:
338
339          ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
340 047700 012746 177777    MOV      #-1,-(SP)    ;ASSUME ERRORS ARE ENABLED
341 047704 032737 001000 051706    BIT      #AOE,500$    ;ARE DATA ERRORS ENABLED ??
342 047712 001002          BNE      185$          ;YES !!
    
```

```

343 047714 042716 001000          BIC    #MXF,(SP)      ;DISABLE MXF ERROR
344 047720 013737 001344 001142 185$: MOV    RMCS2I,$BDDAT  ;GET RECEIVED STATUS
345 047726 042637 001142          BIC    (SP)+,$BDDAT  ;CLEAR MXF IF ENABLED
346 047732 001412          BEQ    190$          ;BRANCH IF MXF IS OK
347 047734 062716 000004          ADD    #4,(SP)       ;MOVE SP TO USERS ERROR CALL
348 047740 112776 000033 000000  MOVB   #33,@(SP)     ;WRITE ERROR NUMBER IN CALL
349 047746 162716 000002          SUB    #2,(SP)       ;MOVE SP TO ERROR RETURN
350 047752 004736          JSR    PC,@(SP)+     ;REPORT ERROR AND RETURN
351 047754 162716 000010          SUB    #10,(SP)      ;MOVE SP TO NO ERROR
352 047760          190$:
353
354          ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
355 047760 012746 177777          MOV    #-1,-(SP)    ;ASSUME DATA ERRORS ARE ENABLED
356 047764 032737 001000 051706  BIT    #AOE,500$    ;ARE DATA ERRORS EANBLED ??
357 047772 001404          BEQ    191$          ;NO !!
358 047774 032737 002000 001346  BIT    #LBT,RMDSI   ;IS LBT ALSO SET ??
359 050002 001002          BNE    195$          ;YES !!
360 050004 042716 001000 191$: BIC    #AOE,(SP)     ;DISABLE AOE
361 050010 013737 001350 001142 195$: MOV    RMER1I,$BDDAT ;GET RECEIVED STATUS
362 050016 042637 001142          BIC    (SP)+,$BDDAT ;CLEAR AOE IF ENABLED
363 050022 001412          BEQ    200$          ;BRANCH IF AOE IS OK
364 050024 062716 000004          ADD    #4,(SP)       ;MOVE SP TO USERS ERROR CALL
365 050030 112776 000020 000000  MOVB   #20,@(SP)    ;WRITE ERROR NUMBER
366 050036 162716 000002          SUB    #2,(SP)       ;MOVE SP TO ERROR RETURN
367 050042 004736          JSR    PC,@(SP)+     ;REPORT ERROR AND RETURN
368 050044 162716 000010          SUB    #10,(SP)      ;MOVE SP TO NO ERROR
369 050050          200$:
370
371          ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
372          ;HEADER ERRORS, I.E.,
373          ; RMER1 - HCRC,HCE,FER
374          ; RMER2 - BSE
375
376          ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
377 050050 032737 002000 001366  BIT    #HCI,RMOFI   ;IS HCI SET ??
378 050056 001403          BEQ    201$          ;NO !!
379 050060 042737 000200 051706  BIC    #HCE,500$    ;YES - DISABLE ALL HEADER ERRORS
380 050066          201$:
381
382          ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
383 050066 012746 177777          MOV    #-1,-(SP)    ;ASSUME ERRORS ENABLED
384 050072 032737 000200 051706  BIT    #HCE,500$    ;ARE HEADER ERRORS ENABLED ??
385 050100 001002          BNE    205$          ;YES !!
386 050102 042716 000400          BIC    #HCRC,(SP)   ;DISABLE HCRC
387 050106 013737 001350 001142 205$: MOV    RMER1I,$BDDAT ;GET RECEIVED STATUS
388 050114 042637 001142          BIC    (SP)+,$BDDAT ;RESET HCRC IF ENABLED
389 050120 001412          BEQ    210$          ;BRANCH IF HCRC IS OK
390 050122 062716 000004          ADD    #4,(SP)       ;MOVE SP TO USERS ERROR CALL
391 050126 112776 000035 000000  MOVB   #35,@(SP)    ;WRITE ERROR NUMBER IN CALL
392 050134 162716 000002          SUB    #2,(SP)       ;MOVE SP TO ERROR RETURN
393 050140 004736          JSR    PC,@(SP)+     ;REPORT ERROR AND RETURN
394 050142 162716 000010          SUB    #10,(SP)      ;MOVE SP TO NO ERROR
395 050146          210$:
396
397          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
398 050146 012746 177777          MOV    #-1,-(SP)    ;ASSUME ERRORS ENABLED
399 050152 032737 000200 051706  BIT    #HCE,500$    ;ARE ERRORS ENABLED ??
    
```

```

400 050160 001002          BNE      215$          :YES !!
401 050162 042716 000200  BIC      #HCE,(SP)    :DISABLE HCE
402 050166 013737 001350 001142 215$:  MOV      RMER1I,$BDDAT :GET RECEIVED STATUS
403 050174 042637 001142  BIC      (SP)+,$BDDAT :CLEAR HCE IF ENABLED
404 050200 001412          BEQ      220$          :BRANCH IF HCE IS OK
405 050202 062716 000004  ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
406 050206 112776 000036 000000  MOVB     #36,@(SP)    :WRITE ERROR NUMBER IN CALL
407 050214 162716 000002  SUB      #2,(SP)      :MOVE SP TO ERROR RETURN
408 050220 004736          JSR      PC,@(SP)+    :REPORT ERROR AND RETURN
409 050222 162716 000010  SUB      #10,(SP)     :MOVE SP TO NO ERROR
410 050226          220$:
411
412          :REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
413 050226 012746 177777  MOV      #-1,-(SP)    :ASSUME FER IS ENABLED
414 050232 032737 000200 051706  BIT      #HCE,500$    :ARE HEADER ERRORS ENABLED ??
415 050240 001002          BNE      225$          :YES !!
416 050242 042716 000020  BIC      #FER,(SP)    :DISABLE FER
417 050246 013737 001350 001142 225$:  MOV      RMER1I,$BDDAT :GET RECEIVED STATUS
418 050254 042637 001142  BIC      (SP)+,$BDDAT :RESET FER IF ENABLED
419 050260 001412          BEQ      230$          :BRANCH IF FER OK
420 050262 062716 000004  ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
421 050266 112776 000037 000000  MOVB     #37,@(SP)    :WRITE ERROR NUMBER IN CALL
422 050274 162716 000002  SUB      #2,(SP)      :MOVE SP TO ERROR RETURN
423 050300 004736          JSR      PC,@(SP)+    :REPORT ERROR AND RETURN
424 050302 162716 000010  SUB      #10,(SP)     :MOVE SP TO NO ERROR
425 050306          230$:
426
427          :REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
428 050306 012746 177777  MOV      #-1,-(SP)    :ASSUME ERRORS ENABLED
429 050312 032737 000200 051706  BIT      #HCE,500$    :ARE THEY ENABLED ??
430 050320 001002          BNE      235$          :YES !!
431 050322 042716 100000  BIC      #BSE,(SP)    :DISABLE BSE
432 050326 013737 001376 001142 235$:  MOV      RMER2I,$BDDAT :GET RECEIVED STATUS
433 050334 042637 001142  BIC      (SP)+,$BDDAT :CLEAR BSE IF ENABLED
434 050340 001412          BEQ      240$          :BRANCH IF BSE OK
435 050342 062716 000004  ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
436 050346 112776 000354 000000  MOVB     #354,@(SP)   :WRITE ERROR NUMBER
437 050354 162716 000002  SUB      #2,(SP)      :MOVE SP TO ERROR RETURN
438 050360 004736          JSR      PC,@(SP)+    :REPORT ERROR AND RETURN
439 050362 162716 000010  SUB      #10,(SP)     :MOVE SP TO NO ERROR
440 050366          240$:
441
442          :BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
443          :FIELD ERRORS, I.E.,
444          :      RMCS2 - MDPE
445          :      RMER1 - DCK,ECH
446          :NOTE:
447          :      ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
448          :      DCK IS SET.
449
450          :REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
451 050366 012746 177777  MOV      #-1,-(SP)    :ASSUME ENABLED
452 050372 032737 000100 051706  BIT      #ECH,500$    :ARE DATA FIELD ERRORS ENABLED ??
453 050400 001002          BNE      245$          :YES !!
454 050402 042716 000400  BIC      #MDPE,(SP)   :DISBALE MDPE
455 050406 013737 001344 001142 245$:  MOV      RMCS2I,$BDDAT :GET RECEIVED STATUS
456 050414 042637 001142  BIC      (SP)+,$BDDAT :CLEAR MDPE IF ENABLED
    
```

```

457 050420 001412          BEQ      250$          ;BRANCH IF MDPE OK
458 050422 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
459 050426 112776 000027 000000    MOVVB   #27,@(SP)      ;WRITE ERROR NUMBER IN CALL
460 050434 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
461 050440 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
462 050442 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
463 050446          250$:
464
465          ;REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
466 050446 012746 177777    MOV      #-1,-(SP)      ;ASSUME ENABLED
467 050452 032737 000100 051706    BIT      #ECH,500$      ;ARE THEY ENABLED ??
468 050460 001002          BNE      255$          ;YES !!
469 050462 042716 100000    BIC      #DCK,(SP)      ;DISABLE DCK
470 050466 013737 001350 001142 255$:    MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
471 050474 042637 001142    BIC      (SP)+,$BDDAT   ;CLEAR DCK IF ENABLED
472 050500 001412          BEQ      260$          ;BRANCH IF DCK IS OK
473 050502 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
474 050506 112776 000030 000000    MOVVB   #30,@(SP)      ;WRITE ERROR NUMBER IN CALL
475 050514 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
476 050520 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
477 050522 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
478 050526          260$:
479
480          ;REPORT ERROR IF ECH IS SET AND,
481          ;DATA FIELD ERRORS ARE NOT ENABLED, OR
482          ;ECI IS SET, OR
483          ;DCK IS NOT SET.
484 050526 012746 177777    MOV      #-1,-(SP)      ;ASSUME ENABLED
485 050532 032737 000100 051706    BIT      #ECH,500$      ;ARE ERRORS ENABLED ??
486 050540 001410          BEQ      265$          ;NO !!
487 050542 032737 004000 001366    BIT      #ECI,RMOFI     ;IS ECI SET ??
488 050550 001004          BNE      265$          ;YES !!
489 050552 032737 100000 001350    BIT      #DCK,RMER1I    ;IS DCK ALSO SET ??
490 050560 001002          BNE      270$          ;YES !!
491 050562 042716 000100 265$:    BIC      #ECH,(SP)      ;DISABLE ECH
492 050566 013737 001350 001142 270$:    MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
493 050574 042637 001142    BIC      (SP)+,$BDDAT   ;CLEAR ECH IF ENABLED
494 050600 001412          BEQ      275$          ;BRANCH IF ECH IS OK
495 050602 062716 000004    ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
496 050606 112776 000031 000000    MOVVB   #31,@(SP)      ;WRITE ERROR NUMBER IN CALL
497 050614 162716 000002    SUB      #2,(SP)        ;MOVE SP TO ERROR RETURN
498 050620 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
499 050622 162716 000010    SUB      #10,(SP)       ;MOVE SP TO NO ERROR
500 050626          275$:
501

```

```

1
2
3
4
5 050626 022737 000030 051714      CMP      #SEARCH,515$      ;WAS DATA TRANSFERRED ?
6 050634 103402                BLO      280$              ;BR IF YES
7 050636 000137 051660                JMP      355$              ;NO - EXIT
8
9
10 050642 013737 001336 001142  ;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
11 050650 001421                280$:  MOV     RMWCI,$BDDAT      ;WORD COUNT ZERO??
12 050652 032737 040000 001334      BEQ      285$              ;YES
13 050660 001015                BIT     #TRE,RMCS11        ;TRANSFER ERROR DETECTED??
14 050662 062716 000004                BNE     285$              ;YES!!
15 050666 112776 000015 000000      ADD     #4,(SP)            ;
16 050674 005037 001140                MOV     #15,@(SP)          ;ERROR NUMBER
17 050700 162716 000002                CLR     $GDDAT             ;GOOD DATA FOR TYPEOUT
18 050704 004736                SUB     #2,(SP)            ;MOVE SP TO RETURN FOR ERROR
19 050706 162716 000010                JSR     PC,@(SP)+          ;REPORT WORD COUNT NOT ZERO
20 050712 000240                SUB     #10,(SP)           ;RESTORE (SP)
21
22
23 050714 013737 001336 001140  ;REPORT ERROR IF RMBA IS NOT CORRECT
24 050722 163737 001412 001140  285$:  MOV     RMWCI,$GDDAT      ;GET WORD COUNT AT END OF TRANSFER AND
25 050730 006337 001140                SUB     RMWCO,$GDDAT      ;SUBTRACT STARTING WORD COUNT.
26 050734 063737 001414 001140      ASL     $GDDAT             ;* 2
27
28 050742 032737 000010 001344      ADD     RMBAO,$GDDAT      ;ADD STARTING BUS ADDRESS
29 050750 001403                BIT     #BAI,RMCS2I        ;WAS BUS ADDRESS INHIBIT (BAI) SET ??
30 050752 013737 001414 001140      BEQ     290$              ;NO !!
31
32 050760 023737 001140 001340  290$:  MOV     RMBAO,$GDDAT      ;ADDRESS SHOULD NOT HAVE CHANGED
33 050766 001416                CMP     $GDDAT,RMBAI      ;BUS ADDRESS OK??
34 050770 013737 001340 001142      BEQ     295$              ;YES!!
35 050776 062716 000004                MOV     RMBAI,$BDDAT      ;BAD DATA FOR TYPEOUT
36 051002 112776 000016 000000      ADD     #4,(SP)            ;
37 051010 162716 000002                MOV     #16,@(SP)          ;ERROR NUMBER
38 051014 004736                SUB     #2,(SP)            ;MOVE SP TO RETURN FOR ERROR
39 051016 162716 000010                JSR     PC,@(SP)+          ;REPORT UNEXPECTED ADDRESS
40 051022 000240                SUB     #10,(SP)           ;RESTORE (SP)
41
42
43 051024 005046                ;COMPUTE NUMBER OF SECTORS TRANSFERRED FROM WORD COUNT
44 051026 013746 001336 295$:  CLR     -(SP)              ;NUMBER OF SECTORS TRANSFERRED
45 051032 163716 001412                MOV     RMWCI,-(SP)        ;GET WORD COUNT AT END OF TRANSFER AND
46
47 051036 012746 000400                SUB     RMWCO,(SP)        ;SUBTRACT STARTING WORD COUNT.
48 051042 032737 000002 001410      MOV     #256,-(SP)        ;ASSUME 256. WORDS PER SECTOR
49 051050 001402                BIT     #BIT1,RMCS10      ;HEADER & DATA COMMAND ??
50 051052 062716 000002                BEQ     300$              ;NO !!
51
52 051056 005266 000004                ADD     #2,(SP)            ;CHANGE TO 258. WORDS PER SECTOR
53 051062 161666 000002 300$:  INC     4(SP)              ;INCREMENT SECTOR COUNT
54 051066 003373                SUB     (SP),2(SP)        ;SUBTRACT ONE SECTOR'S WORTH
55 051070 022626                BGT     300$              ;CONTINUE IF NOT DONE
56
57
;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM

```



```

58
59 051072 013737 001444 051706 ;NUMBER OF SECTORS
60 051100 013737 001416 051710 MOV RMDCO,500$ ;STORE ORIGINAL CYLINDER
61 051106 013737 001416 051712 MOV RMDAO,505$ ;STORE ORIGINAL TRACK
62 051114 013737 001332 051716 MOV RMDAO,510$ ;STORE ORIGINAL SECTOR
63 051122 000337 051716 MOV LSTRK,520$ ;STORE LAST TRACK
64 051126 005237 051716 SWAB 520$ ;GET TRACK ADDRESS TO LO BYTE AND
65 INC 520$ ;INCREMENT TO GET TOTL # OF TRACKS.
66 051132 042737 000377 051710 BIC #^C<TADMSK>,505$ ;SAVE TRACK ADDRESS BITS AND
67 051140 000337 051710 SWAB 505$ ;SWAP TRACK ADDRESS TO LOW BYTE.
68 051144 042737 177400 051712 BIC #^C<SADMSK>,510$ ;SAVE SECTOR ADDRESS BITS
69 051152 062637 051712 ADD (SP)+,510$
70
71 051156 023727 051712 000040 310$: CMP 510$,#32. ;SECTOR OVEFLOWED??
72 051164 103406 BLO 315$ ;NO!!
73 051166 005237 051710 INC 505$ ;INCREMENT TRACK
74 051172 162737 000040 051712 SUB #32.,510$ ;ADJUST SECTOR
75 051200 000766 BR 310$ ;TRY AGAIN
76
77 051202 023737 051710 051716 315$: CMP 505$,520$ ;TRACK OVERFLOWED??
78 051210 103407 BLO 320$ ;NO!!
79 051212 005237 051706 INC 500$ ;INCREMENT CYLINDER
80 051216 163737 051716 051710 SUB 520$,505$ ;ADJUST TRACK
81 051224 000766 BR 315$ ;TRY AGAIN
82 051226 000240 NOP
83
84 ;REPORT ERROR IF 'LBT' IS NOT CORRECT
85 051230 320$:
86 051230 005037 001140 CLR $GDDAT ;SET GOOD DATA FOR LBT = 0
87 051234 023727 051706 001466 CMP 500$,#822. ;SHOULD LBT BE SET??
88 051242 101407 BLOS 325$ ;NO!!
89 051244 032737 002000 001350 BIT #IAE,RMER1I ;WAS IAE SET ??
90 051252 001003 BNE 325$ ;YES - LBT SHOULD NOT BE SET
91 051254 012737 002000 001140 MOV #LBT,$GDDAT ;SET GOOD DATA FOR LBT = 1
92 051262 013737 001346 001142 325$: MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
93 051270 042737 175777 001142 BIC #^CLBT,$BDDAT
94 051276 023737 001140 001142 CMP $GDDAT,$BDDAT ;IS LBT CORRECT??
95 051304 001413 BEQ 330$ ;YES!!
96 051306 062716 000004 ADD #4,(SP)
97 051312 112776 000017 000000 MOVB #17,@(SP) ;ERROR NUMBER
98 051320 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
99 051324 004736 JSR PC,@(SP)+ ;REPORT LBT IS WRONG
100 051326 162716 000010 SUB #10,(SP) ;RESTORE (SP)
101 051332 000240 NOP
102
103 ;REPORT ERROR IF 'AOE' IS INCORRECT
104 051334 005037 001140 330$: CLR $GDDAT ;SET FOR AOE = 0
105 051340 032737 002000 001350 BIT #IAE,RMER1I ;WAS "IAE" DETECTED??
106 051346 001031 BNE 340$ ;YES-"AOE" SHOULD BE ZERO
107 051350 023727 051706 001466 CMP 500$,#822. ;SHOULD AOE BE SET??
108 051356 101425 BLOS 340$ ;NO!!
109 051360 005737 051710 TST 505$ ;MAYBE
110 051364 001012 BNE 335$ ;YES
111 051366 005737 051712 TST 510$
112 051372 001007 BNE 335$ ;YES !!
113 051374 032737 000010 051714 BIT #F2,515$ ;WAS THIS READ OR WRITE CHECK ??
114 051402 001413 BEQ 340$ ;NO !!
    
```

```

115 051404 005737 001336          TST    RMWCI          :WAS ALL DATA TRANSFERRED ??
116 051410 001410          BEQ    340$          :YES !!
117 051412 012737 001000 001140 335$: MOV    #AOE,$GDDAT   :SET FOR AOE = 1
118 051420 005037 051710          CLR    505$         :CLEAR EXPECTED TRACK
119 051424 012737 000001 051712  MOV    #1,510$      :EXPECT SECTOR = 1
120 051432 013737 001350 001142 340$: MOV    RMER1I,$BDDAT :BAD DATA FOR TYPEOUT
121 051440 042737 176777 001142  BIC    #^CAOE,$BDDAT
122 051446 023737 001140 001142  CMP    $GDDAT,$BDDAT :IS AOE CORRECTY??
123 051454 001413          BEQ    345$         :YES!!
124 051456 062716 000004          ADD    #4,(SP)
125 051462 112776 000020 000000  MOVB   #20,@(SP)    :ERROR NUMBER
126 051470 162716 000002          SUB    #2,(SP)     :MOVE SP TO RETURN FOR ERROR
127 051474 004736          JSR    PC,@(SP)+   :REPORT AOE IS WRONG
128 051476 162716 000010          SUB    #10,(SP)    :RESTORE (SP)
129 051502 000240          NOP

130
131
132 051504 032737 002000 001350  :REPORT ERROR IF RMDA IS NOT CORRECT
133 051512 001062 345$: BIT    #IAE,RMER1I  :WAS THERE AN IAE ERROR ??
134 051514 013737 051710 001140  BNE    355$         :YES - DONT CHECK RMDA,RMDC
135 051522 000337 001140          MOV    505$,$GDDAT :SETUP EXPECTED DISK ADDRESS
136 051526 113737 051712 001140  SWAB   $GDDAT
137 051534 013737 001342 001142  MOVB   510$,$GDDAT
138 051542 023737 001140 001142  MOV    RMDAI,$BDDAT :SETUP RECEIVED DISK ADDRESS
139 051550 001413          CMP    $GDDAT,$BDDAT :COMPARE EXPECTED & RECEIVED
140 051552 062716 000004          BEQ    350$         :BRANCH IF EQUAL
141 051556 112776 000021 000000  ADD    #4,(SP)
142 051564 162716 000002          MOVB   #21,@(SP)   :ERROR NUMBER
143 051570 004736          SUB    #2,(SP)     :MOVE SP TO RETURN FOR ERROR
144 051572 162716 000010  JSR    PC,@(SP)+   :REPORT BAD DISK ADDRESS
145 051576 000240          SUB    #10,(SP)    :RESTORE (SP)
146
147
148 051600 013737 051706 001140  :REPORT ERROR IF RMDC IS INCORRECT
149 051606 042737 176000 001140 350$: MOV    500$,$GDDAT  :SETUP EXPECTED CYLINDER
150 051614 013737 001370 001142  BIC    #^C1777,$GDDAT
151 051622 023737 001140 001142  MOV    RMDCI,$BDDAT :SETUP RECEIVED CYLINDER
152 051630 001413          CMP    $GDDAT,$BDDAT :COMPARE CYLINDERS
153 051632 062716 000004          BEQ    355$         :BRANCH IF EQUAL
154 051636 112776 000022 000000  ADD    #4,(SP)
155 051644 162716 000002          MOVB   #22,@(SP)   :ERROR NUMBER
156 051650 004736          SUB    #2,(SP)     :MOVE SP TO RETURN FOR ERROR
157 051652 162716 000010  JSR    PC,@(SP)+   :REPORT BAD CYLINDER
158 051656 000240          SUB    #10,(SP)    :RESTORE (SP)
159
160 051660 062716 000004          ADD    #4,(SP)     :MOVE (SP) TO ERROR CALL
161 051664 105776 000000 355$: TSTB   @(SP)       :WAS ERROR FOUND??
162 051670 001403          BEQ    360$
163 051672 062716 000004          ADD    #4,(SP)     :MOVE (SP) TO ERROR RETURN
164 051676 000402          BR     365$
165 051700 162716 000004 360$: SUB    #4,(SP)     :MOVE (SP) TO NO ERROR RETURN
166 051704 000207 365$: RTS    PC
167
168 051706 000000 500$: .WORD 0          :CYLINDER
169 051710 000000 505$: .WORD 0          :TRACK
170 051712 000000 510$: .WORD 0          :SECTOR
171 051714 000000 515$: .WORD 0          :FUNCTION CODE
    
```

172 051716 000000
173
174

520\$: .WORD 0

:# OF TRACKS FOR DEVICE UNDER TEST = LAST
;TRACK + 1 TRACK

```

1      .SBTTL  COMPOSITE ERROR CHECK SUBROUTINE
2
3      :THIS SUBROUTINE CHECKS THE STORED CONTENTS OF RMER1 AND
4      :RMER2 AFTER MASKING EACH REGISTER WORD WITH THE USER'S STATUS
5      :MASKS AND REPORTS AN ERROR IF ANY BITS ARE LEFT ON AFTER
6      :THE MASKS ARE APPLIED.
7
8
9      :CALL:
10     : (1)   JSR      PC,CMPERRSTS      MASK FOR ERROR REGISTER 1
11           .WORD   .WORD             MASK FOR ERROR REGISTER 2
12           BR      ???                RETURN HERE IF NO ERROR
13           NOP     RETURN HERE TO REPORT AN ERROR
14           ERROR   ERROR NUMBER DEFINED BY SUB
15           JSR     PC,@(SP)+          GO BACK TO SUB FOR MORE ERROR CHECKS
16           ???     RETURN HERE IF NO MORE ERRORS
17
18     :NOTE:  BITS TO BE MASKED SHOULD BE ONE; BITS TO BE TESTED SHOULD
19     :BE ZERO
20
21     051720  CMPERRSTS:
22
23     :MASK AND STORE THE CONTENTS OF RMER1 AND RMER2
24     051720  013737  001350  001176      MOV      RMER1I,$TMP1      ;STORE RMER1 AT TEMP STORAGE
25     051726  047637  000000  001176      BIC      @(SP),$TMP1      ;MASK RMER1
26     051734  062716  000002             ADD      #2,(SP)          ;MOVE SP TO NEXT MASK
27     051740  013737  001376  001200      MOV      RMER2I,$TMP2      ;STORE RMER2 AT TEMP STORAGE
28     051746  047637  000000  001200      BIC      @(SP),$TMP2      ;MASK RMER2
29
30
31     :CLEAR USER'S ERROR CALL
32     051754  062716  000006             ADD      #6,(SP)          ;MOVE SP TO USER'S ERROR CALL
33     051760  105076  000000             CLRB    @(SP)            ;CLEAR ERROR NUMBER
34     051764  162716  000004             SUB      #4,(SP)          ;LEAVE SP AT NO ERROR RETURN
35
36     :SEE IF THERE WERE ANY ERRORS IN RMER1, I.E., $TMP1
37     051770  005737  001176             TST     $TMP1            ;ANY ERRORS TO REPORT??
38     051774  001420             BEQ     5$               ;NO !!
39     051776  013737  001176  001142      MOV     $TMP1,$BDDAT     ;RECEIVED STATUS FOR TYPEOUT
40     052004  005037  001140             CLR     $GDDAT           ;EXPECTED STATUS FOR TYPEOUT
41     052010  062716  000004             ADD     #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
42     052014  112776  000066  000000      MOVVB  #66,@(SP)         ;CORRECTABLE DATA CHECK ERROR #
43     052022  162716  000002             SUB     #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
44     052026  004736             JSR     PC,@(SP)+        ;REPORT ERROR VIA USER
45     052030  162716  000010             SUB     #10,(SP)         ;MOVE SP BACK TO BRANCH
46     052034  000240             NOP
47     052036  5$:
48
49
50     :SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 ($TMP2)
51     052036  005737  001200             TST     $TMP2            ;ANY ERRORS IN RMER2?
52     052042  001420             BEQ     10$              ;NO!!
53
54     052044  013737  001200  001142      MOV     $TMP2,$BDDAT     ;RECEIVED STATUS FOR TYPEOUT
55     052052  005037  001140             CLR     $GDDAT           ;EXPECTED STATUS FOR TYPEOUT
56     052056  062716  000004             ADD     #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
57     052062  112776  000067  000000      MOVVB  #67,@(SP)         ;WRITE ERROR NUMBER IN USER'S CALL
    
```

```
58 052070 162716 000002          SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
59 052074 004736                   JSR    PC,@(SP)+    ;REPORT ERROR VIA USER
60 052076 162716 000010          SUB    #10,(SP)     ;MOVE SP TO NO ERROR RETURN
61 052102 000240                   NOP
62 052104          10$:
63
64          ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
65 052104 062716 000004          ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
66 052110 105776 000000          TSTB  @(SP)         ;WAS THERE AN ERROR CALLED??
67 052114 001403                   BEQ    20$          ;NO!!
68 052116 062716 000004          ADD    #4,(SP)      ;YES - MOVE SP TO ERROR RETURN
69 052122 000402                   BR     30$
70 052124 162716 000004          20$:  SUB    #4,(SP)     ;MOVE SP TO NO ERROR RETURN
71 052130 000207                   30$:  RTS    PC        ;RETURN TO USER
72
```

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 DEVICE SELECT SUBROUTINE
: THIS SUBROUTINE SELECTS THE DEVICE, GETTING THE DEVICE NUMBER FROM THE
: TEST QUEUE.
:CALL:
:(1) JSR PC,DEVSEL
:(2) BR ?? RETURN IF NO ERROR
:(3) NOP RETURN IF ERROR
:(4) ERROR ERROR DEFINED BY SUBROUTINE
DEVSEL:
: CLEAR USER'S ERROR CALL
ADD #4,(SP) ;MOVE SP TO USER'S ERROR
CLRB @ (SP) ;CLEAR LOW ORDER BYTE OF CALL
SJB #4,(SP) ;MOVE SP BACK
:SAVE USER'S INFORMATION AND SETUP REGISTERS
MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV #20$,ERRVEC ;SETUP FOR BUS TIMEOUT
MOV #PR6,ERRVEC+2
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 POINTS TO DEVICE NUMBER
:SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
MOVB (R1),RMCS2(R0) ;WRITE UNIT SELECT BITS
MOV RMCS1(R0),$TMP1 ;GET 'DVA' STATUS
MOV RMCS2(R0),$TMP0 ;GET 'NED' STATUS
BIT #NED,$TMP0 ;IS DEVICE NONEXISTENT ?
BEQ 10$ ;NO!!
ADD #4,10(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #111,@10(SP) ;WRITE ERROR NUMBER
BR 30$
37 052254 032737 004000 001176 10$: BIT #DVA,$TMP1 ;IS DEVICE AVAILABLE ?
38 052262 001021 BNE 35$ ;YES!!
39 052264 062766 000004 000010 ADD #4,10(SP) ;MOVE SP TO USERS ERROR CALL
40 052272 112776 000112 000010 MOVB #112,@10(SP) ;WRITE ERROR NUMBER
41 052300 000407 BR 30$
:HANDLE BUS TIMEOUT
20$: CMP (SP)+,(SP)+ ;ADJUST SP
ADD #4,10(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #113,@10(SP) ;WRITE BUS TIMEOUT ERROR NUMBER
30$: SUB #2,10(SP) ;ADJUST, RETURN TO 'NOP' PRECEDING
;THE ERROR CALL
:RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
35$: MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
MOV (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
    
```

052132
 052132 062716 000004
 052136 105076 000000
 052142 162716 000004
 052146 013746 000004
 052152 013746 000006
 052156 010046
 052160 010146
 052162 012737 052302 000004
 052170 012737 000300 000006
 052176 013700 001276
 052202 013701 001464
 052206 111160 000010
 052212 016037 000000 001176
 052220 016037 000010 001174
 052226 032737 010000 001174
 052234 001407
 052236 062766 000004 000010
 052244 112776 000111 000010
 052252 000422
 052254 032737 004000 001176
 052262 001021
 052264 062766 000004 000010
 052272 112776 000112 000010
 052300 000407
 052302 022626
 052304 062766 000004 000010
 052312 112776 000113 000010
 052320 162766 000002 000010
 052326 012601
 052330 012600
 052332 012637 000006

52 052336 012637 000004
53 052342 000207

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

```

1      .SBTTL  SEEK STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE VERIFIES THE RESULTS OF SEEK TESTS USING STATUS
4      ;STORED IN THE GET BUFFER AND TEST PARAMETERS STORED IN THE PUT BUFFER.
5
6
7      ;
8      ;THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS DETECTED
9      ;AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN THE 'ERROR' TRAP
10     ;OF THE CALLING ROUTINE.  SEEK STATUS IS CHECKED AS FOLLOWS:
11
12     ;CALL:
13     ;(1)  JSR    PC,SEKSTS
14           BR    ???          RETURN HERE IF NO ERROR
15           NOP
16           ERROR          RETURN HERE TO REPORT AN ERROR
17           JSR    PC,@(SP)+   ERROR NUMBER DEFINED BY SUB
18           ???          GO BACK TO SUB FOR MORE ERROR CHECKS
19           RETURN HERE IF NO MORE ERRORS
20
21     SEKSTS:
22
23     ;CLEAR USERS' ERROR CALL
24     NOP
25     ADD    #4,(SP)          ;MOVE (SP) TO ERROR CALL
26     CLRB  @ (SP)          ;CLEAR ERROR NUMBER
27     SUB    #4,(SP)          ;MOVE (SP) TO NO ERROR RETURN
28     CLR   300$           ;CLEAR ERROR FLAGS
29
30     ;TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
31     ;LOCAL REGISTERS, I.E., 'PAR' = 1 AND 'DPE' = 0
32     BIT   #PAR,RMER1I     ;WAS PARITY ERROR DETECTED??
33     BEQ   1$             ;NO!!
34     BIT   #DPE,RMER2I     ;WAS IT DUE TO CONTROL BUS??
35     BNE   1$             ;NOT SURE!!
36
37     ;REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
38     CLR   $GDDAT         ;EXPECTED STATUS
39     MOV   RMER1I,$BDDAT  ;RECEIVED STATUS
40     ADD   #4,(SP)        ;MOVE STACK TO USER'S ERROR
41     MOVB  #50,@(SP)      ;ERROR #50
42     SUB   #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
43     JSR   PC,@(SP)+
44     SUB   #10,(SP)       ;RESTORE STACK
45     BR    3$            ;IAE SHOULD BE ZERO
46
47     ;DETERMINE THE VALUE OF 'IAE' STATUS BASED ON TRACK, SECTOR AND CYLINDER
48     ;ALSO, SET 'SKI' IF CYLINDER ADDRESS IS TOO LARGE.
49     1$: MOV   #IAE,$GDDAT ;SETUP FOR IAE = 1
50         BIS   #SKI,300$   ;SETUP FOR SKI = 1
51         CMP   RMDCO,#822. ;GREATER THAN LAST CYLINDER ?
52         BHI   3$          ;YES - CYLINDER IS INVALID
53         BIC   #SKI,300$   ;CLEAR SKI ERROR FLAG
54
55     CMPB  RMDAO+1,LSTRK+1 ;GREATER THAN LAST TRACK ?
56     BHI   3$            ;YES - TRACK IS INVALID
57
58     CMPB  RMDAO,#29.     ;SECTOR > 29. ?
59     BLOS  2$            ;BR IF NO

```



```

58 052520 032737 010000 001442      BIT      #FMT16,RMOFO      ;18 BIT FORMAT ?
59 052526 001406                    BEQ      3$              ;YES - SECTOR IS INVALID FOR 18 BIT MODE
60 052530 123727 001416 000037      CMPB    RMDAO,#31.      ;SECTOR > 31. ?
61 052536 101002                    BHI      3$              ;YES - SECTOR IS INVALID
62
63 052540 005037 001140      2$:    CLR      $GDDAT      ;'IAE' SHOULD = 0
64
65      ;COMPARE EXPECTED AND RECIEVED 'IAE' STATUS
66 052544 013737 001350 001142      3$:    MOV      RMER1I,$BDDAT ;IS IAE OK??
67 052552 042737 175777 001142      BIC     #^CIAE,$BDDAT   ;SAVE IAE BIT FOR COMPARE
68 052560 023737 001140 001142      CMP     $GDDAT,$BDDAT   ;CORRECT 'IAE' STATUS ?
69 052566 001004                    BNE     35$             ;BR IF NO
70 052570 042737 040000 053602      BIC     #SKI,300$       ;CLEAR SKI FLAG
71 052576 000413                    BR      5$              ;GO CHECK NEXT ERROR
72 052600
73      35$:
74 052600 062716 000004                    ;REPORT INCORRECT 'IAE' STATUS VIA USER'S ERROR CALL
75 052604 112776 000051 000000      ADD     #4,(SP)
76 052612 162716 000002                    MOV     #51,@(SP)       ;ERROR 51
77 052616 004736                    SUB     #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
78 052620 162716 000010                    JSR    PC,@(SP)+       ;REPORT INCORRECT IAE
79 052624 000240                    SUB     #10,(SP)        ;RESTORE (SP)
80 052626
81
82      5$:
83      ;REPORT ANY IVC ERROR AS
84      ; IVC ERROR WITH VOLUME VALID ZERO
85 052626 032737 010000 001376      BIT     #IVC,RMER2I     ;IVC ERROR??
86 052634 001427                    BEQ     52$             ;NO!!
87 052636 005037 001140      CLR     $GDDAT         ;EXPECTED STATUS
88 052642 013737 001376 001142      MOV     RMER2I,$BDDAT  ;RECEIVED STATUS
89 052650 062716 000004                    ADD     #4,(SP)         ;MOVE SP TO USER'S ERROR
90 052654 112776 000060 000000      MOV     #60,@(SP)      ;ERROR 60 IF VV = 0
91 052662 032737 000100 001346      BIT     #VV,RMDSI
92 052670 001403                    BEQ     51$             ;ERROR 61 IF VV = 1
93 052672 112776 000061 000000      MOV     #61,@(SP)
94 052700 162716 000002      51$:    SUB     #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
95 052704 004736                    JSR    PC,@(SP)+       ;REPORT ERROR VIA USER
96 052706 162716 000010                    SUB     #10,(SP)        ;RESTORE SP
97 052712 000240      NOP
98
99 052714 013737 001376 001142      52$:    MOV     RMER2I,$BDDAT ;RECEIVED STATUS
100 052722 042737 137777 001142      BIC     #^CSKI,$BDDAT  ;CLEAR DONT CARES
101 052730 013737 053602 001140      MOV     300$,$GDDAT    ;GET EXPECTED SKI STATUS
102 052736 042737 137777 001140      BIC     #^CSKI,$GDDAT  ;CLEAR DONT CARES
103 052744 001417                    BEQ     53$             ;BRANCH IF 0 EXPECTED
104
105      ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
106 052746 032737 040000 001142      BIT     #SKI,$BDDAT    ;WAS SKI DETECTED ??
107 052754 001032                    BNE     54$             ;YES !!
108 052756 062716 000004                    ADD     #4,(SP)         ;MOVE SP TO USERS ERROR CALL
109 052762 112776 000267 000000      MOV     #267,@(SP)     ;WRITE ERROR NUMBER
110 052770 162716 000002                    SUB     #2,(SP)         ;MOVE SP TO ERROR RETURN
111 052774 004736                    JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
112 052776 162716 000010                    SUB     #10,(SP)        ;MOVE SP TO NO ERROR
113 053002 000443                    BR      6$              ;GO TO NEXT ERROR CHECK
114 053004
115      53$:

```

```

115
116 ;REPORT ERROR IF SKI IS SET
117 053004 032737 040000 001142 BIT #SKI,$BDDAT ;IS SKI SET ??
118 053012 001413 BEQ 54$ ;NO - SKI IS OK
119 053014 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
120 053020 112776 000054 000000 MOVB #54,@(SP) ;LOAD ERROR NUMBER
121 053026 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
122 053032 004736 JSR PC,@(SP)+ ;REPORT SEEK ERROR
123 053034 162716 000010 SUB #10,(SP) ;RESTORE (SP)
124 053040 000240 NOP
125
126 ;REPORT ANY DEVICE CHECK
127 053042 032737 000200 001376 54$: BIT #DVC,$RMER2I ;WAS THERE DVC DURING SEEK??
128 053050 001420 BEQ 6$ ;NO!!
129 053052 005037 001140 CLR $GDDAT ;EXPECTED STATUS
130 053056 013737 001376 001142 MOV $RMER2I,$BDDAT ;RECEIVED STATUS
131 053064 062716 000004 ADD #4,(SP)
132 053070 112776 000055 000000 MOVB #55,@(SP) ;ERROR #55
133 053076 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
134 053102 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
135 053104 162716 000010 SUB #10,(SP) ;RESTORE SP
136 053110 000240 NOP
137
138 ;REPORT ANY 'OPI' ERROR AS OPI WITH MOL = 0, OR OPI
139 ;BECAUSE ON CYLINDER LATCH DIDN'T RESET
140 053112 032737 020000 001350 6$: BIT #OPI,$RMER1I ;'OPI' ERROR??
141 053120 001427 BEQ 8$ ;NO!!
142 053122 005037 001140 CLR $GDDAT ;EXPECTED STATUS
143 053126 013737 001350 001142 MOV $RMER1I,$BDDAT ;RECEIVED STATUS
144 053134 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
145 053140 112776 000052 000000 MOVB #52,@(SP) ;LOAD ERROR NUMBER
146 053146 032737 010000 001346 BIT #MOL,$RMDSI ;WAS MEDIUM ON LINE??
147 053154 001403 BEQ 7$ ;NO!!
148 053156 112776 000053 000000 MOVB #53,@(SP) ;YES - CHANGE ERROR NUMBER
149 053164 162716 000002 7$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
150 053170 004736 JSR PC,@(SP)+ ;REPORT 'OPI' ERROR
151 053172 162716 000010 SUB #10,(SP) ;RESTORE (SP)
152 053176 000240 NOP
153
154 ;SEE IF 'PIP' = 0, AND 'ATA', 'MOL' AND 'VV' = 1
155 053200 013746 001346 8$: MOV $RMDSI,-(SP)
156 053204 042716 047677 BIC #^C<ATA!PIP!MOL!VV>,(SP)
157 053210 022726 110100 CMP #ATA!MOL!VV,(SP)+
158 053214 001002 BNE 9$ ;ERROR IN RMDS
159 053216 000137 053552 JMP 14$ ;RMDS IS OK
160
161 ;REPORT ERROR IF MOL = 0 AND OPI = 0
162 053222 032737 010000 001346 9$: BIT #MOL,$RMDSI ;IS MOL RESET??
163 053230 001030 BNE 10$ ;NO - MOL IS SET
164 053232 032737 020000 001350 BIT #OPI,$RMER1I ;WAS OPI SET
165 053240 001024 BNE 10$ ;YES - DONT REPORT ERROR
166 053242 013737 001346 001140 MOV $RMDSI,$GDDAT ;EXPECTED STATUS
167 053250 052737 010000 001140 BIS #MOL,$GDDAT
168 053256 013737 001346 001142 MOV $RMDSI,$BDDAT ;RECEIVED STATUS
169 053264 062716 000004 ADD #4,(SP)
170 053270 112776 000062 000000 MOVB #62,@(SP)
171 053276 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR

```

```

172 053302 004736          JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
173 053304 162716 000010  SUB      #10,(SP)
174 053310 000240          NOP
175
176          ;REPORT AN ERROR IF 'PIP' IS STIL SET AND SKI NOT SET
177 053312 032737 020000 001346 10$: BIT      #PIP,RMDSI      ;IS 'PIP' STILL SET??
178 053320 001430          BEQ      11$            ;NO!!
179 053322 032737 040000 001376  BIT      #SKI,RMER2I     ;WAS 'SKI' SET??
180 053330 001024          BNE      11$            ;YES-DONT REPORT PIP
181 053332 013737 001346 001140  MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
182 053340 042737 020000 001142  BIC      #PIP,$BDDAT
183 053346 013737 001346 001142  MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
184 053354 062716 000004          ADD      #4,(SP)         ;MOVE (SP) TO ERROR
185 053360 112776 000056 000000  MOVVB   #56,@(SP)        ;LOAD ERROR NUMBER
186 053366 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
187 053372 004736          JSR      PC,@(SP)+      ;REPORT 'PIP' SET AFTER SEEK
188 053374 162716 000010  SUB      #10,(SP)        ;RESTORE (SP)
189 053400 000240          NOP
190
191          ;REPORT AN ERROR IF 'ATA' IS NOT SET
192 053402 032737 100000 001346 11$: BIT      #ATA,RMDSI      ;WAS 'ATA' SET ??
193 053410 001024          BNE      13$            ;YES!!
194 053412 013737 001346 001140  MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
195 053420 052737 110600 001140  BIS      #ATA!MOL!DPR!DRY,$GDDAT
196 053426 013737 001346 001142  MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
197 053434 062716 000004          ADD      #4,(SP)         ;MOVE (SP) TO ERROR
198 053440 112776 000057 000000  MOVVB   #57,@(SP)        ;LOAD ERROR NUMBER
199 053446 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
200 053452 004736          JSR      PC,@(SP)+      ;REPORT ATTENTION NOT SET DURING
201          ;SEEK TEST
202 053454 162716 000010  SUB      #10,(SP)        ;RESTORE (SP)
203 053460 000240          NOP
204
205          ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
206 053462 032737 000100 001346 13$: BIT      #VV,RMDSI      ;IS VV = 0 ??
207 053470 001030          BNE      14$            ;NO!!
208 053472 032737 010000 001376  BIT      #IVC,RMER2I     ;IS IVC ALSO 0 ??
209 053500 001024          BNE      14$            ;NO - IVC IS SET
210 053502 013737 001346 001140  MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
211 053510 052737 000100 001140  BIS      #VV,$GDDAT
212 053516 013737 001346 001142  MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
213 053524 062716 000004          ADD      #4,(SP)
214 053530 112776 000064 000000  MOVVB   #64,@(SP)        ;ERROR #64
215 053536 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
216 053542 004736          JSR      PC,@(SP)+
217 053544 162716 000010  SUB      #10,(SP)
218 053550 000240          NOP
219 053552          14$:
220
221          ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
222 053552 000240          NOP
223 053554 062716 000004          ADD      #4,(SP)         ;MOVE (SP) TO ERROR CALL
224 053560 105776 000000  TSTB    @(SP)            ;WAS ERROR CALLED??
225 053564 001403          BEQ      15$            ;NO!!
226 053566 062716 000004          ADD      #4,(SP)         ;MOVE TO ERROR RETURN
227 053572 000402          BR      16$
228

```

CZRMOAO RM05/3/2 FCTNL TST 3
SEEK STATUS CHECK SUBROUTINE

MACRO V03.01 11-APR-80 13:39:53 PAGE 29-4

H 2

SEQ 0226

229	053574	162716	000004	15\$:	SUB	#4,(SP)	:MOVE (SP) TO NO ERROR RETURN
230	053600	000207		16\$:	RTS	PC	:RETURN
231							
232	053602	000000		300\$:	.WORD	0	:ERROR FLAGS
233							

```

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:
13     053604 010046          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
14     053606 010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
15     053610 013746 000004   MOV      ERRVEC,-(SP)     ;;PUSH ERRVEC ON STACK
16     053614 013746 000006   MOV      ERRVEC+2,-(SP)  ;;PUSH ERRVEC+2 ON STACK
17     13 053620 012737 053660 000004   MOV      #10$,ERRVEC     ;SETUP FOR BUS TIMEOUT
18     14 053626 012737 000300 000006   MOV      #PR6,ERRVEC+2
19     15 053634 013700 001276          MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
20     16 053640 012760 000040 000010   MOV      #CLR,RMCS2(R0) ;CLEAR MASSBUS
21     17 053646 013701 001464          MOV      TSTQUE,R1      ;GET DEVICE UNDER TEST
22     18 053652 111160 000010          MOV      (R1),RMCS2(R0) ;SELECT DEVICE
23     19 053656 000412          BR      20$
24
25     21 053660 022626          10$:  CMP      (SP)+,(SP)+    ;ADJUST STACK
26     22 053662 062766 000004 000010   ADD      #4,10(SP)      ;MOVE SP TO USER'S ERROR CALL
27     23 053670 112776 000007 000010   MOV      #7,@10(SP)    ;WRITE THE ERROR NUMBER
28     24 053676 162766 000002 000010   SUB      #2,10(SP)     ;ADJUST SP TO RETURN TO ERROR
29     25 053704          20$:  MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
30     26 053704 012637 000006          MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
31     27 053710 012637 000004          MOV      (SP)+,R1      ;;POP STACK INTO R1
32     28 053714 012601          MOV      (SP)+,R0      ;;POP STACK INTO R0
33     29 053716 012600          MOV      (SP)+,R0
34     30 053720 000207          RTS      PC
35

```

```

1      .SBTTL  CONTROLLER CLEAR STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE VERIFIES THAT THE SUBSYSTEM IS INITIALIZED BASED ON
4      ;STATUS STORED IN THE GET BUFFER.  THIS SUBROUTINE SHOULD ONLY BE
5      ;USED FOLLOWING A CONTROLLER CLEAR OPERATION, I.E., WRITING A 1 IN BIT
6      ;5 OF RMCS2, BECAUSE THE ERROR MESSAGES ARE BASED ON THAT CONDITION.
7
8      ;STATUS PERTINENT TO THE DEVICE IS NOT CHECKED.  IN PARTICULAR, THE
9      ;FOLLOWING STATUS BITS ARE NOT CHECKED:
10     :
11     :       ATA,ERR,PIP,MOL,WRL,LBT,PGM,VV,OM,UNS,SKI,DVC
12     :
13     :
14     :CALL:
15     : (1)  JSR    PC,CLRSTS
16     :      BR    ???
17     :      NOP
18     :      ERROR
19     :      JSR    PC,@(SP)+
20     :      ???
21
22     CLRSTS:
23
24     ;CLEAR USER'S ERROR CALL
25     053722 062716 000004      ADD    #4,(SP)      ;MOVE SP TO ERROR
26     053726 105076 000000      CLR    @ (SP)      ;CLEAR ERROR NUMBER
27     053732 162716 000004      SUB    #4,(SP)      ;MOVE SP BACK TO NO ERROR
28
29     ;REPORT ERROR IF RMCS1 NOT INITIALIZED
30     053736 013737 001334 001142 4$:  MOV    RMCS1I,$BDDAT ;VERIFY RMCS1
31     053744 042737 100000 001142      BIC    #SC,$BDDAT   ;IGNORE SPECIAL CONDITION
32     053752 012737 004200 001140      MOV    #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
33     053760 023737 001140 001142      CMP    $GDDAT,$BDDAT ;COMPARE EXPECTED, RECEIVED
34     053766 001413                    BEQ    5$           ;BRANCH IF EQUAL
35     053770 062716 000004      ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
36     053774 112776 000126 000000      MOV    #126,@(SP)   ;WRITE ERROR NUMBER IN CALL
37     054002 162716 000002      SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
38     054006 004736                    JSR    PC,@(SP)+    ;REPORT ERROR VIA USER
39     054010 162716 000010      SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR
40     054014 000240                    NOP
41
42     ;REPORT ERROR IF RMBA NOT RESET
43     054016 005037 001140 001142 5$:  CLR    $GDDAT      ;VERIFY RMBA IS ZERO
44     054022 013737 001340      MOV    RMBAI,$BDDAT
45     054030 001413                    BEQ    7$           ;BRANCH IF ZERO
46     054032 062716 000004      ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
47     054036 112776 000127 000000      MOV    #127,@(SP)   ;WRITE ERROR NUMBER IN CALL
48     054044 162716 000002      SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
49     054050 004736                    JSR    PC,@(SP)+    ;REPORT ERROR VIA USER
50     054052 162716 000010      SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR
51     054056 000240                    NOP
52
53     ;REPORT ERROR IF RMCS2 NOT INITIALIZED
54     054060 013737 001344 001142 7$:  MOV    RMCS2I,$BDDAT ;VERIFY RMCS2
55     054066 010146      MOV    R1,-(SP)     ;:PUSH R1 ON STACK
56     054070 005046      CLR    -(SP)        ;:EXPECT IR & UNIT NUMBER
57     054072 013701 001464      MOV    TSTQUE,R1    ;R1 = ADDRESS OF TEST QUE
58     054076 111116      MOV    (R1),(SP)
59     054100 052716 000100      BIS    #IR,(SP)
60     054104 012637 001140      MOV    (SP)+,$GDDAT
    
```

```

58 054110 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
59 054112 023737 001140 001142  CMP      $GDDAT,$BDDAT  ;;COMPARE EXPECTED & RECEIVED
60 054120 001413          BEQ      9$           ;;BRANCH IF EQUAL
61 054122 062716 000004          ADD      #4,(SP)       ;;MOVE SP TO USER'S ERROR CALL
62 054126 112776 000130 000000  MOVB    #130,a(SP)     ;;WRITE ERROR NUMBER IN CALL
63 054134 162716 000002          SUB      #2,(SP)       ;;MOVE SP TO RETURN FOR ERROR
64 054140 004736          JSR     PC,a(SP)+     ;;REPORT ERROR VIA USER
65 054142 162716 000010          SUB      #10,(SP)      ;;MOVE SP BACK TO NO ERROR
66 054146 000240          NOP
67                                ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
68 054150 005037 001140          9$:    CLR      $GDDAT      ;;VERIFY RMER1
69 054154 013737 001350 001142  MOV      RMER1I,$BDDAT
70 054162 042737 040000 001142  BIC     #UNS,$BDDAT    ;;IGNORE UNSAFE
71 054170 001413          BEQ     13$          ;;BRANCH IF ZERO
72 054172 062716 000004          ADD     #4,(SP)      ;;MOVE SP TO USER'S ERROR CALL
73 054176 112776 000131 000000  MOVB    #131,a(SP)     ;;WRITE ERROR NUMBER IN CALL
74 054204 162716 000002          SUB     #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
75 054210 004736          JSR     PC,a(SP)+     ;;REPORT ERROR VIA USER
76 054212 162716 000010          SUB     #10,(SP)     ;;MOVE SP BACK TO NO ERROR
77 054216 000240          NOP
78                                ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
79 054220 013737 001360 001142  13$:   MOV      RMMR1I,$BDDAT  ;;VERIFY RMMR
80 054226 042737 000046 001142  BIC     #WC!LS!LST,$BDDAT ;;IGNORE WORD CLOCK, SCT, TRK
81 054234 012737 000010 001140  MOV      #MWD,$GDDAT    ;;EXPECT WRITE DATA BIT
82 054242 023737 001140 001142  CMP     $GDDAT,$BDDAT  ;;COMPARE EXPECTED AND RECEIVED
83 054250 001413          BEQ     17$          ;;BRANCH IF 0
84 054252 062716 000004          ADD     #4,(SP)      ;;MOVE SP TO USER'S ERROR CALL
85 054256 112776 000133 000000  MOVB    #133,a(SP)     ;;WRITE ERROR NUMBER IN CALL
86 054264 162716 000002          SUB     #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
87 054270 004736          JSR     PC,a(SP)+     ;;REPORT ERROR VIA USER
88 054272 162716 000010          SUB     #10,(SP)     ;;MOVE SP BACK TO NO ERROR
89 054276 000240          NOP
90                                ;REPORT AN ERROR IF RMEC2 IS NOT RESET
91 054300 005037 001140          17$:   CLR      $GDDAT      ;;EXPECT ZEROS
92 054304 013737 001402 001142  MOV      RMEC2I,$BDDAT  ;;VERIFY RMEC2=0
93 054312 001413          BEQ     19$          ;;BRANCH IF 0
94 054314 062716 000004          ADD     #4,(SP)      ;;MOVE SP TO USER'S ERROR CALL
95 054320 112776 000135 000000  MOVB    #135,a(SP)     ;;WRITE ERROR NUMBER IN CALL
96 054326 162716 000002          SUB     #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
97 054332 004736          JSR     PC,a(SP)+     ;;REPORT ERROR VIA USER
98 054334 162716 000010          SUB     #10,(SP)     ;;MOVE SP BACK TO NO ERROR
99 054340 000240          NOP
100                                ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
101 054342 013737 001374 001142  19$:   MOV      RMMR2I,$BDDAT  ;;VERIFY RMMR2
102 054350 042737 140000 001142  BIC     #RQA!RQB,$BDDAT
103 054356 012737 011777 001140  MOV      #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
104 054364 023737 001140 001142  CMP     $GDDAT,$BDDAT
105 054372 001413          BEQ     21$          ;;BRANCH IF 0
106 054374 062716 000004          ADD     #4,(SP)      ;;MOVE SP TO USER'S ERROR CALL
107 054400 112776 000136 000000  MOVB    #136,a(SP)     ;;WRITE ERROR NUMBER IN CALL
108 054406 162716 000002          SUB     #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
109 054412 004736          JSR     PC,a(SP)+     ;;REPORT ERROR VIA USER
110 054414 162716 000010          SUB     #10,(SP)     ;;MOVE SP BACK TO NO ERROR
111 054420 000240          NOP
112                                ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC
113 054422 005037 001140          21$:   CLR      $GDDAT      ;;EXPECT ALL ZEROS
114 054426 013737 001376 001142  MOV      RMER2I,$BDDAT  ;;VERIFY RMER2
    
```

```

115 054434 042737 040200 001142      BIC      #SKI!DVC,$BDDAT ;IGNORE DEVICE ERRORS
116 054442 001413                    BEQ      215$           ;BRANCH IF OTHER BITS 0
117 054444 062716 000004                    ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
118 054450 112776 000174 000000      MOVVB   #174,@(SP)    ;WRITE ERROR NUMBER IN CALL
119 054456 162716 000002                    SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
120 054462 004736                    JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
121 054464 162716 000010      SUB     #10,(SP)      ;MOVE SP BACK TO NO ERROR
122 054470 000240      NOP
123                                ;REPORT ERROR IF RMD5 NOT INITIALIZED
124 054472 013737 001346 001142      215$:  MOV     RMD5I,$BDDAT ;TEST DRIVE STATUS REGISTER
125 054500 042737 177177 001142      BIC     #^C<DRY!DPR>,$BDDAT
126 054506 012737 000600 001140      MOV     #DPR!DRY,$GDDAT ;EXPECTED DRIVE STATUS
127 054514 023737 001140 001142      CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
128 054522 001413                    BEQ     22$           ;BRANCH IF EQUAL
129 054524 062716 000004                    ADD     #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
130 054530 112776 000134 000000      MOVVB  #134,@(SP)    ;WRITE ERROR NUMBER
131 054536 162716 000002                    SUB     #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
132 054542 004736                    JSR     PC,@(SP)+    ;REPORT ERROR TO USER
133 054544 162716 000010      SUB     #10,(SP)      ;MOVE SP BACK TO NO ERROR
134 054550 000240      NOP
135 054552 062716 000004      22$:  ADD     #4,(SP)       ;MOVE SP TO ERROR CALL
136 054556 105776 000000      TSTB   @(SP)         ;WAS AN ERROE DETECTED??
137 054562 001403                    BEQ     23$           ;NO!!
138 054564 062716 000004      ADD     #4,(SP)       ;YES - MOVE TO ERROR RETURN
139 054570 000402      BR     24$           ;
140 054572 162716 000004      23$:  SUB     #4,(SP)      ;MOVE SP TO NO ERROR RETURN
141 054576 000240      24$:  NOP
142 054600 000207      RTS     PC
143
    
```



```

1      .SBTTL  PACK ACKNOWLEDGE STATUS CHECK
2
3      :THIS SUBROUTINE CHECKS THE RESULTS OF A PACK ACKNOWLEDGE
4      :COMMAND USING THE STATUS STORED IN THE GET BUFFER.  ERRORS ARE
5      :REPORTED TO THE USER VIA THE USER'S ERROR CALL.
6
7      :CALL:
8      :(1)  JSR    PC,ACKSTS
9           BR     ???          RETURN HERE IF NO ERROR
10          NOP          RETURN HERE TO REPORT AN ERROR
11          ERROR        ERROR NUMBER DEFINED BY SUB
12          JSR    PC,@(SP)+    GO BACK TO SUB FOR MORE ERROR CHECKS
13          ???          RETURN HERE IF NO MORE ERRORS
14
15 054602  ACKSTS:
16
17      :CLEAR USER'S ERROR CALL
18 054602 062716 000004      ADD    #4,(SP)          :MOVE SP TO ERROR CALL
19 054606 105076 000000      CLRB  @ (SP)          :CLEAR LOW ORDER BYTE
20 054612 162716 000004      SUB    #4,(SP)          :MOVE SP BACK
21
22      :REPORT AN ERROR IF 'VV' IS 0
23 054616 032737 000100 001346  BIT    #VV,RMSI        :IS VOLUME VALID SET??
24 054624 001024          BNE    1$             :YES!!
25 054626 013737 001346 001140  MOV    RMSI,$GDDAT     :EXPECTED STATUS
26 054634 052737 000100 001140  BIS    #VV,$GDDAT
27 054642 013737 001346 001142  MOV    RMSI,$BDDAT     :RECEIVED STATUS
28 054650 062716 000004          ADD    #4,(SP)        :MOVE SP TO ERROR CALL
29 054654 112776 000155 000000  MOVB  #155,@(SP)      :WRITE NUMBER IN ERROR CALL
30 054662 162716 000002          SUB    #2,(SP)        :MOVE SP TO RETURN FOR ERROR
31 054666 004736          JSR    PC,@(SP)+     :REPORT THE ERROR
32 054670 162716 000010          SUB    #10,(SP)       :MOVE SP BACK TO BRANCH
33 054674 000240          NOP
34 054676      1$:
35
36      :REPORT AN ERROR IF 'MOL' IS 0
37 054676 032737 010000 001346  BIT    #MOL,RMSI      :IS MOL SET??
38 054704 001024          BNE    2$             :YES!!
39 054706 013737 001346 001140  MOV    RMSI,$GDDAT     :EXPECTED STATUS
40 054714 052737 010000 001140  BIS    #MOL,$GDDAT
41 054722 013737 001346 001142  MOV    RMSI,$BDDAT     :RECEIVED STATUS
42 054730 062716 000004          ADD    #4,(SP)        :MOVE SP TO ERROR CALL
43 054734 112776 000041 000000  MOVB  #41,@(SP)       :WRITE NUMBER OF ERROR IN CALL
44 054742 162716 000002          SUB    #2,(SP)        :MOVE SP TO RETURN FOR ERROR
45 054746 004736          JSR    PC,@(SP)+     :REPORT TH ERROR
46 054750 162716 000010          SUB    #10,(SP)       :MOVE SP TO BRANCH
47 054754 000240          NOP
48 054756      2$:
49
50      :SEE IF 'UNS','OPI','RMR','ILR', OR 'ILF' IS SET
51 054756 032737 060007 001350  BIT    #UNS!OPI!RMR!ILR!ILF,RMER1I
52 054764 001570          BEQ    7$
53
54      :REPORT AN ERROR IF 'UNS' IS SET
55 054766 032737 040000 001350  BIT    #UNS,RMER1I     :WAS UNS SET??
56 054774 001424          BEQ    3$             :NO!!
57 054776 013737 001350 001142  MOV    RMER1I,$BDDAT   :RECEIVED STATUS
  
```

58	055004	013737	001350	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
59	055012	042737	040000	001140	BIC	#UNS,\$GDDAT	
60	055020	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
61	055024	112776	000042	000000	MOVB	#42,@(SP)	:WRITE NUMBER OF ERROR IN CALL
62	055032	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
63	055036	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
64	055040	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
65	055044	000240			NOP		
66	055046					3\$:	
67							
68						:REPORT ANY OPI ERROR	
69	055046	032737	020000	001350	BIT	#OPI,RMER1I	:WAS OPI SET??
70	055054	001424			BEQ	4\$:NO!!
71	055056	013737	001350	001142	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
72	055064	013737	001350	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
73	055072	042737	020000	001140	BIC	#OPI,\$GDDAT	
74	055100	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
75	055104	112776	000043	000000	MOVB	#43,@(SP)	:WRITE NUMBER OF ERROR IN CALL
76	055112	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
77	055116	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
78	055120	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
79	055124	000240			NOP		
80	055126					4\$:	
81							
82						:REPORT ANY RMR ERROR	
83	055126	032737	000004	001350	BIT	#RMR,RMER1I	:WAS RMR SET??
84	055134	001424			BEQ	5\$:NO!!
85	055136	013737	001350	001142	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
86	055144	013737	001350	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
87	055152	042737	000004	001140	BIC	#RMR,\$GDDAT	
88	055160	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
89	055164	112776	000044	000000	MOVB	#44,@(SP)	:WRITE NUMBER OF ERROR IN CALL
90	055172	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
91	055176	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
92	055200	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
93	055204	000240			NOP		
94	055206					5\$:	
95							
96						:REPORT ANY ILR ERROR	
97	055206	032737	000002	001350	BIT	#ILR,RMER1I	:WAS ILR SET??
98	055214	001424			BEQ	6\$:NO!!
99	055216	013737	001350	001142	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
100	055224	013737	001350	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS
101	055232	042737	000002	001140	BIC	#ILR,\$GDDAT	
102	055240	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
103	055244	112776	000045	000000	MOVB	#45,@(SP)	:WRITE NUMBER OF ERROR IN CALL
104	055252	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
105	055256	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
106	055260	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
107	055264	000240			NOP		
108	055266					6\$:	
109							
110						:REPORT ANY ILF ERROR	
111	055266	032737	000001	001350	BIT	#ILF,RMER1I	:WAS ILF SET??
112	055274	001424			BEQ	7\$:NO!!
113	055276	013737	001350	001142	MOV	RMER1I,\$BDDAT	:RECEIVED STATUS
114	055304	013737	001350	001140	MOV	RMER1I,\$GDDAT	:EXPECTED STATUS

115	055312	042737	000001	001140	BIC	#ILF,\$GDDAT	
116	055320	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
117	055324	112776	000046	000000	MOVB	#46,@(SP)	:WRITE NUMBER OF ERROR IN CALL
118	055332	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
119	055336	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
120	055340	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
121	055344	000240			NOP		
122	055346			7\$:			
123							
124					:AUGMENT RETURN ADDRESS IF ERROR WAS FOUND		
125	055346	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
126	055352	105776	000000		TSTB	@(SP)	:WAS ERROR FOUND??
127	055356	001403			BEQ	8\$:NO!!
128	055360	062716	000004		ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
129	055364	000402			BR	9\$	
130	055366	162716	000004	8\$:	SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
131	055372	000240		9\$:	NOP		
132	055374	000207			RTS	PC	
133							

```

1      .SBTTL RECALIBRATE STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE CHECKS THE RESULTS OF A RECALIBRATE OPERATION
4      ;USING THE STATUS STORED IN THE GET BUFFER.
5
6      ;CALL:
7
8      ;(1) JSR PC,RCLSTS ;CALL SUBROUTINE
9          BR   ???      RETURN HERE IF NO ERROR
10         NOP          RETURN HERE TO REPORT AN ERROR
11         ERROR       ERROR NUMBER DEFINED BY SUB
12         JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
13         ???        RETURN HERE IF NO MORE ERRORS
14
15 055376 RCLSTS:
16
17 ;CLEAR USER'S ERROR NUMBER
18 055376 062716 000004 ADD #4,(SP)
19 055402 105076 000000 CLRB @(SP) ;CLEAR USER'S ERROR CALL
20 055406 162716 000004 SUB #4,(SP) ;MOVE SP BACK TO BRANCH
21
22
23 ;SEE IF 'PAR' OR 'ILF' OR 'OPI' OR 'IAE' IS SET
24 055412 032737 022011 001350 BIT #OPI!PAR!ILF!IAE,RMER1I
25 055420 001553 BEQ 4$ ;NONE ARE SET - GO TO NEXT CHECK
26
27 ;REPORT ANY MASSBUS CONTROL BUS PARITY ERROR, I.E.,
28 ;'PAR' = 1 AND 'DPE' = 0
29 055422 032737 000010 001350 BIT #PAR,RMER1I ;WAS 'PAR' SET??
30 055430 001430 BEQ 1$ ;NO!!
31 055432 032737 000010 001376 BIT #DPE,RMER2I ;WAS 'DPE' SET??
32 055440 001024 BNE 1$ ;YES - NOT A REGISTER ERROR
33 055442 013737 001350 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
34 055450 042737 000010 001140 BIC #PAR,$GDDAT
35 055456 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
36 055464 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
37 055470 112776 000050 000000 MOVB #50,@(SP) ;WRITE ERROR NUMBER IN CALL
38 055476 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
39 055502 004736 JSR PC,@(SP)+ ;GO REPORT ERROR
40 055504 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO BRANCH
41 055510 000240 NOP
42 055512
43
44 ;REPORT ANY 'ILF' ERROR
45 055512 032737 000001 001350 BIT #ILF,RMER1I ;WAS 'ILF' SET??
46 055520 001424 BEQ 2$ ;NO!!
47 055522 013737 001350 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
48 055530 042737 000001 001140 BIC #ILF,$GDDAT
49 055536 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
50 055544 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
51 055550 112776 000071 000000 MOVB #71,@(SP) ;WRITE ERROR NUMBER IN CALL
52 055556 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
53 055562 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
54 055564 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO BRANCH
55 055570 000240 NOP
56 055572
57

```

```

58      ;REPORT ANY 'OPI' ERROR AS
59      :
60      : . OPI DUE TO 'MOL' = 0
61      : . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
61 055572 032737 020000 001350      BIT      #OPI,RMER1I      ;WAS OPI SET??
62 055600 001433                    BEQ      31$              ;NO!!
63 055602 013737 001350 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
64 055610 042737 020000 001140      BIC      #OPI,$GDDAT
65 055616 013737 001350 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
66 055624 062716 000004              ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
67 055630 112776 000072 000000      MOV      #72,@(SP)        ;WRITE ERROR NUMBER IN USER'S CALL
68 055636 032737 010000 001346      BIT      #MOL,RMDSI      ;WAS 'MOL' = 0??
69 055644 001403                    BEQ      3$              ;YES!!
70 055646 112776 000073 000000      MOV      #73,@(SP)        ;NO - CHANGE ERROR NUMBER
71 055654 162716 000002      3$:    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
72 055660 004736                    JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
73 055662 162716 000010              SUB      #10,(SP)         ;MOVE SP BACK TO BRANCH
74 055666 000240                    NOP
75 055670      31$:
76
77      ;REPORT AN ERROR IF 'IAE' IS SET
78 055670 032737 002000 001350      BIT      #IAE,RMER1I      ;IS 'IAE' SET??
79 055676 001424                    BEQ      4$              ;NO!!
80 055700 013737 001350 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
81 055706 042737 002000 001140      BIC      #IAE,$GDDAT
82 055714 013737 001350 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
83 055722 062716 000004              ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
84 055726 112776 000070 000000      MOV      #70,@(SP)        ;WRITE ERROR NUMBER IN USER'S CALL
85 055734 162716 000002              SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
86 055740 004736                    JSR      PC,@(SP)+        ;REPORT ERROR
87 055742 162716 000010              SUB      #10,(SP)         ;MOVE SP BACK TO NO ERROR RETURN
88 055746 000240                    NOP
89 055750      4$:
90
91      ;SEE IF 'SKI' OR 'IVC' OR 'DVC' IS SET
92 055750 032737 050200 001376      BIT      #SKI!IVC!DVC,RMER2I
93 055756 001517                    BEQ      8$              ;NONE OF THE BITS ARE SET
94
95
96      ;REPORT ANY 'IVC' ERROR AS
97      :
98      : . IVC WITH VV = 0
99      : . ERRONEOUS IVC ERROR
99 055760 032737 010000 001376      BIT      #IVC,RMER2I      ;WAS IVC SET??
100 055766 001433                    BEQ      6$              ;NO!!
101 055770 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
102 055776 042737 010000 001140      BIC      #IVC,$GDDAT
103 056004 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
104 056012 062716 000004              ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
105 056016 112776 000074 000000      MOV      #74,@(SP)        ;WRITE ERROR NUMBER IN CALL
106 056024 032737 000100 001346      BIT      #VV,RMDSI        ;WAS VV = 0??
107 056032 001403                    BEQ      5$              ;YES!!
108 056034 112776 000075 000000      MOV      #75,@(SP)        ;NO - CHANGE ERROR NUMBER
109 056042 162716 000002      5$:    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
110 056046 004736                    JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
111 056050 162716 000010              SUB      #10,(SP)         ;MOVE SP BACK TO BRANCH
112 056054 000240                    NOP
113 056056      6$:
114

```

```

115 ;REPORT ANY 'SKI' ERROR
116 056056 032737 040000 001376 BIT #SKI,RMER2I ;WAS SKI SET??
117 056064 001424 BEQ 7$ ;NO!!
118 056066 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
119 056074 042737 040000 001140 BIC #SKI,$GDDAT
120 056102 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
121 056110 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
122 056114 112776 000076 000000 MOVB #76,a(SP) ;WRITE ERROR NUMBER
123 056122 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
124 056126 004736 JSR PC,a(SP)+ ;REPORT ERROR VIA USER
125 056130 162716 000010 SUB #10,(SP) ;MOVE SP TO BRANCH
126 056134 000240 NOP
127 056136 7$:
128
129 ;REPORT ANY 'DVC' ERROR
130 056136 032737 000200 001376 BIT #DVC,RMER2I ;WAS 'DVC' SET??
131 056144 001424 BEQ 8$ ;NO!!
132 056146 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
133 056154 042737 000200 001140 BIC #DVC,$GDDAT
134 056162 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
135 056170 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
136 056174 112776 000077 000000 MOVB #77,a(SP) ;WRITE ERROR NUMBER
137 056202 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
138 056206 004736 JSR PC,a(SP)+ ;REPORT ERROR VIA USER
139 056210 162716 000010 SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
140 056214 000240 NOP
141 056216 8$:
142
143 ;SEE IF 'PIP' AND 'OM' ARE 0, AND 'ATA','MOL' AND 'VV' ARE 1
144 056216 013746 001346 MOV RMDSI,-(SP) ;PUT RMDS ON STACK
145 056222 042716 047676 BIC #^C<PIP!MOL!VV!OM!ATA>,(SP)
146 056226 022726 110100 CMP #ATA!MOL!VV,(SP)+
147 056232 001002 BNE 85$
148 056234 000137 056650 JMP 13$
149 056240 85$:
150
151 ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
152 ;LINE AFTER RECALIBRATE WAS INITIATED
153 056240 032737 010000 001346 BIT #MOL,RMDSI ;DID MOL DROP??
154 056246 001030 BNE 9$ ;NO!!
155 056250 032737 020000 001350 BIT #OPI,RMER1I ;WAS OPI ERROR REPORTED??
156 056256 001024 BNE 9$ ;YES - DON'T REPORT MOL=0
157 056260 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
158 056266 052737 010000 001140 BIS #MOL,$GDDAT
159 056274 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
160 056302 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
161 056306 112776 000100 000000 MOVB #100,a(SP) ;WRITE ERROR NUMBER
162 056314 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
163 056320 004736 JSR PC,a(SP)+ ;REPORT ERROR VIA USER
164 056322 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
165 056326 000240 NOP
166 056330 9$:
167
168 ;REPORT AN ERROR IF 'VV' = 0 AND 'IVC' = 0
169 056330 032737 000100 001346 BIT #VV,RMDSI ;DID 'VV' DROP??
170 056336 001030 BNE 10$ ;NO!!
171 056340 032737 010000 001376 BIT #IVC,RMER2I ;WAS THERE A IVC ERROR??
    
```

```

172 056346 001024          BNE      10$          ;YES - DONT REPORT VV=0
173 056350 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
174 056356 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
175 056364 052737 000100 001140  BIS      #VV,$GDDAT
176 056372 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
177 056376 112776 000101 000000  MOVVB   #101,a(SP)   ;WRITE ERROR NUMBER IN CALL
178 056404 162716 000002          SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
179 056410 004736          JSR      PC,a(SP)+
180 056412 162716 000010  SUB      #10,(SP)    ;MOVE SP BACK TO USER'S BRANCH
181 056416 000240  NOP
182 056420          10$:
183
184          ;REPORT AN ERROR IF ATA IS NOT SET
185 056420 032737 100000 001346  BIT      #ATA,RMDSI  ;WAS ATA SET DURING RECALIBRATE??
186 056426 001024          BNE      11$          ;YES!!
187 056430 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
188 056436 052737 100000 001140  BIS      #ATA,$GDDAT
189 056444 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
190 056452 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
191 056456 112776 000102 000000  MOVVB   #102,a(SP)   ;WRITE ERROR NUMBER IN CALL
192 056464 162716 000002          SUB      #2,(SP)     ;MOVE SP TO USER'S BRANCH
193 056470 004736          JSR      PC,a(SP)+
194 056472 162716 000010  SUB      #10,(SP)
195 056476 000240  NOP
196
197 056500          11$:
198
199          ;REPORT AN ERROR IF 'DM' IS NOT ZERO BECAUSE RECALIBRATE SHOULD
200          ;ALWAYS CLEAR OFFSET MODE
201 056500 032737 000001 001346  BIT      #DM,RMDSI   ;WAS 'DM' RESET??
202 056506 001424          BEQ      12$          ;YES!!
203 056510 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
204 056516 042737 000001 001140  BIC      #DM,$GDDAT
205 056524 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
206 056532 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
207 056536 112776 000103 000000  MOVVB   #103,a(SP)   ;WRITE ERROR NUMBER
208 056544 162716 000002          SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
209 056550 004736          JSR      PC,a(SP)+  ;REPORT ERROR VIA USER
210 056552 162716 000010  SUB      #10,(SP)    ;MOVE SP TO USER'S BRANCH
211 056556 000240  NOP
212 056560          12$:
213
214          ;REPORT AN ERROR IF 'PIP' IS STIL ON, I.E., DRIVE NOT ON
215          ;CYLINDER
216 056560 032737 020000 001346  BIT      #PIP,RMDSI  ;IS DRIVE OFF CYLINDER??
217 056566 001430          BEQ      13$          ;NO!!
218 056570 032737 040000 001376  BIT      #SKI,RMER2I ;WAS 'SKJ' DETECTED??
219 056576 001024          BNE      13$          ;YES-DONT REPORT 'PIP'
220 056600 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
221 056606 042737 020000 001140  BIC      #PIP,$GDDAT
222 056614 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
223 056622 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
224 056626 112776 000104 000000  MOVVB   #104,a(SP)   ;WRITE ERROR NUMBER
225 056634 162716 000002          SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
226 056640 004736          JSR      PC,a(SP)+
227 056642 162716 000010  SUB      #10,(SP)    ;MOVE SP BACK TO USER'S BRANCH
228 056646 000240  NOP
    
```

```

229 056650      13$:
230
231             ;SEE IF 'ILR' OR 'RMR' OR 'UNS' IS SET
232 056650 032737 040006 001350      BIT      #ILR!RMR!UNS,RMER1I
233 056656 001514             BEQ      16$
234
235             ;REPORT AN ERROR IF 'ILR' IS SET
236 056660 032737 000002 001350      BIT      #ILR,RMER1I      ;WAS ILR SET DURING RECALIBRATE??
237 056666 001424             BEQ      14$      ;NO!!
238 056670 013737 001350 001140      MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
239 056676 042737 000002 001140      BIC      #ILR,$GDDAT
240 056704 013737 001350 001142      MOV      RMER1I,$BDDAT      ;RECEIVED STATUS
241 056712 062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
242 056716 112776 000105 000000      MOVVB   #105,@(SP) ;WRITE ERROR NUMBER IN CALL
243 056724 162716 000002             SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
244 056730 004736             JSR      PC,@(SP)+
245 056732 162716 000010             SUB      #10,(SP) ;MOVE SP TO USER'S BRANCH
246 056736 000240             NOP
247 056740
248
249             ;REPORT AN ERROR IF 'RMR' IS SET
250 056740 032737 000004 001350      BIT      #RMR,RMER1I      ;WAS RMR SET??
251 056746 001424             BEQ      15$      ;NO!!
252 056750 013737 001350 001140      MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
253 056756 042737 000004 001140      BIC      #RMR,$GDDAT
254 056764 013737 001350 001142      MOV      RMER1I,$BDDAT      ;RECEIVED STATUS
255 056772 062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
256 056776 112776 000106 000000      MOVVB   #106,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
257 057004 162716 000002             SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
258 057010 004736             JSR      PC,@(SP)+ ;REPORT ERROR VIA USER
259 057012 162716 000010             SUB      #10,(SP) ;MOVE SP TO USER'S BRANCH
260 057016 000240             NOP
261 057020
262
263             ;REPORT AN ERROR IF 'UNS' IS SET AND 'DVC' IS 0
264 057020 032737 040000 001350      BIT      #UNS,RMER1I      ;WAS UNSAFE ON??
265 057026 001430             BEQ      16$      ;NO!!
266 057030 032737 000200 001376      BIT      #DVC,RMER2I      ;WAS THERE A DEVICE CHECK??
267 057036 001024             BNE     16$      ;YES - DON'T REPORT UNSAFE
268 057040 013737 001350 001140      MOV      RMER1I,$GDDAT      ;EXPECTED STATUS
269 057046 042737 040000 001140      BIC      #UNS,$GDDAT
270 057054 013737 001350 001142      MOV      RMER1I,$BDDAT      ;RECEIVED STATUS
271 057062 062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
272 057066 112776 000107 000000      MOVVB   #107,@(SP) ;WRITE ERROR NUMBER
273 057074 162716 000002             SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
274 057100 004736             JSR      PC,@(SP)+ ;REPORT ERROR VIA USER
275 057102 162716 000010             SUB      #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
276 057106 000240             NOP
277 057110
278
279             ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
280 057110 062716 000004             ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
281 057114 105776 000000             TSTB   @(SP) ;WAS AN ERROR REPORTED??
282 057120 001403             BEQ      17$      ;NO!!
283 057122 062716 000004             ADD      #4,(SP) ;YES - AUGMENT SP RETURN
284 057126 000402             BR      18$
285 057130 162716 000004             SUB      #4,(SP) ;NO ERROR - RETURN SP TO BRANCH
    
```


286 057134 000240
287 057136 000207
288

18\$:

NOP
RTS

PC

;STATUS CECK IS COMPLETE

```

1      .SBTTL  DRIVE CLEAR STATUS CHECK SUBROUTINE
2
3      :      BR      ???      RETURN HERE IF NO ERROR
4      :      NOP      RETURN HERE TO REPORT AN ERROR
5      :      ERROR   ERROR NUMBER DEFINED BY SUB
6      :      JSR     PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
7      :      ???      RETURN HERE IF NO MORE ERRORS
8
9 057140  DRVSTS:
10
11      :CLEAR USER'S ERROR CALL
12 057140 062716 000004      ADD     #4,(SP)      ;MOVE SP TO ERROR CALL
13 057144 105076 000000      CLRB   @(SP)      ;CLEAR ERROR CALL
14 057150 162716 000004      SUB     #4,(SP)      ;MOVE SP TO USER'S BRANCH
15
16 057154 013737 001334 001142 4$:  :REPORT ERROR IF RMCS1 NOT INITIALIZED
17 057162 042737 173700 001142      MOV     RMCSI,$BDDAT ;CHECK RMCS1
18 057170 012737 004010 001140      BIC     #^C<DVA!FNCMSK>,$BDDAT ;CLEAR DONT CARES
19 057176 023737 001140 001142      MOV     #DVA!DRVCLR,$GDDAT ;EXPECT DVA
20 057204 001443      CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
21 057206 062716 000004      BEQ     6$          ;BRANCH IF EQUAL
22 057212 112776 000141 000000      ADD     #4,(SP)      ;MOVE SP TO ERROR CALL
23 057220 162716 000002      MOVB   #141,@(SP)   ;WRITE NUMBER OF ERROR IN CALL
24 057224 004736      SUB     #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
25 057226 162716 000010      JSR     PC,@(SP)+   ;REPORT THE ERROR VIA USER
26 057232 000240      SUB     #10,(SP)    ;MOVE SP TO NO ERROR RETURN
27      NOP
28 057234 013737 001346 001142 5$:  :REPORT ERROR IF RMDS NOT INITIALIZED
29 057242 042737 021101 001142      MOV     RMDSI,$BDDAT ;CHECK RMDS
30 057250 012737 010600 001140      BIC     #PGM!OM!VV!PIP,$BDDAT ;CLEAR DONT CARES
31 057256 023737 001140 001142      MOV     #MOL!DPR!DRY,$GDDAT ;EXPECT DRY & DPR
32 057264 001413      CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
33 057266 062716 000004      BEQ     6$          ;BRANCH IF EQUAL
34 057272 112776 000142 000000      ADD     #4,(SP)      ;MOVE SP TO ERROR CALL
35 057300 162716 000002      MOVB   #142,@(SP)   ;WRITE NUMBER OF ERROR IN CALL
36 057304 004736      SUB     #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
37 057306 162716 000010      JSR     PC,@(SP)+   ;REPORT THE ERROR VIA USER
38 057312 000240      SUB     #10,(SP)    ;MOVE SP TO NO ERROR RETURN
39      NOP
40 057314 005037 001140 6$:  :REPORT ERROR IF RMER1 NOT INITIALIZED
41 057320 013737 001350 001142      CLR     $GDDAT      ;EXPECT 0'S
42 057326 001413      MOV     RMER1I,$BDDAT ;CHECK RMER1
43 057330 062716 000004      BEQ     8$          ;BRANCH IF EQUAL
44 057334 112776 000143 000000      ADD     #4,(SP)      ;MOVE SP TO ERROR CALL
45 057342 162716 000002      MOVB   #143,@(SP)   ;WRITE NUMBER OF ERROR IN CALL
46 057346 004736      SUB     #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
47 057350 162716 000010      JSR     PC,@(SP)+   ;REPORT THE ERROR VIA USER
48 057354 000240      SUB     #10,(SP)    ;MOVE SP TO NO ERROR RETURN
49      NOP
50 057356 013737 001352 001142 8$:  :REPORT ERROR IF ATA NOT INITIALIZED
51 057364 010146      MOV     RMASI,$BDDAT ;CHECK ATTENTION BIT
52 057366 010246      MOV     R1,-(SP)    ;;PUSH R1 ON STACK
53 057370 013701 001464      MOV     R2,-(SP)    ;;PUSH R2 ON STACK
54 057374 116102 000001      MOV     TSTQUE,R1
55 057400 042702 177400      MOVB   1(R1),R2
56 057404 005102      BIC     #^CATNMSK,R2
57 057406 040237 001142      COM     R2
      BIC     R2,$BDDAT
    
```

```

58 057412 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
59 057414 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
60 057416 005737 001142  TST      $BDDAT        ;IS ATTENTION CLEARED??
61 057422 001413      BEQ      9$           ;BRANCH IF ATTENTION CLEARED
62 057424 062716 000004  ADD      #4,(SP)       ;MOVE SP TO ERROR CALL
63 057430 112776 000144 000000  MOVB     #144,@(SP)    ;WRITE NUMBER OF ERROR IN CALL
64 057436 162716 000002      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
65 057442 004736      JSR      PC,@(SP)+    ;REPORT THE ERROR VIA USER
66 057444 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR RETURN
67 057450 000240      NOP
68
69 057452 013737 001360 001142 9$:      :REPORT ERROR IF RMMR1 NOT INITIALIZED
70 057460 042737 000046 001142  MOV      RMMR1I,$BDDAT ;CHECK RMMR
71 057466 012737 000010 001140  BIC      #WC!LS!LST,$BDDAT ;CLEAR DONT CARES
72 057474 023737 001140 001142  MOV      #MWD,$GDDAT   ;EXPECT WRITE DATA ON
73 057502 001413      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
74 057504 062716 000004  BEQ      11$          ;BRANCH IF ZERO
75 057510 112776 000145 000000  ADD      #4,(SP)       ;MOVE SP TO ERROR CALL
76 057516 162716 000002  MOVB     #145,@(SP)    ;WRITE NUMBER OF ERROR IN CALL
77 057522 004736      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
78 057524 162716 000010      JSR      PC,@(SP)+    ;REPORT THE ERROR VIA USER
79 057530 000240      SUB      #10,(SP)     ;MOVE SP TO NO ERROR RETURN
80
81 057532 013737 001374 001142 11$:    :REPORT ERROR IF RMMR2 NOT INITIALIZED
82 057540 042737 140000 001142  MOV      RMMR2I,$BDDAT ;CHECK RMMR2
83 057546 012737 011777 001140  BIC      #RQA!RQB,$BDDAT ;CLEAR REQA, REQB
84 057554 023737 001140 001142  MOV      #TST!1777,$GDDAT ;EXPECT TEST BIT ON
85 057562 001413      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
86 057564 062716 000004  BEQ      15$          ;BRANCH IF EQUAL
87 057570 112776 000146 000000  ADD      #4,(SP)       ;MOVE SP TO ERROR CALL
88 057576 162716 000002  MOVB     #146,@(SP)    ;WRITE NUMBER OF ERROR IN CALL
89 057602 004736      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
90 057604 162716 000010      JSR      PC,@(SP)+    ;REPORT THE ERROR VIA USER
91 057610 000240      SUB      #10,(SP)     ;MOVE SP TO NO ERROR RETURN
92 057612 005037 001140 15$:    CLR      $GDDAT        ;EXPECT ZEROS
93
94 057616 013737 001402 001142 17$:    :REPORT ERROR IF RMEC2 NOT RESET
95 057624 001413      MOV      RMEC2I,$BDDAT ;CHECK RMEC2
96 057626 062716 000004  BEQ      17$          ;BRANCH IF 0
97 057632 112776 000150 000000  ADD      #4,(SP)       ;MOVE SP TO ERROR CALL
98 057640 162716 000002  MOVB     #150,@(SP)    ;WRITE NUMBER OF ERROR IN CALL
99 057644 004736      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
100 057646 162716 000010      JSR      PC,@(SP)+    ;REPORT THE ERROR VIA USER
101 057652 000240      SUB      #10,(SP)     ;MOVE SP TO NO ERROR RETURN
102
103 057654 013737 001376 001142 18$:    :REPORT ERROR IF RMER2 NOT RESET
104 057662 001413      MOV      RMER2I,$BDDAT ;CHECK RMER2
105 057664 062716 000004  BEQ      18$          ;BRANCH IF NO ERROR
106 057670 112776 000147 000000  ADD      #4,(SP)       ;MOVE SP TO ERROR CALL
107 057676 162716 000002  MOVB     #147,@(SP)    ;WRITE NUMBER OF ERROR IN CALL
108 057702 004736      SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
109 057704 162716 000010      JSR      PC,@(SP)+    ;REPORT THE ERROR VIA USER
110 057710 000240      SUB      #10,(SP)     ;MOVE SP TO NO ERROR RETURN
111 057712
112
113 057712
114

```

```
115 ;AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND
116 057712 062716 000004 ADD #4,(SP) ;MOVE SP TO ERROR CALL
117 057716 105776 000000 TSTB @ (SP) ;WAS AN ERROR DETECTED??
118 057722 001403 BEQ 21$ ;NO!!
119 057724 062716 000004 ADD #4,(SP) ;YES - MOVE SP TO ERROR RETURN
120 057730 000402 BR 23$
121 057732 162716 000004 21$: SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
122 057736 000240 23$: NOP
123 057740 000207 RTS PC ;RETURN TO USER
124
```

```

1      .SBTTL DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE
2
3      :THIS SUBROUTINE VERIFIES THE RESULTS OF ALL DATA TRANSFER COMMANDS
4      :USING STATUS STORED IN THE GET BUFFER AND TEST PARAMETERS
5      :STORED IN THE PUT BUFFER. ERRORS ARE REPORTED BY WRITING
6      :THE ERROR NUMBER IN THE USERS ERROR CALL.
7
8      :USER'S SUBROUTINE CALL:
9      :(1) JSR PC,DTASTS
10     :(2) BR ?? RETURN HERE IF NO DATA ERRORS
11     :(3) NOP RETURN HERE TO REPORT AN ERROR
12     :(4) ERROR SUB WRITES ERROR NUMBER
13     :(5) JSR PC,@(SP)+ USER RETURNS FOR MORE CHECKS
14     :(6) ?? SUB RETURNS HERE AFTER ALL
15     : ERRORS ARE REPORTED
16
17 057742 DTASTS:
18
19     :CLEAR USER'S ERROR CALL AND ERROR FLAGS
20 057742 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
21 057746 105076 000000 CLR @ (SP) ;CLEAR LOW ORDER BYTE OF TRAP
22 057752 162716 000004 SUB #4,(SP) ;RESTORE SP TO NO ERROR
23 057756 005037 063336 CLR 500$ ;CLEAR ERROR FLAGS
24
25     :REPORT ANY CONTROL BUS PARITY ERROR WHILE READING REMOTE REGISTERS,
26     :I.E., MCPE = 1
27 057762 032737 020000 001334 BIT #MCPE,RMCS1I ;WAS THERE A PARITY ERROR??
28 057770 001422 BEQ 10$ ;NO!!
29 057772 013737 001334 001140 MOV RMCS1I,$GDDAT ;EXPECTED STATUS
30 060000 042737 020000 001140 BIC #MCPE,$GDDAT
31 060006 013737 001334 001142 MOV RMCS1I,$BDDAT ;RECEIVED STATUS
32 060014 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
33 060020 112776 000013 000000 MOV #13,@(SP) ;WRITE ERROR NUMBER
34 060026 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
35 060032 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
36 060034 000466 BR 30$
37 060036
38
39     :REPORT ANY CONTROL BUS PARITY ERROR WHILE WRITTING REMOTE REGISTERS,
40     :I.E., PAR = 1 AND DPE = 0
41 060036 032737 000010 001350 BIT #PAR,RMER1I ;WAS THERE A PARITY ERROR??
42 060044 001435 BEQ 20$ ;NO!!
43 060046 032737 000010 001376 BIT #DPE,RMER2I ;DATA PARITY ERROR ?
44 060054 001031 BNE 20$ ;YES!!
45 060056 013737 001350 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
46 060064 042737 000010 001140 BIC #PAR,$GDDAT
47 060072 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
48 060100 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
49 060104 112776 000050 000000 MOV #50,@(SP) ;WRITE ERROR NUMBER
50 060112 032737 001000 001344 BIT #MXF,RMCS2I ;DID MXF GET SET??
51 060120 001003 BNE 15$ ;YES!!
52 060122 112776 000274 000000 MOV #274,@(SP) ;NO - CHANGE ERROR NUMBER
53 060130 162716 000002 15$: SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
54 060134 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
55 060136 000425 BR 30$
56
57     :LOOK FOR ANY ERRORS WHICH MAY HAVE OCCURRED DURING COMMAND INITIATION OR
    
```

```

58      ;MECHANICAL POSITIONING
59
60      ;FIRST TEST MXF WHICH WOULD INDICATE COMPOSITE ERROR SET WHEN FUNCTION
61      ;CODE AND GO BIT WERE LOADED
62      060140
63      060140 032737 001000 001344      20$:      BIT      #MXF,RMCS2I      ;WAS 'MISSED TRANSFER' SET??
64      060146 001425                      BEQ      40$              ;NO!!
65      060150 013737 001344 001140      MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
66      060156 042737 001000 001140      BIC      #MXF,$GDDAT
67      060164 013737 001344 001142      MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
68      060172 062716 000004                      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
69      060176 112776 000275 000000      MOVVB   #275,@(SP)        ;WRITE ERROR NUMBER
70      060204 162716 000002                      SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
71      060210 004736                      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
72      060212
73
74      ;RESTORE SP TO NO ERROR RETURN AND BYPASS FURHTER STATUS CHECKING
75      060212 162716 000010                      SUB      #10,(SP)         ;MOVE SP TO NO ERROR
76      060216 000137 063310                      JMP      380$             ;SKIP TO END OF SUB
77
78      40$:
79
80      ;REPORT AN ERROR IF 'DPI' ERROR OCCURRED DUE TO 'MOL' = 0, OR IF 'DPI'
81      ;AND 'MOL' ARE SET, BUT 'VV' IS RESET, INDICATING AN INTERMITTENT
82      ;'MOL'
83      060222 032737 020000 001350      40$:      BIT      #OPI,RMER1I      ;IS 'DPI' SET??
84      060230 001447                      BEQ      60$              ;NO!!
85      060232 013737 001350 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
86      060240 042737 020000 001140      BIC      #OPI,$GDDAT
87      060246 013737 001350 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
88      060254 032737 010000 001346      BIT      #MOL,RMDSI      ;WAS MEDIUM OFF LINE??
89      060262 001404                      BEQ      45$              ;YES!!
90      060264 032737 000100 001346      BIT      #VV,RMDSI        ;WAS 'MOL' INTERMITTENT??
91      060272 001013                      BNE     50$              ;NO!!
92      060274 062716 000004                      45$:      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
93      060300 112776 000276 000000      MOVVB   #276,@(SP)        ;WRITE ERROR NUMBER IN CALL
94      060306 162716 000002                      SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
95      060312 004736                      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
96      060314 162716 000010                      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
97      060320 000413
98      060322
99
100     ;REPORT 'DPI' ERROR, WHICH IS DUE TO 'ON CYLINDER' NOT DROPPING OR
101     ;'RUN' TIMEOUT (20 MS) OR SEARCH TIMEOUT (50 MS)
102     060322 062716 000004                      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
103     060326 112776 000277 000000      MOVVB   #277,@(SP)        ;WRITE ERROR NUMBER IN CALL
104     060334 162716 000002                      SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
105     060340 004736                      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
106     060342 162716 000010                      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
107     060346 000240
108     060350
109
110     ;LOOK FOR 'IVC' ERROR DURING COMMAND INITIATION
111     060350 032737 010000 001376      60$:      BIT      #IVC,RMER2I      ;WAS THERE AN 'IVC' ERROR??
112     060356 001432                      BEQ      70$              ;NO!!
113     ;REPORT 'IVC' ERROR DUE TO 'VV' = 0, OR REPORT ERRONEOUS 'IVC' ERROR
114     060360 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
    
```

```

115 060366 042737 010000 001140      BIC      #IVC,$GDDAT
116 060374 013737 001376 001142      MOV      RMER2I,$BDDAT      ;RECEIVED STATUS
117 060402 062716 000004      ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR
118 060406 112776 000300 000000      MOVVB   #300,@(SP)        ;WRITE ERROR NUMBER IN CALL
119 060414 032737 000100 001346      BIT      #VV,RMSDI        ;WAS VOLUME VALID??
120 060422 001403      BEQ      65$              ;NO!!
121 060424 112776 000301 000000      MOVVB   #301,@(SP)        ;CHANGE ERROR NUMBER
122 060432 162716 000002      65$:    SUB      #2,(SP)      ;MOVE SP TO RETURN IF ERROR
123 060436 004736      JSR      PC,@(SP)+        ;REPORT 'IVC' ERROR AND RETURN
124 060440 162716 000010      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
125 060444      70$:
126
127      ;SEE IF 'ILF' OR 'RMR' IS SET
128 060444 032737 000007 001350      BIT      #ILR!ILF!RMR,RMER1I
129 060452 001510      BEQ      100$            ;NO ERRORS DETECTED
130      ;REPORT AN ERROR IF 'ILR' IS SET
131 060454 032737 000002 001350      BIT      #ILR,RMER1I     ;WAS 'ILR' DETECTED??
132 060462 001424      BEQ      80$              ;NO!!
133 060464 013737 001350 001140      MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
134 060472 042737 000002 001140      BIC      #ILR,$GDDAT
135 060500 013737 001350 001142      MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
136 060506 062716 000004      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
137 060512 112776 000302 000000      MOVVB   #302,@(SP)        ;WRITE ERROR NUMBER IN CALL
138 060520 162716 000002      SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
139 060524 004736      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
140 060526 162716 000010      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
141 060532 000240      NOP
142 060534      80$:
143
144      ;REPORT AN ERROR IF 'ILF' IS SET
145 060534 032737 000001 001350      BIT      #ILF,RMER1I     ;WAS 'ILF' DETECTED??
146 060542 001424      BEQ      90$              ;NO!!
147 060544 013737 001350 001140      MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
148 060552 042737 000001 001140      BIC      #ILF,$GDDAT
149 060560 013737 001350 001142      MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
150 060566 062716 000004      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
151 060572 112776 000303 000000      MOVVB   #303,@(SP)        ;WRITE ERROR NUMBER IN CALL
152 060600 162716 000002      SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
153 060604 004736      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
154 060606 162716 000010      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
155 060612 000240      NOP
156 060614      90$:
157
158      ;REPORT AN ERROR IF 'RMR' IS SET
159 060614 032737 000004 001350      BIT      #RMR,RMER1I     ;WAS 'RMR' DETECTED??
160 060622 001424      BEQ      100$            ;NO!!
161 060624 013737 001350 001140      MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
162 060632 042737 000004 001140      BIC      #RMR,$GDDAT
163 060640 013737 001350 001142      MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
164 060646 062716 000004      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
165 060652 112776 000304 000000      MOVVB   #304,@(SP)        ;WRITE ERROR NUMBER IN CALL
166 060660 162716 000002      SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
167 060664 004736      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
168 060666 162716 000010      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
169 060672 000240      NOP
170 060674      100$:
171      ;DETERMINE WHETHER OR NOT 'IAE' SHOULD BE SET AND CHECK FOR ERROR
    
```

```

172 060674 012737 002000 001140      MOV      #IAE,$GDDAT      ;SETUP FOR "IAE" = 1
173 060702 052737 040000 063336      BIS      #SKI,500$       ;SETUP FOR "SKI" = 1
174 060710 023727 001444 001166      CMP      RMDCO,#822.     ;GREATER THAN LAST CYLINDER ?
175 060716 101025                BHI      110$           ;YES - CYLINDER IS INVALID
176 060720 042737 040000 063336      BIC      #SKI,500$       ;RESET SKI FLAG
177
178 060726 123737 001417 001333      CMPB     RMDAO+1,LSTRK+1 ;GREATER THAN LAST TRACK ?
179 060734 101016                BHI      110$           ;YES - TRACK IS INVALID
180
181 060736 123727 001416 000035      CMPB     RMDAO,#29.      ;IS SECTOR > 29. ?
182 060744 101410                BLOS     105$           ;NO
183 060746 032737 010000 001442      BIT      #FMT16,RMOFO   ;18 BIT FORMAT ?
184 060754 001406                BEQ      110$           ;YES - SECTOR IS INVALID FOR 18 BIT MODE
185 060756 123727 001416 000037      CMPB     RMDAO,#31.      ;IS SECTOR > 31. ?
186 060764 101002                BHI      110$           ;YES - SECTOR IS INVALID
187 060766 005037 001140          105$:   CLR      $GDDAT      ;"IAE" SHOULD = 0
188
189 060772 013737 001350 001142      110$:   MOV      RMER1I,$BDDAT    ;GET RECEIVED STATUS
190 061000 042737 175777 001142      BIC      #^CIAE,$BDDAT
191 061006 023737 001140 001142      CMP      $GDDAT,$BDDAT  ;IS "IAE" STATUS OK??
192 061014 001004                BNE      115$           ;NO!!
193 061016 042737 040000 063336      BIC      #SKI,500$       ;IAE OK - SKI SHOULD BE 0
194 061024 000412                BR
195 061026 062716 000004          115$:   ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
196 061032 112776 000305 000000      MOVVB    #305,a(SP)      ;WRITE ERROR NUMBER
197 061040 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
198 061044 004736                JSR      PC,a(SP)+       ;REPORT ERROR AND RETURN
199 061046 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
200 061052                120$:
201
202          ;REPORT AN ERROR IF "SKI" IS SET AND "IAE" STATUS WAS OK
203 061052 013737 001376 001142      MOV      RMER2I,$BDDAT  ;RECEIVED STATUS
204 061060 042737 137777 001142      BIC      #^CSKI,$BDDAT
205 061066 013737 063336 001140      MOV      500$,$GDDAT    ;EXPECTED STATUS
206 061074 042737 137777 001140      BIC      #^CSKI,$GDDAT
207 061102 032737 040000 001376      BIT      #SKI,RMER2I    ;WAS "SKI" SET??
208 061110 001417                BEQ      140$           ;NO!!
209 061112 032737 040000 063336      BIT      #SKI,500$       ;WAS SKI CAUSED BY IAE = 0??
210 061120 001032                BNE      150$           ;YES - DON'T REPORT SKI
211 061122 062716 000004          ADD      #4,(SP)         ;MOVE SP TO USERS ERROR CALL
212 061126 112776 000306 000000      MOVVB    #306,a(SP)      ;WRITE ERROR NUMBER
213 061134 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
214 061140 004736                JSR      PC,a(SP)+       ;REPORT ERROR AND RETURN
215 061142 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
216 061146 000417                BR      150$
217
218 061150                140$:
219
220          ;REPORT AN ERROR IF SKI = 0 AND IAE WAS NOT DETECTED
221 061150 032737 040000 063336      BIT      #SKI,500$       ;SHOULD SKI BE SET??
222 061156 001413                BEQ      150$           ;NO!!
223 061160 062716 000004          ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
224 061164 112776 000307 000000      MOVVB    #307,a(SP)      ;WRITE ERROR NUMBER IN CALL
225 061172 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
226 061176 004736                JSR      PC,a(SP)+       ;REPORT ERROR AND RETURN
227 061200 162716 000010          SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
228 061204 000240                NOP
    
```



```

229 061206      150$:
230
231             ;LOOK FOR 'LSC' OR 'LBC' OR 'DVC' IN ERROR REGISTER #2
232 061206 032737 006200 001376      BIT      #LSC!LBC!DVC,RMER2I
233 061214 001512             BEQ      180$             ;NO ERRORS SET
234
235             ;REPORT ANY DEVICE FAULT, I.E., 'DVC' = 1
236 061216 032737 000200 001376      BIT      #DVC,RMER2I      ;IS 'DVC' = 1??
237 061224 001424             BEQ      160$             ;NO!!
238 061226 013737 001376 001140      MOV      RMER2I,$GDDAT      ;EXPECTED STATUS
239 061234 042737 000200 001140      BIC      #DVC,$GDDAT
240 061242 013737 001376 001142      MOV      RMER2I,$BDDAT      ;RECEIVED STATUS
241 061250 062716 000004             ADD      #4,(SP)           ;MOVE SP TO USERS ERROR
242 061254 112776 000310 000000      MOVVB   #310,@(SP)        ;WRITE ERROR NUMBER IN CALL
243 061262 162716 000002             SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
244 061266 004736             JSR      PC,@(SP)+         ;REPORT ERROR AND RETURN
245 061270 162716 000010             SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
246 061274 000240             NOP
247 061276
248
249             ;REPORT LOSS OF BIT CLOCK, I.E.: 'LBC' = 1, IF 'MOL' = 1
250 061276 032737 002000 001376      BIT      #LBC,RMER2I      ;IS LBC SET??
251 061304 001430             BEQ      170$             ;NO!!
252 061306 032737 010000 001346      BIT      #MOL,RMDSI       ;WAS LBC ERROR BY MOL = 0
253 061314 001424             BEQ      170$             ;YES!!
254 061316 013737 001376 001140      MOV      RMER2I,$GDDAT      ;EXPECTED STATUS
255 061324 042737 002000 001140      BIC      #LBC,$GDDAT
256 061332 013737 001376 001142      MOV      RMER2I,$BDDAT      ;RECEIVED STATUS
257 061340 062716 000004             ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
258 061344 112776 000311 000000      MOVVB   #311,@(SP)        ;WRITE ERROR NUMBER IN CALL
259 061352 162716 000002             SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
260 061356 004736             JSR      PC,@(SP)+         ;REPORT ERROR AND RETURN
261 061360 162716 000010             SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
262 061364 000240             NOP
263 061366
264
265             ;REPORT LOS OF SYSTEM CLOCK, I.E., 'LSC' = 1
266 061366 032737 004000 001376      BIT      #LSC,RMER2I      ;IS 'LSC' = 1??
267 061374 001422             BEQ      180$             ;NO!!
268 061376 013737 001376 001140      MOV      RMER2I,$GDDAT      ;EXPECTED STATUS
269 061404 042737 004000 001140      BIC      #LSC,$GDDAT
270 061412 013737 001376 001142      MOV      RMER2I,$BDDAT      ;RECEIVED STATUS
271 061420 062716 000004             ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
272 061424 112776 000312 000000      MOVVB   #312,@(SP)        ;WRITE ERROR NUMBER
273 061432 004736             JSR      PC,@(SP)+         ;REPORT ERROR AND RETURN
274 061434 162716 000010             SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
275 061440 000240             NOP
276 061442
277
278             ;LOOK FOR 'UNS' OR 'DTE' OR 'WLE' IN ERROR REGISTER #1
279 061442 032737 054000 001350      BIT      #UNS!DTE!WLE,RMER1I
280 061450 001527             BEQ      220$             ;NO BITS SET
281
282             ;REPORT 'UNS' ERROR IF 'DVC' = 0
282 061452 032737 040000 001350      BIT      #UNS,RMER1I      ;IS 'UNS' SET??
283 061460 001427             BEQ      190$             ;NO!!
284 061462 032737 000200 001376      BIT      #DVC,RMER2I      ;WAS 'UNS' CAUSED BY 'DVC'??
285 061470 001023             BNE      190$             ;YES!!
    
```

```

286 061472 013737 001350 001140      MOV      RMER1I,$GDDAT      :EXPECTED STATUS
287 061500 042737 040000 001140      BIC      #UNS,$GDDAT
288 061506 013737 001350 001142      MOV      RMER1I,$BDDAT      :RECEIVED STATUS
289 061514 062716 000004                ADD      #4,(SP)            :MOVE SP TO USERS ERROR CALL
290 061520 112776 000313 000000      MOVVB   #313,@(SP)        :WRITE ERROR NUMBER
291 061526 162716 000002                SUB      #2,(SP)            :MOVE SP TO RETURN IF ERROR
292 061532 004736                JSR      PC,@(SP)+         :REPORT ERROR AND RETURN
293 061534 162716 000010                SUB      #10,(SP)         :RESTORE SP TO NO ERROR
294 061540
295
296
297 061540 032737 010000 001350      :REPORT ANY DRIVE TIMING ERROR, I.E., 'DTE' = 1
298 061546 001423                BIT      #DTE,RMER1I      :IS DTE SET??
299 061550 013737 001350 001140      BEQ      200$              :NO!!
300 061556 042737 010000 001140      MOV      RMER1I,$GDDAT      :EXPECTED STATUS
301 061564 013737 001350 001142      BIC      #DTE,$GDDAT
302 061572 062716 000004                MOV      RMER1I,$BDDAT      :RECEIVED STATUS
303 061576 112776 000314 000000      ADD      #4,(SP)            :MOVE SP TO USER'S ERROR CALL
304 061604 162716 000002                MOVVB   #314,@(SP)        :WRITE ERROR NUMBER IN CALL
305 061610 004736                SUB      #2,(SP)            :MOVE SP TO RETURN IF ERROR
306 061612 162716 000010                JSR      PC,@(SP)+         :REPORT ERROR AND RETURN
307 061616
308
309
310
311 061616 032737 004000 001350      :REPORT AN ERROR IF WRITE LOCK ERROR IS SET.  SEE IF DRIVE IS NOT
312 061624 001441                :WRITE PROTECTED, OR IF FUNCTION WAS NOT A WRITE
313 061626 013737 001350 001142      BIT      #WLE,RMER1I      :WAS 'WLE' SET??
314 061634 013737 001350 001140      BEQ      220$              :NO!!
315 061642 052737 004000 001140      MOV      RMER1I,$BDDAT      :RECEIVED STATUS
316 061650 062716 000004                MOV      RMER1I,$GDDAT      :EXPECTED STATUS
317 061654 112776 000315 000000      BIS      #WLE,$GDDAT
318 061662 032737 004000 001346      ADD      #4,(SP)            :MOVE SP TO USERS ERROR CALL
319 061670 001404                MOVVB   #315,@(SP)        :WRITE ERROR NUMBER IN CALL
320 061672 032737 000010 001410      BIT      #WRL,RMDSI        :WAS DRIVE WRITE PROTECTED??
321 061700 001406                BEQ      205$              :NO!!
322 061702 112776 000316 000000      BIT      #BIT3,RMCS10      :WAS COMMAND A WRITE??
323 061710 042737 004000 001140      BEQ      210$              :YES!!
324 061716 162716 000002                MOVVB   #316,@(SP)        :CHANGE ERROR NUMBER
325 061722 004736                BIC      #WLE,$GDDAT
326 061724 162716 000010                SUB      #2,(SP)            :MOVE SP TO RETURN IF ERROR
327
328 061730                JSR      PC,@(SP)+         :REPORT ERROR AND RETURN
329
330
331 061730 062716 000004                MOV      #4,(SP)            :MOVE SP TO USER'S ERROR
332 061734 105776 000000                TSTB    @(SP)              :WAS ERROR DETECTED??
333 061740 001404                BEQ      225$              :NO - DO DATA CHECKS
334 061742 162716 000004                SUB      #4,(SP)            :RESTORE SP
335 061746 000137 062750                JMP      340$              :SKIP DATA CHECKS
336 061752 162716 000004                SUB      #4,(SP)            :RESTORE SP
337
338
339
340 061756 013737 001410 063340      :CHECK HEADER ERRORS IF FUNCTION WAS NOT WRITE HEADER AND DATA, AND
341 061764 042737 177700 063340      :IF HEADER COMPARE IS NOT INHIBITED
342 061772 022737 000063 063340      MOV      RMCS10,510$       :STRIP AND STORE FUNCTION CODE
                                BIC      #^CFNCMSK,510$
                                CMP      #WH!GO,510$       :WAS FUNCTION WRITE HEADER & DATA??
    
```

```

343 062000 001512          BEQ      250$          ;YES - SKIP HEADER CHECKS
344 062002 032737 002000 001366  BIT      #HCI,RMOFI      ;WAS HCI SET??
345 062010 001106          BNE      250$          ;YES - SKIP HEADER CHECKS
346
347
348 062012 032737 000620 001350  ;SEE IF ANY HEADER ERRORS ARE SET, I.E., 'FER' OR 'HCRC' OR 'HCE'
349 062020 001533          BEQ      270$          ;NO ERRORS SET
350
351
352 062022 032737 000400 001350  ;REPORT HEADER CRC ERROR IF SET
353 062030 001422          BIT      #HCRC,RMER1I    ;WAS HCRC SET??
354 062032 013737 001350 001140  BEQ      230$          ;NO!!
355 062040 042737 000400 001140  MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
356 062046 013737 001350 001142  BIC      #HCRC,$GDDAT
357 062054 062716 000004          MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
358 062060 112776 000317 000000  ADD      #4,(SP)         ;MOVE SP TO USERS ERROR
359 062066 162716 000002          MOVVB   #317,@(SP)       ;WRITE ERROR NUMBER
360 062072 004736          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
361 062074 000501          JSR     PC,@(SP)+       ;REPORT ERROR AND RETURN
362 062076          BR      260$
363
364
365 062076 032737 000020 001350  ;REPORT FORMAT ERROR IF SET
366 062104 001422          BIT      #FER,RMER1I    ;WAS 'FER' SET??
367 062106 013737 001350 001140  BEQ      240$          ;NO!!
368 062114 042737 000020 001140  MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
369 062122 013737 001350 001142  BIC      #FER,$GDDAT
370 062130 062716 000004          MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
371 062134 112776 000320 000000  ADD      #4,(SP)         ;MOVE SP TO USERS ERROR
372 062142 162716 000002          MOVVB   #320,@(SP)       ;WRITE ERROR NUMBER
373 062146 004736          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
374 062150 000453          JSR     PC,@(SP)+       ;REPORT ERROR AND RETURN
375 062152          BR      260$
376
377
378 062152 032737 000200 001350  ;REPORT HEADER COMPARE ERROR IF SET
379 062160 001453          BIT      #HCE,RMER1I    ;WAS 'HCE' SET??
380 062162 013737 001350 001140  BEQ      270$          ;NO!!
381 062170 042737 000200 001140  MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
382 062176 013737 001350 001142  BIC      #HCE,$GDDAT
383 062204 062716 000004          MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
384 062210 112776 000321 000000  ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR
385 062216 162716 000002          MOVVB   #321,@(SP)       ;WRITE ERROR NUMBER
386 062222 004736          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
387 062224 000425          JSR     PC,@(SP)+       ;REPORT ERROR AND RETURN
388
389
390
391 062226 032737 000620 001350  ;THERE SHOULD BE NO HEADER ERRORS BECAUSE
392 062234 001425          ;.COMMAND WAS WRITE HEADER AND DATA, OR
393 062236 013737 001350 001140  ;.HEADER COMPARE INHIBIT WAS SET
394 062244 042737 000620 001140  250$: BIT      #HCE!FER!HCRC,RMER1I
395 062252 013737 001350 001142  BEQ      270$          ;NO ERRORS WERE SET
396 062260 062716 000004          MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
397 062264 112776 000322 000000  BIC      #HCE!FER!HCRC,$GDDAT
398 062272 162716 000002          MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
399 062276 004736          ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
                                MOVVB   #322,@(SP)       ;WRITE ERROR NUMBER
                                SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
                                JSR     PC,@(SP)+       ;REPORT ERROR AND RETURN
    
```

```

400 062300 162716 000010      260$:  SUB    #10,(SP)      ;MOVE SP TO NO ERROR
401 062304 000137 062750      JMP    340$      ;OMIT FURTHER DATA CHECKS
402
403 062310      270$:
404
405      ;IF COMMAND WAS A WRITE COMMAND, GO DO WRITE ERROR CHECKS, OTHERWISE
406      ;DO READ ERROR CHECKS
407 062310 032737 000010 063340      BIT    #BIT3,510$  ;WAS THIS A WRITE COMMAND?
408 062316 001002      BNE    275$      ;NO!!
409 062320 000137 062536      JMP    310$      ;GO DO WRITE STATUS CHECK
410 062324
411
412      ;REPORT DATA CHECK IF SET
413 062324 032737 100000 001350      BIT    #DCK,RMER1I ;DATA CHECK ERROR??
414 062332 001450      BEQ    290$      ;NO!!
415 062334 013737 001350 001140      MOV    RMER1I,$GDDAT ;EXPECTED STATUS
416 062342 042737 100000 001140      BIC    #DCK,$GDDAT
417 062350 013737 001350 001142      MOV    RMER1I,$BDDAT ;RECEIVED STATUS
418 062356 062716 000004      ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR
419 062362 112776 000323 000000      MOV    #323,@(SP)   ;WRITE ERROR NUMBER
420 062370 032737 004000 001366      BIT    #ECI,RMOFI   ;WAS ECC CORRECTION DISABLED??
421 062376 001021      BNE    280$      ;YES!!
422 062400 112776 000324 000000      MOV    #324,@(SP)   ;CHANGE TO RECOVERABLE ERROR
423 062406 032737 000100 001350      BIT    #ECH,RMER1I ;IS ERROR RECOVERABLE??
424 062414 001007      BNE    276$      ;NO !!
425      ;DO NOT REPORT RECOVERABLE ERROR IF READ COMMAND
426 062416 032737 000020 063340      BIT    #BIT4,510$  ;WAS THIS A READ COMMAND ??
427 062424 001406      BEQ    280$      ;NO !!
428 062426 162716 000004      SUB    #4,(SP)      ;RESTORE SP
429 062432 000410      BR    290$      ;SKIP ERROR - DATA WILL BE CORRECTED
430 062434 112776 000325 000000 276$:  MOV    #325,@(SP)   ;CHANGE TO NON RECOVERABLE
431 062442 162716 000002      280$:  SUB    #2,(SP)      ;MOVE SP TO RETURN IF ERROR
432 062446 004736      JSR    PC,@(SP)+   ;REPORT ERROR AND RETURN
433 062450 162716 000010      SUB    #10,(SP)    ;RESTORE SP TO NO ERROR
434
435 062454      290$:
436
437      ;REPORT DATA BUS PARITY ERROR IF SET, I.E., MDPE = 1
438 062454 032737 000400 001344      BIT    #MDPE,RMCS2I ;PARITY ERROR SET??
439 062462 001423      BEQ    300$      ;NO!!
440 062464 013737 001344 001140      MOV    RMCS2I,$GDDAT ;EXPECTED STATUS
441 062472 042737 000400 001140      BIC    #MDPE,$GDDAT
442 062500 013737 001344 001142      MOV    RMCS2I,$BDDAT ;RECEIVED STATUS
443 062506 062716 000004      ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR
444 062512 112776 000326 000000      MOV    #326,@(SP)   ;WRITE ERROR NUMBER
445 062520 162716 000002      SUB    #2,(SP)      ;MOVE SP TO RETURN IF ERROR
446 062524 004736      JSR    PC,@(SP)+   ;REPORT ERROR AND RETURN
447 062526 162716 000010      SUB    #10,(SP)    ;MOVE SP TO NO ERROR
448 062532 000137 062750      300$:  JMP    340$      ;SKIP WRITE STATUS CHECK
449
450 062536      310$:
451
452      ;TEST TO SEE THAT OFFSET MODE WAS RESET; REPORT ERROR IF 'OM' = 1
453 062536 032737 000001 001346      BIT    #OM,RMDSI   ;IS OFFSET ON??
454 062544 001423      BEQ    320$      ;NO
455 062546 013737 001346 001140      MOV    RMDSI,$GDDAT ;EXPECTED STATUS
456 062554 042737 000001 001140      BIC    #OM,$GDDAT
    
```

```

457 062562 013737 001346 001142      MOV      RMDSI, $BDDAT      ;RECEIVED STATUS
458 062570 062716 000004              ADD      #4, (SP)          ;MOVE SP TO USER'S ERROR CALL
459 062574 112776 000327 000000      MOVB    #327, @ (SP)      ;WRITE ERROR NUMBER IN CALL
460 062602 162716 000002              SUB      #2, (SP)          ;MOVE SP TO RETURN IF ERROR
461 062606 004736                      JSR      PC, @ (SP)+       ;REPORT ERROR AND RETURN
462 062610 162716 000010              SUB      #10, (SP)         ;MOVE SP TO NO ERROR
463 062614
464
465                                     ;TEST FOR DATA BUS PARITY ERROR; REPORT ERROR IF 'DPE' = 1
466 062614 032737 000010 001376      BIT      #DPE, RMER2I      ;DATA PARITY ERROR??
467 062622 001423                      BEQ      330$              ;NO!!
468 062624 013737 001376 001140      MOV      RMER2I, $GDDAT    ;EXPECTED STATUS
469 062632 042737 000010 001140      BIC      #DPE, $GDDAT
470 062640 013737 001376 001142      MOV      RMER2I, $BDDAT    ;RECEIVED STATUS
471 062646 062716 000004              ADD      #4, (SP)          ;MOVE SP TO USER'S ERROR CALL
472 062652 112776 000330 000000      MOVB    #330, @ (SP)      ;WRITE ERROR NUMBER
473 062660 162716 000002              SUB      #2, (SP)          ;MOVE SP TO RETURN IF ERROR
474 062664 004736                      JSR      PC, @ (SP)+       ;REPORT ERROR AND RETURN
475 062666 162716 000010              SUB      #10, (SP)         ;MOVE SP TO NO ERROR
476 062672
477
478                                     ;TEST FOR WRITE CLOCK FAILURE; REPORT ERROR IF 'WCF' = 1
479 062672 032737 000040 001350      BIT      #WCF, RMER1I      ;IS 'WCF' SET??
480 062700 001423                      BEQ      340$              ;NO!!
481 062702 013737 001350 001140      MOV      RMER1I, $GDDAT    ;EXPECTED STATUS
482 062710 042737 000040 001140      BIC      #WCF, $GDDAT
483 062716 013737 001350 001142      MOV      RMER1I, $BDDAT    ;RECEIVED STATUS
484 062724 062716 000004              ADD      #4, (SP)          ;MOVE SP TO USERS ERROR CALL
485 062730 112776 000331 000000      MOVB    #331, @ (SP)      ;WRITE ERROR NUMBER
486 062736 162716 000002              SUB      #2, (SP)          ;MOVE SP TO RETURN IF ERROR
487 062742 004736                      JSR      PC, @ (SP)+       ;REPORT ERROR AND RETURN
488 062744 162716 000010              SUB      #10, (SP)         ;MOVE SP TO NO ERROR
489 062750
490
491                                     ;REPORT 'DATA LATE' ERROR IF 'DLT' = 1
492 062750 032737 100000 001344      BIT      #DLT, RMCS2I      ;IS 'DLT' SET??
493 062756 001423                      BEQ      350$              ;NO!!
494 062760 013737 001344 001140      MOV      RMCS2I, $GDDAT    ;EXPECTED STATUS
495 062766 042737 100000 001140      BIC      #DLT, $GDDAT
496 062774 013737 001344 001142      MOV      RMCS2I, $BDDAT    ;RECEIVED STATUS
497 063002 062716 000004              ADD      #4, (SP)          ;MOVE SP TO USERS ERROR CALL
498 063006 112776 000332 000000      MOVB    #332, @ (SP)      ;WRITE ERROR NUMBER
499 063014 162716 000002              SUB      #2, (SP)          ;MOVE SP TO RETURN IF ERROR
500 063020 004736                      JSR      PC, @ (SP)+       ;REPORT ERROR AND RETURN
501 063022 162716 000010              SUB      #10, (SP)         ;MOVE SP TO NO ERROR
502 063026
503
504                                     ;LOOK FOR UNEXPECTED CHANGES IN DRIVE STATUS
504 063026 013746 001346              MOV      RMDSI, -(SP)      ;STACK DRIVE STATUS
505 063032 042716 147677              BIC      #^C<PIP!MOL!VV>, (SP) ;CLEAR DONT CARES
506 063036 022726 010100              CMP      #MOL!VV, (SP)+    ;IS DRIVE STATUS OK??
507 063042 001522                      BEQ      380$              ;YES!!
508
509                                     ;REPORT ERROR IF POSITIONING IN PROGRESS AND NO SEEK INCOMPLETE ERROR,
510                                     ;I.E. PIP = 1 AND SKI = 0
511 063044 032737 020000 001346      BIT      #PIP, RMDSI       ;IS 'PIP' SET??
512 063052 001430                      BEQ      360$              ;NO!!
513 063054 032737 040000 001376      BIT      #SKI, RMER2I      ;WAS 'SKI' ERROR REPORTED??
    
```

```

514 063062 001024      BNE      360$      ;YES-DONT REPORT PIP
515 063064 013737 001346 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
516 063072 042737 020000 001140      BIC      #PIP,$GDDAT
517 063100 013737 001346 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
518 063106 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
519 063112 112776 000333 000000      MOVB    #333,@(SP)   ;WRITE ERROR NUMBER
520 063120 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN IF ERROR
521 063124 004736      JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
522 063126 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR
523 063132 000240
524 063134      360$:
525
526      ;REPORT ERROR IF MEDIUM IS NOT ON LINE AND OPI ERROR WAS NOT
527      ;REPORTED, I.E., MOL = OPI = 0
528 063134 032737 010000 001346      BIT      #MOL,RMDSI   ;IS MEDIUM ON LINE??
529 063142 001027      BNE      370$      ;YES!!
530 063144 032737 020000 001350      BIT      #OPI,RMER1I ;WAS OPI ERROR REPORTED??
531 063152 001023      BNE      370$      ;YES!!
532 063154 013737 001346 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
533 063162 052737 010000 001140      BIS      #MOL,$GDDAT
534 063170 013737 001346 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
535 063176 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR
536 063202 112776 000334 000000      MOVB    #334,@(SP)   ;WRITE ERROR NUMBER
537 063210 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN IF ERROR
538 063214 004736      JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
539 063216 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR
540 063222      370$:
541
542      ;REPORT ERROR IF VOLUME IS NOT VALID AND 'IVC' ERROR WAS NOT
543      ;REPORTED, I.E., VV = IVC = 0
544 063222 032737 000100 001346      BIT      #VV,RMDSI    ;IS VOLUME VALID??
545 063230 001027      BNE      380$      ;YES!!
546 063232 032737 010000 001376      BIT      #IVC,RMER2I ;WAS IVC ERROR REPORTED??
547 063240 001033      BNE      390$      ;YES!!
548 063242 013737 001346 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
549 063250 052737 000100 001140      BIS      #VV,$GDDAT
550 063256 013737 001346 001142      MOV      RMDSI,$BDDAT ;RECEIVED STATUS
551 063264 062716 000004      ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
552 063270 112776 000335 000000      MOVB    #335,@(SP)   ;WRITE ERROR NUMBER
553 063276 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN IF ERROR
554 063302 004736      JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
555 063304 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR
556 063310      380$:
557
558      ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS FOUND
559 063310 062716 000004      ADD      #4,(SP)      ;MOVE SP TO ERROR CALL
560 063314 105776 000000      TSTB    @(SP)        ;ANY ERROR??
561 063320 001403      BEQ      390$      ;NO!!
562 063322 062716 000004      ADD      #4,(SP)      ;YES - MOVE SP TO ERROR RETURN
563 063326 000402      BR       400$
564 063330 162716 000004      390$: SUB      #4,(SP)      ;MOVE SP TO NO ERROR RETURN
565
566 063334 000207      400$: RTS      PC      ;RETURN TO USER
567
568 063336 000000      500$: .WORD
569 063340 000000      510$: .WORD
570
    
```

C
E

```

1      .SBTTL  STATIC DRIVE STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
4      ;STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID.  THE SUBROUTINE
5      ;CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
6      ;SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.
7
8      ;THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
9      ;IF TRUE:
10
11      ;      .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
12      ;      THAT MOL IS ASSUMED TO HAVE BEEN SET
13      ;      .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
14      ;      .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
15      ;      .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
16      ;      .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT
17
18      ;THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.
19
20      ;(1)  JSR      PC,STCDRVSTS
21      ;      BR      ???          RETURN HERE IF NO ERROR
22      ;      NOP          RETURN HERE TO REPORT AN ERROR
23      ;      ERROR      ERROR NUMBER DEFINED BY SUB
24      ;      JSR      PC,@(SP)+   GO BACK TO SUB FOR MORE ERROR CHECKS
25      ;      ???          RETURN HERE IF NO MORE ERRORS
26
27      STCDRVSTS:
28
29      ;CLEAR USER'S ERROR CALL
30      ADD      #4,(SP) ;MOVE SP TO USER'S ERROR CALL
31      CLRB    @(SP)    ;CLEAR ERROR NUMBER
32      SUB     #4,(SP)  ;MOVE SP BACK TO NO ERROR RETURN
33
34      ;SEE IF 'MOL' = 'VV' = 1, AND 'PIP' = 0
35      MOV     RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
36      BIC    #^C<PIP!MOL!VV>,(SP)
37      CMP    #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
38      BEQ    30$        ;YES!!
39
40      ;REPORT AN ERROR IF MOL = 0 AND 'OPI' = 0
41      BIT    #MOL,RMDSI  ;IS MOL ON ??
42      BNE    10$        ;YES!!
43      BIT    #OPI,RMER1I ;WAS 'OPI' SET??
44      BNE    10$        ;YES-DONT REPORT 'MOL' = 0
45      MOV    RMDSI,$GDDAT ;EXPECTED STATUS
46      BIS    #MOL,$GDDAT
47      MOV    RMDSI,$BDDAT ;RECEIVED STATUS
48      ADD    #4,(SP)    ;MOVE SP TO USER'S ERROR CALL
49      MOVB  #207,@(SP) ;WRITE ERROR NUMBER IN CALL
50      SUB    #2,(SP)    ;MOVE SP TO RETURN FOR ERROR
51      JSR    PC,@(SP)+ ;REPORT ERROR VIA USER
52      SUB    #10,(SP)  ;MOVE SP BACK TO NO ERROR RETURN
53      NOP
54
55      10$:
56      ;REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND 'IVC' = 0
57      BIT    #VV,RMDSI  ;IS 'VV' = 0??
58      BNE    20$        ;NO!!

```

```

27 063342
28
29
30 063342 062716 000004
31 063346 105076 000000
32 063352 162716 000004
33
34 063356 013746 001346
35 063362 042716 147677
36 063366 022726 010100
37 063372 001524
38
39
40 063374 032737 010000 001346
41 063402 001030
42 063404 032737 020000 001350
43 063412 001024
44 063414 013737 001346 001140
45 063422 052737 010000 001140
46 063430 013737 001346 001142
47 063436 062716 000004
48 063442 112776 000207 000000
49 063450 162716 000002
50 063454 004736
51 063456 162716 000010
52 063462 000240
53 063464
54
55
56 063464 032737 000100 001346
57 063472 001030

```

```

58 063474 032737 010000 001376 BIT #IVC,RMER2I ;WAS 'IVC' SET??
59 063502 001024 BNE 20$ ;YES-DONT REPORT 'VV' = 0
60 063504 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
61 063512 052737 000100 001346 BIS #VV,RMDSI
62 063520 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
63 063526 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
64 063532 112776 000210 000000 MOVVB #210,@(SP) ;WRITE ERROR NUMBER IN CALL
65 063540 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
66 063544 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
67 063546 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
68 063552 000240 NOP
69 063554 20$:
70
71 ;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND 'SKI' = 0
72 063554 032737 020000 001346 BIT #PIP,RMDSI ;IS DRIVE OFF CYLINDER??
73 063562 001430 BEQ 30$ ;NO!!
74 063564 032737 040000 001376 BIT #SKI,RMER2I ;WAS 'SKI' SET??
75 063572 001024 BNE 30$ ;YES-DONT REPORT 'PIP' = 1
76 063574 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
77 063602 042737 020000 001140 BIC #PIP,$GDDAT
78 063610 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
79 063616 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
80 063622 112776 000211 000000 MOVVB #211,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
81 063630 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
82 063634 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
83 063636 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR RETURN
84 063642 000240 NOP
85 063644 30$:
86
87 ;SEE IF 'SKI' = 'DVC' = 0
88 063644 013746 001376 MOV RMER2I,-(SP) ;PUT ERROR REG 2 ON STACK
89 063650 042726 137577 BIC #^C<SKI!DVC>,(SP)+
90 063654 001460 BEQ 60$ ;BRANCH IF NO ERROR
91 063656 40$:
92
93 ;REPORT AN ERROR IF THERE IS A DEVICE FAULT
94 063656 032737 000200 001376 BIT #DVC,RMER2I ;ANY DEVICE FAULT??
95 063664 001424 BEQ 50$ ;NO!!
96 063666 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
97 063674 042737 000200 001140 BIC #DVC,$GDDAT
98 063702 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
99 063710 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S CALL
100 063714 112776 000212 000000 MOVVB #212,@(SP) ;WRITE NUMBER OF ERROR IN CALL
101 063722 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
102 063726 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
103 063730 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
104 063734 000240 NOP
105 063736 50$:
106
107 ;REPORT AN ERROR IF 'SKI' = 1
108 063736 032737 040000 001376 BIT #SKI,RMER2I ;IS THERE A SEEK INCOMPLETE ERROR
109 063744 001424 BEQ 60$ ;NO!!
110 063746 013737 001376 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
111 063754 042737 040000 001140 BIC #SKI,$GDDAT
112 063762 013737 001376 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
113 063770 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
114 063774 112776 000213 000000 MOVVB #213,@(SP) ;WRITE ERROR NUMBER IN USER'S ERROR CALL
    
```



```
115 064002 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
116 064006 004736              JSR      PC,@(SP)+   ;REPORT ERROR VIA USER
117 064010 162716 000010      SUB      #10,(SP)    ;MOVE SP BACK TO NO ERROR
118 064014 000240              NOP
119 064016              60$:
120
121              ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
122 064016 062716 000004      ADD      #4,(SP)    ;MOVE SP TO USER'S ERROR CALL
123 064022 105776 000000      TSTB    @(SP)       ;WAS AN ERROR DETECTED??
124 064026 001403              BEQ      70$        ;NO!!
125 064030 062716 000004      ADD      #4,(SP)    ;YES - MOVE SP TO USER'S ERROR RETURN
126 064034 000402              BR       80$
127 064036 162716 000004      70$: SUB      #4,(SP)    ;NO - MOVE SP TO NO ERROR RETURN
128 064042 000240      80$: NOP
129 064044 000207      RTS      PC        ;RETURN TO USER
130
```

```

1      .SBTTL  STOP AND SHUTDOWN SUBROUTINES
2
3 064046 STOP:
4
5      ;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT
6 064046 012746 000140      MOV    #PR3,-(SP)      ;;PUT NEW PS ON STACK
   064052 012746 064060      MOV    #64$,-(SP)     ;;PUT NEW PC ON STACK
   064056 000002      RTI      ;;POP NEW PC AND PS
   064060
7 064060 000240      64$:      NOP
8
9      ;RAISE PRIORITY TO INHIBIT CONSOLE INTERRUPT
10 064062 012746 000300     MOV    #PR6,-(SP)     ;;PUT NEW PS ON STACK
   064066 012746 064074     MOV    #65$,-(SP)    ;;PUT NEW PC ON STACK
   064072 000002      RTI      ;;POP NEW PC AND PS
   064074
11 064074 000207      65$:      RTS      PC      ;CONTINUE
12
13 064076 005737 001326     SHUT:  TST    CTLFG      ;HALT ?
14 064102 001002      BNE    5$             ;BR IF YES
15 064104 000137 007764     JMP    READY          ;CONTINUE
16 064110 005737 000042     5$:    TST    @#42      ;ANY MONITOR PRESENT ?
17 064114 001015      BNE    10$           ;BR IF YES
18 064116 104401 064124     TYPE   ,65$          ;;TYPE ASCIZ STRING
   064122 000410      BR     64$          ;;GET OVER THE ASCIZ
   064144      ;;65$: .ASCIZ <CRLF><07>/TEST HALTED/<CRLF>
19 064144 000137 005420     64$:      JMP    START          ;GO TO START
20 064150 000137 037460     10$:     JMP    $EOP          ;RETURN CONTROL TO MONITOR
21
22 064154 012737 177777 001326 SHUT2:  MOV    #-1,CTLFG      ;SET THE CONTROL-C FLAG
23 064162 000002      RTI      ;EXIT FROM INTERRUPT
24

```

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
    
```

```

064164
064164 010046
064166 010146
064170 010246
064172 010346
064174 010446
064176 010546
064200 016646 000022
064204 016646 000022
064210 016646 000022
064214 016646 000022
064220 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
    
```

```

064222
064222 012666 000022
064226 012666 000022
064232 012666 000022
064236 012666 000022
064242 012605
064244 012604
064246 012603
064250 012602
064252 012601
064254 012600
064256 000002
    
```

2

.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
    
```

```

064260 010146
064262 016601 000006
064266 000261
    
```

```

$TYPBN: MOV R1,-(SP)      ;;SAVE R1 ON THE STACK
        MOV 6(SP),R1      ;;GET THE INPUT NUMBER
        SEC              ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
    
```

```

064270 112737 000060 064332 1$:   MOVB   #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
064276 006101                ROL    R1             ;;GET THIS BIT
064300 001406                BEQ    2$             ;;DONE?
064302 105537 064332        ADCB   $BIN             ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
064306 104401 064332        TYPE   , $BIN          ;;GO TYPE THIS BIT
064312 000241                CLC                    ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
064314 000765                BR     1$             ;;GO DO THE NEXT BIT
064316 012601                MOV    (SP)+,R1       ;;POP THE STACK INTO R1
064320 016666 000002 000004 2$:   MOV    2(SP),4(SP)    ;;ADJUST THE STACK
064326 012616                MOV    (SP)+,(SP)
064330 000002                RTI                    ;;RETURN TO USER
064332 000      000        $BIN:  .BYTE  0,0          ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
3      .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
    
```

```

064334 010046                $TYPDS: MOV    R0,-(SP)      ;;PUSH R0 ON STACK
064336 010146                MOV    R1,-(SP)      ;;PUSH R1 ON STACK
064340 010246                MOV    R2,-(SP)      ;;PUSH R2 ON STACK
064342 010346                MOV    R3,-(SP)      ;;PUSH R3 ON STACK
064344 010546                MOV    R5,-(SP)      ;;PUSH R5 ON STACK
064346 012746 020200        MOV    #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
064352 016605 000020        MOV    20(SP),R5     ;;GET THE INPUT NUMBER
064356 100004                BPL    1$             ;;BR IF INPUT IS POS.
064360 005405                NEG    R5             ;;MAKE THE BINARY NUMBER POS.
064362 112766 000055 000001 1$:   MOVB   #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
064370 005000                CLR    R0             ;;ZERO THE CONSTANTS INDEX
064372 012703 064550        MOV    #$DBLK,R3     ;;SETUP THE OUTPUT POINTER
064376 112723 000040        MOVB   #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
064402 005002                2$:   CLR    R2             ;;CLEAR THE BCD NUMBER
064404 016001 064540        MOV    $DTBL(R0),R1  ;;GET THE CONSTANT
064410 160105                3$:   SUB    R1,R5         ;;FORM THIS BCD DIGIT
064412 002402                BLT    4$             ;;BR IF DONE
064414 005202                INC    R2             ;;INCREASE THE BCD DIGIT BY 1
064416 000774                BR     3$
064420 060105                4$:   ADD    R1,R5         ;;ADD BACK THE CONSTANT
064422 005702                TST    R2             ;;CHECK IF BCD DIGIT=0
064424 001002                BNE    5$             ;;FALL THROUGH IF 0
064426 105716                TSTB   (SP)          ;;STILL DOING LEADING 0'S?
064430 100407                BMI    7$             ;;BR IF YES
064432 106316                5$:   ASLB   (SP)          ;;MSD?
064434 103003                BCC    6$             ;;BR IF NO
064436 116663 000001 177777 6$:   MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
064444 052702 000060        BIS    #'0,R2        ;;MAKE THE BCD DIGIT ASCII
064450 052702 000040        7$:   BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
064454 110223                MOVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
064456 005720                TST    (R0)+        ;;JUST INCREMENTING
064460 020027 000010        CMP    R0,#10        ;;CHECK THE TABLE INDEX
    
```

```

064464 002746          BLT      2$          ;;GO DO THE NEXT DIGIT
064466 003002          BGT      8$          ;;GO TO EXIT
064470 010502          MOV      R5,R2        ;;GET THE LSD
064472 000764          BR       6$          ;;GO CHANGE TO ASCII
064474 105726          8$:    TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
064476 100003          BPL      9$          ;;BR IF NO
064500 116663 177777 177776  MOVB    -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
064506 105013          9$:    CLRB     (R3)      ;;SET THE TERMINATOR
064510 012605          MOV      (SP)+,R5    ;;POP STACK INTO R5
064512 012603          MOV      (SP)+,R3    ;;POP STACK INTO R3
064514 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
064516 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
064520 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
064522 104401 064550  TYPE     $DBLK      ;;NOW TYPE THE NUMBER
064526 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
064534 012616          MOV      (SP)+,(SP)
064536 000002          RTI                    ;;RETURN TO USER
064540 023420          $DTBL: 10000.
064542 001750          1000.
064544 000144          100.
064546 000012          10.
064550          $DBLK: .BLKW 4
          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
    
```

4

```

;*****
;*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
    
```

```

064560 017646 000000          $TYPOS: MOV      @ (SP),-(SP)  ;;PICKUP THE MODE
064564 116637 000001 065003  MOVB    1(SP),$OFILL  ;;LOAD ZERO FILL SWITCH
064572 112637 065005          MOVB    (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
064576 062716 000002          ADD     #2,(SP)      ;;ADJUST RETURN ADDRESS
064602 000406          BR      $TYPON
064604 112737 000001 065003  $TYPOC: MOVB    #1,$OFILL  ;;SET THE ZERO FILL SWITCH
064612 112737 000006 065005  MOVB    #6,$OMODE+1  ;;SET FOR SIX(6) DIGITS
064620 112737 000005 065002  $TYPON: MOVB    #5,$OCNT  ;;SET THE ITERATION COUNT
064626 010346          MOV     R3,-(SP)    ;;SAVE R3
064630 010446          MOV     R4,-(SP)    ;;SAVE R4
    
```

```

064632 010546          MOV      R5,-(SP)      ;;SAVE R5
064634 113704 065005  MOVVB   $OMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
064640 005404          NEG      R4
064642 062704 000006  ADD     #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
064646 110437 065004  MOVVB   R4,$OMODE   ;;SAVE IT FOR USE
064652 113704 065003  MOVVB   $OFILL,R4   ;;GET THE ZERO FILL SWITCH
064656 016605 000012  MOV     12(SP),R5   ;;PICKUP THE INPUT NUMBER
064662 005003          CLR     R3         ;;CLEAR THE OUTPUT WORD
064664 006105          1$:    ROL     R5         ;;ROTATE MSB INTO 'C'
064666 000404          BR     3$         ;;GO DO MSB
064670 006105          2$:    ROL     R5         ;;FORM THIS DIGIT
064672 006105          ROL     R5
064674 006105          ROL     R5
064676 010503          MOV     R5,R3
064700 006103          3$:    ROL     R3         ;;GET LSB OF THIS DIGIT
064702 105337 065004  DECB   $OMODE      ;;TYPE THIS DIGIT?
064706 100016          BPL    7$         ;;BR IF NO
064710 042703 177770  BIC    #177770,R3  ;;GET RID OF JUNK
064714 001002          BNE    4$         ;;TEST FOR 0
064716 005704          TST   R4         ;;SUPPRESS THIS 0?
064720 001403          BEQ   5$         ;;BR IF YES
064722 005204          4$:    INC   R4         ;;DON'T SUPPRESS ANYMORE 0'S
064724 052703 000060  BIS   #'0,R3      ;;MAKE THIS DIGIT ASCII
064730 052703 000040  5$:    BIS   #' ,R3    ;;MAKE ASCII IF NOT ALREADY
064734 110337 065000  MOVVB   R3,8$     ;;SAVE FOR TYPING
064740 104401 065000  TYPE   ,8$       ;;GO TYPE THIS DIGIT
064744 105337 065002  7$:    DECB   $OCNT   ;;COUNT BY 1
064750 003347          BGT   2$         ;;BR IF MORE TO DO
064752 002402          BLT   6$         ;;BR IF DONE
064754 005204          INC   R4         ;;INSURE LAST DIGIT ISN'T A BLANK
064756 000744          BR    2$         ;;GO DO THE LAST DIGIT
064760 012605          6$:    MOV   (SP)+,R5  ;;RESTORE R5
064762 012604          MOV   (SP)+,R4    ;;RESTORE R4
064764 012603          MOV   (SP)+,R3    ;;RESTORE R3
064766 016666 000002 000004  MOV   2(SP),4(SP) ;;SET THE STACK FOR RETURNING
064774 012616          MOV   (SP)+,(SP)
064776 000002          RTI
065000          8$:    .BYTE 0      ;;RETURN
065001          .BYTE 0      ;;STORAGE FOR ASCII DIGIT
065002          .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
065003          .BYTE 0      ;;OCTAL DIGIT COUNTER
065004 000000          .WORD 0       ;;ZERO FILL SWITCH
                    .SBTTL TYPE ROUTINE  ;;NUMBER OF DIGITS TO TYPE

```

5

```

*****
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;*      TYPE ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;*      TYPE
;*      MESADR

```

TYPE ROUTINE

```

;*
065006 105737 001173 $TYPE: TSTB $TPFLG ::IS THERE A TERMINAL?
065012 100002 BPL 1$ ::BR IF YES
065014 000000 HALT ::HALT HERE IF NO TERMINAL
065016 000430 BR 3$ ::LEAVE
065020 010046 1$: MOV R0,-(SP) ::SAVE R0
065022 017600 000002 MOV @2(SP),R0 ::GET ADDRESS OF ASCIZ STRING
065026 122737 000001 001242 CMPB #APTENV,$ENV ::RUNNING IN APT MODE
065034 001011 BNE 62$ ::NO,GO CHECK FOR APT CONSOLE
065036 132737 000100 001243 BITB #APTSPOOL,$ENVM ::SPOOL MESSAGE TO APT
065044 001405 BEQ 62$ ::NO,GO CHECK FOR CONSOLE
065046 010037 065056 MOV R0,61$ ::SETUP MESSAGE ADDRESS FOR APT
065052 004737 067770 JSR PC,$ATY3 ::SPOOL MESSAGE TO APT
065056 000000 61$: .WORD 0 ::MESSAGE ADDRESS
065060 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ::APT CONSOLE SUPPRESSED
065066 001003 BNE 60$ ::YES,SKIP TYPE OUT
065070 112046 2$: MOVB (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
065072 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
065074 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
065076 012600 60$: MOV (SP)+,R0 ::RESTORE R0
065100 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
065104 000002 RTI ::RETURN
065106 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
065112 001430 BEQ 8$
065114 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
065120 001006 BNE 5$
065122 005726 TST (SP)+ ::POP <CR><LF> EQUIV
065124 104401 TYPE ::TYPE A CR AND LF
065126 001217 $CRLF
065130 105037 065336 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
065134 000755 BR 2$ ::GET NEXT CHARACTER
065136 004737 065220 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
065142 123726 001172 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
065146 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
065150 013746 001170 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
::AND THE NULL CHAR.
065154 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
065160 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
065162 004737 065220 JSR PC,$TYPEC ::GO TYPE A NULL
065166 105337 065336 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
065172 000770 BR 7$ ::LOOP

```

;HORIZONTAL TAB PROCESSOR

```

065174 112716 000040 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
065200 004737 065220 9$: JSR PC,$TYPEC ::TYPE A SPACE
065204 132737 000007 065336 BITB #7,$CHARCNT ::BRANCH IF NOT AT
065212 001372 BNE 9$ ::TAB STOP
065214 005726 TST (SP)+ ::POP SPACE OFF STACK
065216 000724 BR 2$ ::GET NEXT CHARACTER
065220 $TYPEC:
065220 105777 113734 TSTB @$TKS ::CHAR IN KYBD BUFFER?
065224 100022 BPL 10$ ::BR IF NOT
065226 017746 113730 MOV @$TKB,-(SP) ::GET CHAR
065232 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS
065236 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF

```

```

065242 001012          BNE      102$      ;;BR IF NOT
065244          101$:      TSTB     @STKS      ;;WAIT FOR CHAR
065244 105777 113710    BPL      101$
065250 100375          MOVB     @STKB,(SP)  ;;GET CHAR
065252 117716 113704    BIC      #177600,(SP)  ;;STRIP IT
065256 042716 177600    CMPB     #$XON,(SP)  ;;WAS IT XON?
065262 122716 000021    BNE      101$      ;;BR IF NOT
065266 001366          102$:      TST      (SP)+      ;;FIX STACK
065270 005726          10$:      TSTB     @STPS      ;;WAIT UNTIL PRINTER IS READY
065272          BPL      10$
065272 105777 113666    MOVB     2(SP),@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
065276 100375          CMPB     #CR,2(SP)  ;;IS CHARACTER A CARRIAGE RETURN?
065300 116677 000002 113660 BNE      1$        ;;BRANCH IF NO
065306 122766 000015 000002 CLRB     $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
065314 001003          BR      $TYPEX     ;;EXIT
065316 105037 065336    CMPB     #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
065322 000406          BEQ     $TYPEX     ;;BRANCH IF YES
065324 122766 000012 000002 INCB     (PC)+      ;;COUNT THE CHARACTER
065332 001402          $CHARCNT: .WORD  0  ;;CHARACTER COUNT STORAGE
065334 105227          $TYPEX: RTS      PC
065336 000000
065340 000207
    
```

6

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW11=1      INHIBIT ITERATIONS
*SW09=1      LOOP ON ERROR
*SW08=1      LOOP ON TEST IN SWR<7:0>
*CALL
*          SCOPE          ;;SCOPE=IOT
    
```

```

065342          $SCOPE:      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
065342 104410          JSR      PC,STOP
065344 004737 064046    1$:      BIT      #BIT14,@SWR  ;;LOOP ON PRESENT TEST?
065350 032777 040000 113576 BNE      $OVER      ;;YES IF SW14=1
065356 001131          ;#####START OF CODE FOR THE XOR TESTER#####
065360 000416          $XTSTR: BR      6$
065362 013746 000004          MOV     @#ERRVEC,-(SP)  ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
065366 012737 065406 000004 MOV     #5$,@#ERRVEC  ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
065374 005737 177060          TST     @#177060  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
065400 012637 000004          MOV     (SP)+,@#ERRVEC  ;;SET FOR TIMEOUT
065404 000500          BR      $SVLAD     ;;TIME OUT ON XOR?
065406 022626          5$:      CMP     (SP)+,(SP)+  ;;RESTORE THE ERROR VECTOR
065410 012637 000004          MOV     (SP)+,@#ERRVEC  ;;GO TO THE NEXT TEST
065414 000440          BR      7$        ;;CLEAR THE STACK AFTER A TIME OUT
065416          6$:;#####END OF CODE FOR THE XOR TESTER#####
065416 032777 000400 113530 BIT      #BIT08,@SWR  ;;RESTORE THE ERROR VECTOR
065424 001421          BEQ     2$        ;;LOOP ON THE PRESENT TEST
                                ;;LOOP ON SPEC. TEST?
                                ;;BR IF NO
    
```


SCOPE HANDLER ROUTINE

065426	005046			CLR	-(SP)	::CLEAR A TEMP. LOCATION	
065430	117716	113520		MOVB	@SWR,(SP)	::PICKUP THE DESIRED TEST NUMBER	
065434	001414			BEQ	8\$::BRANCH IF BAD TEST NUMBER IN SWR	
065436	022716	000027		CMP	#27,(SP)	::CHECK THE NUMBER IN THE SWR	
065442	002411			BLT	8\$::BRANCH IF TEST NUMBER IS OUT OF RANGE	
065444	011637	001116		MOV	(SP), \$TSTNM	::UPDATE THE TEST NUMBER	
065450	005316			DEC	(SP)	::BACKUP BY ONE	
065452	006316			ASL	(SP)	::SCALE THE TEST NUMBER AS AN INDEX	
065454	062716	065660		ADD	#\$SW08TBL,(SP)	::FORM THE ADDRESS OF TEST POINTER	
065460	013637	001122		MOV	@(SP)+,\$LPADR	::SET LOOP ADDRESS TO DESIRED TEST	
065464	000466			BR	\$OVER	::GO LOOP ON THE TEST	
065466	005726		8\$:	TST	(SP)+	::CLEAN THE BAD TEST NUMBER OFF OF THE STACK	
065470	105737	001117	2\$:	TSTB	\$ERFLG	::HAS AN ERROR OCCURRED?	
065474	001421			BEQ	3\$::BR IF NO	
065476	123737	001131	001117	CMPB	\$ERMAX,\$ERFLG	::MAX. ERRORS FOR THIS TEST OCCURRED?	
065504	101015			BHI	3\$::BR IF NO	
065506	032777	001000	113440	BIT	#BIT09,@SWR	::LOOP ON ERROR?	
065514	001404			BEQ	4\$::BR IF NO	
065516	013737	001124	001122	7\$:	MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
065524	000446			BR	\$OVER		
065526	105037	001117	4\$:	CLRB	\$ERFLG	::ZERO THE ERROR FLAG	
065532	005037	001206		CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE	
065536	000415			BR	1\$::ESCAPE TO THE NEXT TEST	
065540	032777	004000	113406	3\$:	BIT	#BIT11,@SWR	::INHIBIT ITERATIONS?
065546	001011			BNE	1\$::BR IF YES	
065550	005737	001230		TST	\$PASS	::IF FIRST PASS OF PROGRAM	
065554	001406			BEQ	1\$::INHIBIT ITERATIONS	
065556	005237	001120		INC	\$ICNT	::INCREMENT ITERATION COUNT	
065562	023737	001206	001120	CMP	\$TIMES,\$ICNT	::CHECK THE NUMBER OF ITERATIONS MADE	
065570	002024			BGE	\$OVER	::BR IF MORE ITERATION REQUIRED	
065572	012737	000001	001120	1\$:	MOV	#1,\$ICNT	::REINITIALIZE THE ITERATION COUNTER
065600	013737	065656	001206	MOV	\$MXCNT,\$TIMES	::SET NUMBER OF ITERATIONS TO DO	
065606	105237	001116		\$SVLAD: INCB	\$TSTNM	::COUNT TEST NUMBERS	
065612	113737	001116	001226	MOVB	\$TSTNM,\$TESTN	::SET TEST NUMBER IN APT MAILBOX	
065620	011637	001122		MOV	(SP),\$LPADR	::SAVE SCOPE LOOP ADDRESS	
065624	011637	001124		MOV	(SP),\$LPERR	::SAVE ERROR LOOP ADDRESS	
065630	005037	001210		CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS	
065634	112737	000001	001131	MOVB	#1,\$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST	
065642	013777	001116	113306	\$OVER: MOV	\$TSTNM,@DISPLAY	::DISPLAY TEST NUMBER	
065650	013716	001122		MOV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS	
065654	000002			RTI		::FIXES PS	
065656	000012			\$MXCNT: 10.		::MAX. NUMBER OF ITERATIONS	
065660				\$SW08TBL:			
065660	010132			.WORD	TST1+2	::STARTING ADDRESS OF TEST 1	
065662	010316			.WORD	TST2+2	::STARTING ADDRESS OF TEST 2	
065664	010502			.WORD	TST3+2	::STARTING ADDRESS OF TEST 3	
065666	010704			.WORD	TST4+2	::STARTING ADDRESS OF TEST 4	
065670	011712			.WORD	TST5+2	::STARTING ADDRESS OF TEST 5	
065672	012670			.WORD	TST6+2	::STARTING ADDRESS OF TEST 6	
065674	014064			.WORD	TST7+2	::STARTING ADDRESS OF TEST 7	
065676	015072			.WORD	TST10+2	::STARTING ADDRESS OF TEST 10	
065700	016050			.WORD	TST11+2	::STARTING ADDRESS OF TEST 11	
065702	017246			.WORD	TST12+2	::STARTING ADDRESS OF TEST 12	
065704	020252			.WORD	TST13+2	::STARTING ADDRESS OF TEST 13	
065706	021476			.WORD	TST14+2	::STARTING ADDRESS OF TEST 14	
065710	022704			.WORD	TST15+2	::STARTING ADDRESS OF TEST 15	
065712	023726			.WORD	TST16+2	::STARTING ADDRESS OF TEST 16	

065714 025574
065716 027200
065720 030220
065722 031136
065724 032174
065726 033200
065730 034206
065732 035214
065734 036242

.WORD TST17+2 ::STARTING ADDRESS OF TEST 17
.WORD TST20+2 ::STARTING ADDRESS OF TEST 20
.WORD TST21+2 ::STARTING ADDRESS OF TEST 21
.WORD TST22+2 ::STARTING ADDRESS OF TEST 22
.WORD TST23+2 ::STARTING ADDRESS OF TEST 23
.WORD TST24+2 ::STARTING ADDRESS OF TEST 24
.WORD TST25+2 ::STARTING ADDRESS OF TEST 25
.WORD TST26+2 ::STARTING ADDRESS OF TEST 26
.WORD TST27+2 ::STARTING ADDRESS OF TEST 27

7

.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

065736
065736 104410
065740 105237 001117
065744 001775
065746 013777 001116 113202
065754 032777 002000 113172
065762 001402
065764 104401 001212
065770 005237 001126
065774 011637 001132
066000 162737 000002 001132
066006 117737 113120 001130
066014 032777 020000 113132
066022 001004
066024 004737 037670
066030 104401 001217
066034
066034 122737 000001 001242
066042 001007
066044 113737 001130 066056
066052 004737 070000
066056 000
066057 000
066060 000777
066062 005777 113066
066066 100002
066070 000000
066072 104410
066074 032777 001000 113052
066102 001402
066104 013716 001124
066110 005737 001210
066114 001402
066116 013716 001210

\$ERROR:
7\$: CKSWR ::TEST FOR CHANGE IN SOFT-SWR
 INCB \$ERFLG ::SET THE ERROR FLAG
 BEQ 7\$::DON'T LET THE FLAG GO TO ZERO
 MOV \$TSTNM,@DISPLAY ::DISPLAY TEST NUMBER AND ERROR FLAG
 BIT #BIT10,@SWR ::BELL ON ERROR?
 BEQ 1\$::NO - SKIP
 TYPE ,\$BELL ::RING BELL
1\$: INC \$ERTTL ::COUNT THE NUMBER OF ERRORS
 MOV (SP),\$ERRPC ::GET ADDRESS OF ERROR INSTRUCTION
 SUB #2,\$ERRPC
 MOVB @\$ERRPC,\$ITEMB ::STRIP AND SAVE THE ERROR ITEM CODE
 BIT #BIT13,@SWR ::SKIP TYPEOUT IF SET
 BNE 20\$::SKIP TYPEOUTS
 JSR PC,ERRTP ::GO TO USER ERROR ROUTINE
 TYPE ,\$CRLF
20\$: CMPB #APTENV,\$ENV ::RUNNING IN APT MODE
 BNE 2\$::NO,SKIP APT ERROR REPORT
 MOVB \$ITEMB,21\$::SET ITEM NUMBER AS ERROR NUMBER
 JSR PC,\$ATY4 ::REPORT FATAL ERROR TO APT
21\$: .BYTE 0
 .BYTE 0
22\$: BR 22\$::APT ERROR LOOP
2\$: TST @SWR ::HALT ON ERROR
 BPL 3\$::SKIP IF CONTINUE
 HALT ::HALT ON ERROR!
3\$: CKSWR ::TEST FOR CHANGE IN SOFT-SWR
 BIT #BIT09,@SWR ::LOOP ON ERROR SWITCH SET?
 BEQ 4\$::BR IF NO
 MOV \$LPERR,(SP) ::FUDGE RETURN FOR LOOPING
4\$: TST \$ESCAPE ::CHECK FOR AN ESCAPE ADDRESS
 BEQ 5\$::BR IF NONE
 MOV \$ESCAPE,(SP) ::FUDGE RETURN ADDRESS FOR ESCAPE

ERROR HANDLER ROUTINE

```

066122      066122 022737 037650 000042 5$:      CMP      #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
066130      066130 001001                BNE      6$                ;;BRANCH IF NO
066132      066132 000000                HALT                    ;;YES
066134      066134 000002                RTI                      ;;RETURN
8          .SBTTL  TTY INPUT ROUTINE

;*****
;ENABL  LSB
066136      066136 000000  $TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
066140      066140 000000  $TKQIN: .WORD 0          ;;INPUT POINTER
066142      066142 C00000  $TKQOUT: .WORD 0         ;;OUTPUT POINTER
066144      066144 066145  $TKQSR: .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
;
066146      066146 005037 066136  $TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
066152      066152 012737 066144 066140  MOV      #$TKQSR,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
066160      066160 013737 066140 066142  MOV      $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
066166      066166 012737 066216 000060  MOV      #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
066174      066174 012737 000200 000062  MOV      #200,@#TKVEC+2  ;;'BR' LEVEL 4
066202      066202 005777 112754          TST      @$TKB           ;;CLEAR DONE FLAG
066206      066206 012777 000100 112744  MOV      #100,@$TKS     ;;ENABLE TTY KEYBOARD INTERRUPT
066214      066214 000207          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)
;
066216      066216 117746 112740  $TKSRV: MOVB   @$TKB,-(SP)  ;;PICKUP THE CHARACTER
066222      066222 042716 177600          BIC      #^C177,(SP)    ;;STRIP THE JUNK
066226      066226 021627 000003          CMP      (SP),#3        ;;IS IT A CONTROL C?
066232      066232 001007          BNE      1$            ;;BRANCH IF NO
066234      066234 104401 067332          TYPE    ,SCNTLC        ;;TYPE A CONTROL-C (^C)
066240      066240 004737 066146          JSR     PC,$TKINT      ;;INIT THE KEYBOARD
066244      066244 005726          TST     (SP)+          ;;CLEAN UP STACK
066246      066246 000137 064154          JMP     SHUT2          ;;CONTROL C RESTART
066252      066252 021627 000007 1$:      CMP      (SP),#7        ;;IS IT A CONTROL G?
066256      066256 001004          BNE     2$            ;;BRANCH IF NO
066260      066260 022737 000176 001154  CMP     #SWREG,SWR     ;;IS SOFT-SWR SELECTED?
066266      066266 001500          BEQ     6$            ;;GO TO SWR CHANGE

066270      066270 022737 000001 066136 2$:      CMP     #1,$TKCNT     ;;IS THE QUEUE FULL?
066276      066276 001004          BNE     3$            ;;BRANCH IF NO

```

```

066300 104401 001212      TYPE      ,SBELL      ;;RING THE TTY BELL
066304 005726            TST        (SP)+      ;;CLEAN CHARACTER OFF OF STACK
066306 000451            BR         5$         ;;EXIT
066310 021627 000023      3$:      CMP        (SP),#23      ;;IS IT A CONTROL-S?
066314 001021            BNE        32$         ;;BRANCH IF NO
066316 005077 112636      CLR        @STKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
066322 005726            TST        (SP)+      ;;CLEAN CHAR OFF STACK
066324 105777 112630      31$:     TSTB       @STKS      ;;WAIT FOR A CHAR
066330 100375            BPL        31$         ;;LOOP UNTIL ITS THERE
066332 117746 112624      MOVB       @STKB,-(SP) ;;GET THE CHARACTER
066336 042716 177600      BIC        #^C177,(SP) ;;MAKE IT 7-BIT ASCII
066342 022627 000021      CMP        (SP)+,#21   ;;IS IT A CONTROL-Q?
066346 001366            BNE        31$         ;;BRANCH IF NO
066350 012777 000100 112602 MOV        #100,@STKS  ;;REENABLE TTY KEYBOARD INTERRUPTS
066356 000002            RTI                          ;;RETURN
066360 005237 066136      32$:     INC        $TKCNT      ;;COUNT THIS CHARACTER
066364 021627 000140      CMP        (SP),#140   ;;IS IT UPPER CASE?
066370 002405            BLT        4$         ;;BRANCH IF YES
066372 021627 000175      CMP        (SP),#175   ;;IS IT A SPECIAL CHAR?
066376 003002            BGT        4$         ;;BRANCH IF YES
066400 042716 000040      BIC        #40,(SP)    ;;MAKE IT UPPER CASE
066404 112677 177530      4$:     MOVB       (SP)+,@$TKQIN ;;AND PUT IT IN QUEUE
066410 005237 066140      INC        $TKQIN      ;;UPDATE THE POINTER
066414 023727 066140 066145 CMP        $TKQIN,$$TKQEND ;;GO OFF THE END?
066422 001003            BNE        5$         ;;BRANCH IF NO
066424 012737 066144 066140 MOV        $$TKQSR,$$TKQIN ;;RESET THE POINTER
066432 000002            5$:     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.
  
```

```

066434 022737 000176 001154 $CKSWR:  CMP        #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
066442 001124            BNE        15$         ;;EXIT IF NOT
066444 105777 112510      TSTB       @STKS      ;;IS A CHAR WAITING?
066450 100121            BPL        15$         ;;IF NOT, EXIT
066452 117746 112504      MOVB       @STKB,-(SP) ;;YES
066456 042716 177600      BIC        #^C177,(SP) ;;MAKE IT 7-BIT ASCII
066462 021627 000007      CMP        (SP),#7     ;;IS IT A CONTROL-G?
066466 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.
  
```

```

066470 123727 001150 000001 6$:      CMPB       $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
066476 001674            BEQ        2$         ;;BRANCH IF YES
066500 005726            TST        (SP)+      ;;CLEAR CONTROL-G OFF STACK
066502 004737 066146      JSR        PC,$$TKINT  ;;FLUSH THE TTY INPUT QUEUE
066506 005077 112446      CLR        @STKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
066512 112737 000001 001151 MOVB       #1,$$INTAG   ;;SET INTERRUPT MODE INDICATOR

066520 104401 067344      TYPE      ,SCNTLG     ;;ECHO THE CONTROL-G (^G)
066524 104401 067351      $GTSWR:  TYPE      ,SMSWR      ;;TYPE CURRENT CONTENTS
066530 013746 000176      MOV        SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
  
```

066534	104402			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
066536	104401	067362		TYPE	,SMNEW	::PROMPT FOR NEW SWR
066542	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
066544	005046			CLR	-(SP)	::THE NEW SWR
066546	105777	112406	7\$:	TSTB	@\$TKS	::CHAR THERE?
066552	100375			BPL	7\$::IF NOT TRY AGAIN
066554	117746	112402		MOVB	@\$TKB, -(SP)	::PICK UP CHAR
066560	042716	177600		BIC	#^C177, (SP)	::MAKE IT 7-BIT ASCII
066564	021627	000003		CMP	(SP), #3	::IS IT A CONTROL-C?
066570	001015			BNE	9\$::BRANCH IF NOT
066572	104401	067332		TYPE	, \$CNTLC	::YES, ECHO CONTROL-C (^C)
066576	062706	000006		ADD	#6, SP	::CLEAN UP STACK
066602	123727	001151	000001	CMPB	\$INTAG, #1	::REENABLE TTY KEYBOARD INTERRUPTS?
066610	001003			BNE	8\$::BRANCH IF NO
066612	012777	000100	112340	MOV	#100, @\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
066620	000137	064154	8\$:	JMP	SHUT2	::CONTROL-C RESTART
066624	021627	000025	9\$:	CMP	(SP), #25	::IS IT A CONTROL-U?
066630	001005			BNE	10\$::BRANCH IF NOT
066632	104401	067337		TYPE	, \$CNTLU	::YES, ECHO CONTROL-U (^U)
066636	062706	000006	20\$:	ADD	#6, SP	::IGNORE PREVIOUS INPUT
066642	000737			BR	19\$::LET'S TRY IT AGAIN
066644	021627	000015	10\$:	CMP	(SP), #15	::IS IT A <CR>?
066650	001022			BNE	16\$::BRANCH IF NO
066652	005766	000004		TST	4, (SP)	::YES, IS IT THE FIRST CHAR?
066656	001403			BEQ	11\$::BRANCH IF YES
066660	016677	000002	112266	MOV	2(SP), @SWR	::SAVE NEW SWR
066666	062706	000006	11\$:	ADD	#6, SP	::CLEAR UP STACK
066672	104401	001217	14\$:	TYPE	, \$CRLF	::ECHO <CR> AND <LF>
066676	123727	001151	000001	CMPB	\$INTAG, #1	::RE-ENABLE TTY KBD INTERRUPTS?
066704	001003			BNE	15\$::BRANCH IF NOT
066706	012777	000100	112244	MOV	#100, @\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
066714	000002		15\$:	RTI		::RETURN
066716	004737	065220	16\$:	JSR	PC, \$TYPEC	::ECHO CHAR
066722	021627	000060		CMP	(SP), #60	::CHAR < 0?
066726	002420			BLT	18\$::BRANCH IF YES
066730	021627	000067		CMP	(SP), #67	::CHAR > 7?
066734	003015			BGT	18\$::BRANCH IF YES
066736	042726	000060		BIC	#60, (SP)+	::STRIP-OFF ASCII
066742	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
066746	001403			BEQ	17\$::BRANCH IF YES
066750	006316			ASL	(SP)	::NO, SHIFT PRESENT
066752	006316			ASL	(SP)	::CHAR OVER TO MAKE
066754	006316			ASL	(SP)	::ROOM FOR NEW ONE.
066756	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
066762	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
066766	000667			BR	7\$::GET THE NEXT ONE
066770	104401	001216	18\$:	TYPE	, \$QUES	::TYPE ?<CR><LF>
066774	000720			BR	20\$::SIMULATE CONTROL-U
				.DSABL	LSB	


```

067206 001003          BNE      8$          ;;BR IF NO
067210 104401 067337  TYPE      ,SCNTLU      ;;TYPE A CONTROL 'U'
067214 000726          BR        1$          ;;GO START OVER
067216 122713 000022  8$:      CMPB     #22,(R3)      ;;IS CHARACTER A '^R'?
067222 001011          BNE      3$          ;;BRANCH IF NO
067224 105013          CLRB     (R3)          ;;CLEAR THE CHARACTER
067226 104401 001217  TYPE      ,SCRLF      ;;TYPE A 'CR' & 'LF'
067232 104401 067322  TYPE      ,STTYIN      ;;TYPE THE INPUT STRING
067236 000717          BR        2$          ;;GO PICKUP ANOTHER CHACTER
067240 104401 001216  4$:      TYPE      ,SQUES      ;;TYPE A '?'
067244 000712          BR        1$          ;;CLEAR THE BUFFER AND LOOP
067246 111337 067320  3$:      MOVB     (R3),9$      ;;ECHO THE CHARACTER
067252 104401 067320  TYPE      ,9$
067256 122723 000015  CMPB     #15,(R3)+      ;;CHECK FOR RETURN
067262 001305          BNE      2$          ;;LOOP IF NOT RETURN
067264 105063 177777  CLRB     -1(R3)        ;;CLEAR RETURN (THE 15)
067270 104401 001220  TYPE      ,SLF          ;;TYPE A LINE FEED
067274 005726          TST      (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
067276 012603          MOV      (SP)+,R3      ;;RESTORE R3
067300 011646          MOV      (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
067302 016666 000004 000002  MOV      4(SP),2(SP)      ;; FIRST ASCII CHARACTER ON IT
067310 012766 067322 000004  MOV      #$TTYIN,4(SP)
067316 000002          RTI          ;;RETURN
067320 000          9$:      .BYTE     0          ;;STORAGE FOR ASCII CHAR. TO TYPE
067321 000          .BYTE     0          ;;TERMINATOR
067322          $TTYIN: .BLKB     8          ;;RESERVE 8 BYTES FOR TTY INPUT
067332 136 103 015  $CNTLC: .ASCIZ  /^C/<15><12>      ;;CONTROL 'C'
067337 136 125 015  $CNTLU: .ASCIZ  /^U/<15><12>      ;;CONTROL 'U'
067344 136 107 015  $CNTLG: .ASCIZ  /^G/<15><12>      ;;CONTROL 'G'
067351 015 012 123  $MSWR: .ASCIZ  <15><12>/SWR = /
067362 040 040 116  $MNEW: .ASCIZ  / NEW = /

```

9

.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

```

```

067374 011646          $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
067376 016666 000004 000002  MOV      4(SP),2(SP)      ;;INPUT NUMBER
067404 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
067406 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
067410 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
067412 104412 1$:      RDLIN      ;;READ AN ASCII LINE
067414 012600          MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
067416 005001          CLR      R1          ;;CLEAR DATA WORD
067420 005002          CLR      R2
067422 112046 2$:      MOVB     (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
067424 001412          BEQ      3$          ;;IF ZERO GET OUT
067426 006301          ASL     R1          ;;*2
067430 006102          ROL     R2
067432 006301          ASL     R1          ;;*4
067434 006102          ROL     R2

```

```

067436 006301 ASL R1 ;;*8
067440 006102 ROL R2
067442 042716 177770 BIC #^C7,(SP) ;;STRIP THE ASCII JUNK
067446 062601 ADD (SP)+,R1 ;;ADD IN THIS DIGIT
067450 000764 BR 2$ ;;LOOP
067452 005726 3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
067454 010166 000012 MOV R1,12(SP) ;;SAVE THE RESULT
067460 010237 067474 MOV R2,$HIOCT
067464 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
067466 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
067470 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
067472 000002 RTI ;;RETURN
067474 000000 $HIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE
10 .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.

```

067476 016646 000002 $TRAP: MOV 2(SP),-(SP) ;;ASSUME THE STATUS OF
067502 042716 000020 BIC #20,(SP) ;; THE CALLER--DO NOT ALLOW
067506 012746 067514 MOV #1$,-(SP) ;; T-BIT TRAPS
067512 000002 RTI ;;SET THE NEW STATUS
067514 000046 1$: MOV R0,-(SP) ;;SAVE R0
067516 016600 000002 MOV 2(SP),R0 ;;GET TRAP ADDRESS
067522 005740 TST -(R0) ;;BACKUP BY 2
067524 111000 MOV (R0),R0 ;;GET RIGHT BYTE OF TRAP
067526 006300 ASL R0 ;;POSITION FOR INDEXING
067530 016000 067550 MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
067534 000200 RTS R0 ;;GO TO ROUTINE
  
```

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

```

067536 011646 $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
067540 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
067546 000002 RTI ;;RESTORE THE PSW
  
```

.SBTTL TRAP TABLE

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

	ROUTINE		
067550	067536	\$TRPAD: .WORD \$TRAP2	
067552	065006	\$TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
067554	064604	\$TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
067556	064560	\$TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
067560	064620	\$TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
067562	064334	\$TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
067564	064260	\$TYPBN	TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
067566	066524	\$GTSWR	TRAP+7(104407) GET SOFT-SWR SETTING

TRAP TABLE

067570	066434			\$CKSWR	::CALL=CKSWR	TRAP+10(104410)	TEST FOR CHANGE IN SOFT-SWR
067572	066776			\$RDCHR	::CALL=RDCHR	TRAP+11(104411)	TTY TYPEIN CHARACTER ROUTINE
067574	067066			\$RDLIN	::CALL=RDLIN	TRAP+12(104412)	TTY TYPEIN STRING ROUTINE
067576	067374			\$RDOCT	::CALL=RDOCT	TRAP+13(104413)	READ AN OCTAL NUMBER FROM TTY
067600	064164			\$SAVREG	::CALL=SAVREG	TRAP+14(104414)	SAVE R0-R5 ROUTINE
067602	064222			\$RESREG	::CALL=RESREG	TRAP+15(104415)	RESTORE R0-R5 ROUTINE

11

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

067604	012737	067744	000024	\$PWRDN:	MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST UP
067612	012737	000340	000026		MOV	#340,@#PWRVEC+2	::PRIO:7
067620	010046				MOV	R0,-(SP)	::PUSH R0 ON STACK
067622	010146				MOV	R1,-(SP)	::PUSH R1 ON STACK
067624	010246				MOV	R2,-(SP)	::PUSH R2 ON STACK
067626	010346				MOV	R3,-(SP)	::PUSH R3 ON STACK
067630	010446				MOV	R4,-(SP)	::PUSH R4 ON STACK
067632	010546				MOV	R5,-(SP)	::PUSH R5 ON STACK
067634	017746	111314			MOV	@SWR,-(SP)	::PUSH @SWR ON STACK
067640	010637	067750			MOV	SP,\$SAVR6	::SAVE SP
067644	012737	067656	000024		MOV	#\$PWRUP,@#PWRVEC	::SET UP VECTOR
067652	000000				HALT		
067654	000776				BR	.-2	::HANG UP

:POWER UP ROUTINE

067656	012737	067744	000024	\$PWRUP:	MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST DOWN
067664	013706	067750			MOV	\$SAVR6,SP	::GET SP
067670	005037	067750			CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
067674	005237	067750		1\$:	INC	\$SAVR6	::WAIT FOR THE INC
067700	001375				BNE	1\$::OF WORD
067702	012677	111246			MOV	(SP)+,@SWR	::POP STACK INTO @SWR
067706	012605				MOV	(SP)+,R5	::POP STACK INTO R5
067710	012604				MOV	(SP)+,R4	::POP STACK INTO R4
067712	012603				MOV	(SP)+,R3	::POP STACK INTO R3
067714	012602				MOV	(SP)+,R2	::POP STACK INTO R2
067716	012601				MOV	(SP)+,R1	::POP STACK INTO R1
067720	012600				MOV	(SP)+,R0	::POP STACK INTO R0
067722	012737	067604	000024		MOV	#\$PWRDN,@#PWRVEC	::SET UP THE POWER DOWN VECTOR
067730	012737	000340	000026		MOV	#340,@#PWRVEC+2	::PRIO:7
067736	104401				TYPE		::REPORT THE POWER FAILURE
067740	067752			\$PWRMG:	.WORD	\$POWER	::POWER FAIL MESSAGE POINTER
067742	000002				RTI		
067744	000000			\$ILLUP:	HALT		::THE POWER UP SEQUENCE WAS STARTED
067746	000776				BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
067750	000000			\$SAVR6:	0		::PUT THE SP HERE
067752	015	012	120	\$POWER:	.ASCIZ	<15><12>'POWER'	

12

.SBTTL APT COMMUNICATIONS ROUTINE

067762	112737	000001	070226	\$ATY1:	MOVB	#1,\$FFLG	::TO REPORT FATAL ERROR
067770	112737	000001	070224	\$ATY3:	MOVB	#1,\$MFLG	::TO TYPE A MESSAGE
067776	000403				BR	\$ATYC	
070000	112737	000001	070226	\$ATY4:	MOVB	#1,\$FFLG	::TO ONLY REPORT FATAL ERROR
070006				\$ATYC:			
070006	010046				MOV	R0,-(SP)	::PUSH R0 ON STACK

```

070010 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
070012 105737 070224  TSTB     $MFLG        ;;SHOULD TYPE A MESSAGE?
070016 001450          BEQ      5$           ;;IF NOT: BR
070020 122737 000001 001242  CMPB     #APTENV,$ENV  ;;OPERATING UNDER APT?
070026 001031          BNE     3$           ;;IF NOT: BR
070030 132737 000100 001243  BITB     #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
070036 001425          BEQ      3$           ;;IF NOT: BR
070040 017600 000004          MOV     @4(SP),R0     ;;GET MESSAGE ADDR.
070044 062766 000002 000004  ADD     #2,4(SP)      ;;BUMP RETURN ADDR.
070052 005737 001222          TST     $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
070056 001375          BNE     1$           ;;IF NOT: WAIT
070060 010037 001236          MOV     R0,$MSGAD    ;;PUT ADDR IN MAILBOX
070064 105720          TSTB    (R0)+        ;;FIND END OF MESSAGE
070066 001376          BNE     2$
070070 163700 001236          SUB     $MSGAD,R0    ;;SUB START OF MESSAGE
070074 006200          ASR     R0           ;;GET MESSAGE LNTH IN WORDS
070076 010037 001240          MOV     R0,$MSGGLT  ;;PUT LENGTH IN MAILBOX
070102 012737 000004 001222  MOV     #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
070110 000413          BR      5$
070112 017637 000004 070136 3$:      MOV     @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
070120 062766 000002 000004  ADD     #2,4(SP)      ;;BUMP RETURN ADDRESS
070126 013746 177776          MOV     177776,-(SP) ;;PUSH 177776 ON STACK
070132 004737 065006          JSR     PC,$TYPE     ;;CALL TYPE MACRO
070136 000000          .WORD  0
070140          5$:
070140 105737 070226          10$:   TSTB     $FFLG        ;;SHOULD REPORT FATAL ERROR?
070144 001416          BEQ     12$         ;;IF NOT: BR
070146 005737 001242          TST     $ENV        ;;RUNNING UNDER APT?
070152 001413          BEQ     12$         ;;IF NOT: BR
070154 005737 001222          11$:   TST     $MSGTYPE     ;;FINISHED LAST MESSAGE?
070160 001375          BNE     11$        ;;IF NOT: WAIT
070162 017637 000004 001224          MOV     @4(SP),$FATAL ;;GET ERROR #
070170 062766 000002 000004  ADD     #2,4(SP)      ;;BUMP RETURN ADDR.
070176 005237 001222          INC     $MSGTYPE     ;;TELL APT TO TAKE ERROR
070202 105037 070226          12$:   CLRB     $FFLG        ;;CLEAR FATAL FLAG
070206 105037 070225          CLRB     $LFLG        ;;CLEAR LOG FLAG
070212 105037 070224          CLRB     $MFLG        ;;CLEAR MESSAGE FLAG
070216 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
070220 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
070222 000207          RTS     PC          ;;RETURN
070224 000          $MFLG: .BYTE 0     ;;MESSG. FLAG
070225 000          $LFLG: .BYTE 0     ;;LOG FLAG
070226 000          $FFLG: .BYTE 0     ;;FATAL FLAG
                                .EVEN
                                APTSIZE = 200
                                APTENV  = 001
                                APTPOOL = 100
                                APTCSUP = 040
000200
000001
000100
000040

```

```

1
2
3 070230      200      106      101  SCTMSG: .ASCII <CRLF>@FAILED TO RECOVER THE BAD SECTOR FILE (DEC 144)@
4 070310      200      117      116  .ASCIZ <CRLF>@ON THIS DRIVE@
5 070327      075      000      EQUALS: .ASCIZ @=@
6 070331      101      114      114  ALL: .ASCIZ @ALL@<CRLF>
7 070336      040      077      040  QUES: .ASCIZ @ ? @
8 070342      054      040      000  COMMA: .ASCIZ @, @
9 070345      200      124      117  MSHELP: .ASCII <CRLF>@TO ENSURE THAT NO BAD HEADERS ARE LEFT ON THE DISK@
10 070433      200      120      101  .ASCII <CRLF>@PACK, THIS PROGRAM SHOULD BE HALTED BY TYPING A (^C)@
11 070521      200      103      117  .ASCII <CRLF>@CONTROL C. AS A RESULT, THE PROGRAM WILL BE HALTED@
12 070607      200      127      110  .ASCII <CRLF>@WHEN THE DRIVE UNDER TEST HAS COMPLETED TESTING.@<CRLF>
13 070671      200      124      131  .ASCIZ <CRLF>@TYPE HELP TEXT (Y/N) ? @
14 070722
15 070722      200      103      110  UBUSQST: .ASCIZ <CRLF>@CHANGE ADDRESSES (Y/N) ? @
16 070755      200      125      123  CNSL00: .ASCIZ <CRLF>@USE SAME DEVICES (Y/N) ? @
17 071010      200      102      125  CNSL01: .ASCIZ <CRLF>@BUS ADDRESS @
18 071026      040      114      111  CNSL02: .ASCIZ @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
19 071070      126      105      103  CNSL03: .ASCIZ @VECTOR ADDRESS @
20 071110      040      114      111  CNSL04: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
21 071144      102      122      040  CNSL05: .ASCIZ @BR LEVEL @
22 071156      040      114      111  CNSL06: .ASCIZ @ LIMITS - LO= 0, HI= 7@<CRLF><LF>
23 071207      200
24 071210      200      124      131  CNSL07: .ASCII <CRLF>
25 071275      200      101      116  .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
26 071352      200
27 071353      040      077      111  .ASCII <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
28 071374      200      104      122  CNSL08: .ASCII <CRLF>
29 071410      104      122      111  CNSL09: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
30 071416      040      111      123  MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
31 071436      040      116      117  MSGDRV: .ASCIZ /DRIVE/
32 071453      040      116      117  LODEV: .ASCIZ / IS LOAD DEVICE/
33 071472      040      117      106  NOTPRS: .ASCIZ / NOT PRESENT/
34 071504      040      117      116  NOTAVL: .ASCIZ / NOT AVAILABLE/
OFFLIN: .ASCIZ / OFF LINE/
ONLINE: .ASCIZ / ON LINE /
35
36
37
.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
```

Line	Code	Address	Instruction	Function
58	071520	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	071522	132000	.WORD ATA!OPI!IVC!IAE	:SEEK
60	071524	130000	.WORD ATA!OPI!IVC	:RECALIBRATE
61	071526	020000	.WORD OPI	:DRIVE CLEAR
62	071530	030000	.WORD OPI!IVC	:RELEASE
63	071532	130000	.WORD OPI!ATA!IVC	:OFFSET
64	071534	130000	.WORD OPI!ATA!IVC	:RETURN TO CENTERLINE
65	071536	020000	.WORD OPI	:READ IN PRESET
66	071540	020000	.WORD OPI	:PACK ACKNOWLEDGE
67	071542	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	071544	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	071546	132000	.WORD ATA!OPI!IVC!IAE	:SEARCH
70	071550	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	071552	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	071554	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	071556	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	071560	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	071562	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	071564	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	071566	073300	.WORD WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	071570	073300	.WORD WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	071572	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	071574	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	071576	037200	.WORD OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	071600	037000	.WORD OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	071602	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	071604	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	071606	033300	.WORD OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	071610	033300	.WORD OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	071612	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	071614	130001	.WORD OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)
89				

1
2
3 071616 001
4 071617 002
5 071620 004
6 071621 010
7 071622 020
8 071623 040
9 071624 100
10 071625 200
11

.SBTTL ATTENTION (ATA) TABLE

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

		.SBTTL DATA PATTERN TABLE	
1			
2			
3	071626	RGDTPT:	
4	071626	MIXED:	
5	071626 000000	.WORD	0.
6	071630 000001	.WORD	1.
7	071632 000003	.WORD	3.
8	071634 000007	.WORD	7.
9	071636 000017	.WORD	15.
10	071640 000037	.WORD	31.
11	071642 000077	.WORD	63.
12	071644 000177	.WORD	127.
13	071646 000377	.WORD	255.
14	071650 000777	.WORD	511.
15	071652 001777	.WORD	1023.
16	071654 003777	.WORD	2047.
17	071656 007777	.WORD	4095.
18	071660 017777	.WORD	8191.
19	071662 037777	.WORD	16383.
20	071664 077777	.WORD	32767.
21	071666 177777	ONES:	.WORD 65535.
22	071670 177777	.WORD	65535.
23	071672 077777	.WORD	32767.
24	071674 037777	.WORD	16383.
25	071676 017777	.WORD	8191.
26	071700 007777	.WORD	4095.
27	071702 003777	.WORD	2047.
28	071704 001777	.WORD	1023.
29	071706 000777	.WORD	511.
30	071710 000377	.WORD	255.
31	071712 000177	.WORD	127.
32	071714 000077	.WORD	63.
33	071716 000037	.WORD	31.
34	071720 000017	.WORD	15.
35	071722 000007	.WORD	7.
36	071724 000003	.WORD	3.
37	071726 000001	.WORD	1.
38	071730 000000	ZEROS:	.WORD 0.
39	071732 000000	.WORD	0.
40	071734 000001	.WORD	1.
41	071736 000002	.WORD	2.
42	071740 000004	.WORD	4.
43	071742 000010	.WORD	8.
44	071744 000020	.WORD	16.
45	071746 000040	.WORD	32.
46	071750 000100	.WORD	64.
47	071752 000200	.WORD	128.
48	071754 000400	.WORD	256.
49	071756 001000	.WORD	512.
50	071760 002000	.WORD	1024.
51	071762 004000	.WORD	2048.
52	071764 010000	.WORD	4096.
53	071766 020000	.WORD	8192.
54	071770 040000	.WORD	16384.
55	071772 100000	.WORD	32768.
56	071774 100000	.WORD	32768.
57	071776 040000	.WORD	16384.

Line	Code	Value	Label	Count
58	072000	020000	.WORD	8192.
59	072002	010000	.WORD	4096.
60	072004	004000	.WORD	2048.
61	072006	002000	.WORD	1024.
62	072010	001000	.WORD	512.
63	072012	000400	.WORD	256.
64	072014	000200	.WORD	128.
65	072016	000100	.WORD	64.
66	072020	000040	.WORD	32.
67	072022	000020	.WORD	16.
68	072024	000010	.WORD	8.
69	072026	000004	.WORD	4.
70	072030	000002	.WORD	2.
71	072032	000001	.WORD	1.
72	072034	000000	.WORD	0.
73	072036	177777	.WORD	65535.
74	072040	177776	.WORD	65534.
75	072042	177774	.WORD	65532.
76	072044	177770	.WORD	65528.
77	072046	177760	.WORD	65520.
78	072050	177740	.WORD	65504.
79	072052	177700	.WORD	65472.
80	072054	177600	.WORD	65408.
81	072056	177400	.WORD	65280.
82	072060	177000	.WORD	65024.
83	072062	176000	.WORD	64512.
84	072064	174000	.WORD	63488.
85	072066	170000	.WORD	61440.
86	072070	160000	.WORD	57344.
87	072072	140000	.WORD	49152.
88	072074	100000	.WORD	32768.
89	072076	000000	.WORD	0.
90	072100	000000	.WORD	0.
91	072102	100000	.WORD	32768.
92	072104	140000	.WORD	49152.
93	072106	160000	.WORD	57344.
94	072110	170000	.WORD	61440.
95	072112	174000	.WORD	63488.
96	072114	176000	.WORD	64512.
97	072116	177000	.WORD	65024.
98	072120	177400	.WORD	65280.
99	072122	177600	.WORD	65408.
100	072124	177700	.WORD	65472.
101	072126	177740	.WORD	65504.
102	072130	177760	.WORD	65520.
103	072132	177770	.WORD	65528.
104	072134	177774	.WORD	65532.
105	072136	177776	.WORD	65534.
106	072140	177777	.WORD	65535.
107	072142	125252	.WORD	43690.
108	072144	152525	.WORD	43690./2
109	072146	125252	.WORD	43690.
110	072150	177777	.WORD	65535.
111	072152	177776	.WORD	65534.
112	072154	177775	.WORD	65533.
113	072156	177773	.WORD	65531.
114	072160	177767	.WORD	65527.

EARLY:

115	072162	177757	.WORD	65519.
116	072164	177737	.WORD	65503.
117	072166	177677	.WORD	65471.
118	072170	177577	.WORD	65407.
119	072172	177377	.WORD	65279.
120	072174	176777	.WORD	65023.
121	072176	175777	.WORD	64511.
122	072200	173777	.WORD	63487.
123	072202	167777	.WORD	61439.
124	072204	157777	.WORD	57343.
125	072206	137777	.WORD	49151.
126	072210	077777	.WORD	32767.
127	072212	077777	.WORD	32767.
128	072214	137777	.WORD	49151.
129	072216	157777	.WORD	57343.
130	072220	167777	.WORD	61439.
131	072222	173777	.WORD	63487.
132	072224	175777	.WORD	64511.
133	072226	176777	.WORD	65023.
134	072230	177377	.WORD	65279.
135	072232	177577	.WORD	65407.
136	072234	177677	.WORD	65471.
137	072236	177737	.WORD	65503.
138	072240	177757	.WORD	65519.
139	072242	177767	.WORD	65527.
140	072244	177773	.WORD	65531.
141	072246	177775	.WORD	65533.
142	072250	177776	.WORD	65534.
143	072252	177777	.WORD	65535.
144	072254			
145				

ENRGDT :

					.SBTTL	ERROR MESSAGE TABLE
1						
2						
3	072254	076650	000000		EMT1:	.WORD EMS1,0
4	072260	076717	076734	000000	EMT2:	.WORD EMS2,EMS3,0
5	072266	076717	076777	000000	EMT3:	.WORD EMS2,EMS4,0
6	072274	077042	077072	000000	EMT4:	.WORD EMS5,EMS6,0
7	072302	077042	077204	000000	EMT5:	.WORD EMS5,EMS10,0
8	072310	103644	101063	000000	EMT6:	.WORD EMS167,EMS64,0
9	072316	101631	103671	000000	EMT7:	.WORD EMS110,EMS170,0
10	072324	077137	000000		EMT10:	.WORD EMS7,0
11	072330	077204	000000		EMT11:	.WORD EMS10,0
12	072334	077246	077257	000000	EMT12:	.WORD EMS11,EMS12,0
13	072342	077320	077331	077342	EMT13:	.WORD EMS13,EMS14,EMS15,EMS16,0
14	072354	077414	101063	000000	EMT14:	.WORD EMS17,EMS64,0
15	072362	077246	077477	000000	EMT15:	.WORD EMS11,EMS21,0
16	072370	077246	077522	077651	EMT16:	.WORD EMS11,EMS22,EMS27,0
17	072400	077246	077536	077662	EMT17:	.WORD EMS11,EMS23,EMS30,0
18	072410	077246	077564	077662	EMT20:	.WORD EMS11,EMS24,EMS30,0
19	072420	077246	077613	077651	EMT21:	.WORD EMS11,EMS25,EMS27,0
20	072430	077246	077630	077651	EMT22:	.WORD EMS11,EMS26,EMS27,0
21	072440	077246	077672	077662	EMT23:	.WORD EMS11,EMS31,EMS30,0
22	072450	077246	077721	077662	EMT24:	.WORD EMS11,EMS32,EMS30,0
23	072460	077246	077750	077662	EMT25:	.WORD EMS11,EMS33,EMS30,0
24	072470	077246	077776	077662	EMT26:	.WORD EMS11,EMS34,EMS30,0
25	072500	077246	100047	077662	EMT27:	.WORD EMS11,EMS35,EMS30,0
26	072510	077246	100076	077662	EMT30:	.WORD EMS11,EMS36,EMS30,0
27	072520	077246	100125	077662	EMT31:	.WORD EMS11,EMS37,EMS30,0
28	072530	077246	100153	077662	EMT32:	.WORD EMS11,EMS40,EMS30,0
29	072540	077246	100202	077662	EMT33:	.WORD EMS11,EMS41,EMS30,0
30	072550	077246	100230	077662	EMT34:	.WORD EMS11,EMS42,EMS30,0
31	072560	077246	100257	077662	EMT35:	.WORD EMS11,EMS43,EMS30,0
32	072570	077246	100306	077662	EMT36:	.WORD EMS11,EMS44,EMS30,0
33	072600	077246	100361	077662	EMT37:	.WORD EMS11,EMS45,EMS30,0
34	072610	101147	077454	000000	EMT40:	.WORD EMS66,EMS20,0
35	072616	101335	103046	101343	EMT41:	.WORD EMS75,EMS141,EMS76,0
36	072626	103137	103147	101271	EMT42:	.WORD EMS144,EMS145,EMS72,EMS76,0
37	072640	100453	100567	101343	EMT43:	.WORD EMS47,EMS53,EMS76,0
38	072650	101374	100567	101343	EMT44:	.WORD EMS77,EMS53,EMS76,0
39	072660	101422	100567	101343	EMT45:	.WORD EMS100,EMS53,EMS76,0
40	072670	101450	100567	101343	EMT46:	.WORD EMS101,EMS53,EMS76,0
41	072700	101101	101063	000000	EMT47:	.WORD EMS65,EMS64,0
42	072706	077320	077342	101035	EMT50:	.WORD EMS13,EMS15,EMS63,0
43	072716	077246	100424	077662	EMT51:	.WORD EMS11,EMS46,EMS30,EMS67,0
44	072730	100453	100567	101217	EMT52:	.WORD EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
45	072746	100453	100567	101217	EMT53:	.WORD EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
46	072764	100502	100567	101217	EMT54:	.WORD EMS50,EMS53,EMS67,0
47	072774	103074	103111	100567	EMT55:	.WORD EMS142,EMS143,EMS53,EMS67,0
48	073006	100531	101271	101217	EMT56:	.WORD EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
49	073024	103644	101301	101217	EMT57:	.WORD EMS167,EMS73,EMS67,0
50	073034	100652	101217	102031	EMT60:	.WORD EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
51	073052	101256	100652	101217	EMT61:	.WORD EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
52	073072	103025	101217	102031	EMT62:	.WORD EMS140,EMS67,EMS115,EMS47,EMS70,0
53	073106	000000			EMT63:	.WORD
54	073110	103232	103274	101244	EMT64:	.WORD EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
55	073130	100557	104271	104452	EMT65:	.WORD EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
56	073150	103603	101525	101271	EMT66:	.WORD EMS165,EMS103,EMS72,EMS124,0
57	073162	103603	101525	101271	EMT67:	.WORD EMS165,EMS103,EMS72,EMS171,0

58	073174	100424	077454	103476	EMT70: .WORD	EMS46,EMS20,EMS163,0
59	073204	101256	101450	103476	EMT71: .WORD	EMS71,EMS101,EMS163,0
60	073214	100453	101271	103476	EMT72: .WORD	EMS47,EMS72,EMS163,EMS115,EMS140,EMS141,0
61	073232	100453	101271	103476	EMT73: .WORD	EMS47,EMS72,EMS163,EMS115,EMS141,EMS72,0
62	073250	100652	100567	103476	EMT74: .WORD	EMS56,EMS53,EMS163,0
63	073260	101256	100652	103476	EMT75: .WORD	EMS71,EMS56,EMS163,EMS115,EMS150,EMS152,EMS72,0
64	073300	100502	100567	103476	EMT76: .WORD	EMS50,EMS53,EMS163,0
65	073310	103074	103111	100567	EMT77: .WORD	EMS142,EMS143,EMS53,EMS163,0
66	073322	101335	103046	103476	EMT100: .WORD	EMS75,EMS141,EMS163,EMS115,EMS47,EMS70,0
67	073340	101335	103232	103476	EMT101: .WORD	EMS75,EMS150,EMS163,EMS115,EMS56,EMS73,0
68	073356	103644	101301	103476	EMT102: .WORD	EMS167,EMS73,EMS163,0
69	073366	103215	103250	101316	EMT103: .WORD	EMS147,EMS151,EMS74,EMS163,0
70	073400	100531	101316	103476	EMT104: .WORD	EMS51,EMS74,EMS163,EMS115,EMS50,EMS70,0
71	073416	103603	101422	103476	EMT105: .WORD	EMS165,EMS100,EMS163,0
72	073426	103603	101374	103476	EMT106: .WORD	EMS165,EMS77,EMS163,0
73	073436	103137	103147	100567	EMT107: .WORD	EMS144,EMS145,EMS53,EMS163,EMS115,EMS143,EMS70,0
74	073456	101631	101671	102034	EMT110: .WORD	EMS110,EMS112,EMS116,EMS111,0
75	073470	101755	076777	000000	EMT111: .WORD	EMS113,EMS4,0
76	073476	101755	076734	000000	EMT112: .WORD	EMS113,EMS3,0
77	073504	101631	102031	102034	EMT113: .WORD	EMS110,EMS115,EMS116,EMS117,EMS114,0
78	073520	101755	102106	000000	EMT114: .WORD	EMS113,EMS120,0
79	073526	102127	102567	102613	EMT115: .WORD	EMS121,EMS132,EMS133,0
80	073536	102172	102567	102613	EMT116: .WORD	EMS122,EMS132,EMS133,0
81	073546	102227	102567	102613	EMT117: .WORD	EMS123,EMS132,EMS133,0
82	073556	102272	102567	102613	EMT120: .WORD	EMS124,EMS132,EMS133,0
83	073566	102324	102567	102613	EMT121: .WORD	EMS125,EMS132,EMS133,0
84	073576	102367	102567	102613	EMT122: .WORD	EMS126,EMS132,EMS133,0
85	073606	102770	102567	102613	EMT123: .WORD	EMS137,EMS132,EMS133,0
86	073616	102471	102567	102613	EMT124: .WORD	EMS130,EMS132,EMS133,0
87	073626	102527	102567	102613	EMT125: .WORD	EMS131,EMS132,EMS133,0
88	073636	102127	102567	102636	EMT126: .WORD	EMS121,EMS132,EMS134,EMS123,0
89	073650	102172	102567	102636	EMT127: .WORD	EMS122,EMS132,EMS134,EMS123,0
90	073662	102227	102567	102636	EMT130: .WORD	EMS123,EMS132,EMS134,EMS123,0
91	073674	102272	102567	102636	EMT131: .WORD	EMS124,EMS132,EMS134,EMS123,0
92	073706	102324	102567	102636	EMT132: .WORD	EMS125,EMS132,EMS134,EMS123,0
93	073720	102367	102567	102636	EMT133: .WORD	EMS126,EMS132,EMS134,EMS123,0
94	073732	102770	102567	102636	EMT134: .WORD	EMS137,EMS132,EMS134,EMS123,0
95	073744	102471	102567	102636	EMT135: .WORD	EMS130,EMS132,EMS134,EMS123,0
96	073756	102527	102567	102636	EMT136: .WORD	EMS131,EMS132,EMS134,EMS123,0
97	073770	102127	102567	102700	EMT137: .WORD	EMS121,EMS132,EMS135,0
98	074000	102227	102567	102700	EMT140: .WORD	EMS123,EMS132,EMS135,0
99	074010	102127	102567	102743	EMT141: .WORD	EMS121,EMS132,EMS136,0
100	074020	102770	102567	102743	EMT142: .WORD	EMS137,EMS132,EMS136,0
101	074030	102272	102567	102743	EMT143: .WORD	EMS124,EMS132,EMS136,0
102	074040	102324	102567	102743	EMT144: .WORD	EMS125,EMS132,EMS136,0
103	074050	102367	102567	102743	EMT145: .WORD	EMS126,EMS132,EMS136,0
104	074060	102527	102567	102743	EMT146: .WORD	EMS131,EMS132,EMS136,0
105	074070	103720	102567	102743	EMT147: .WORD	EMS171,EMS132,EMS136,0
106	074100	102471	102567	102743	EMT150: .WORD	EMS130,EMS132,EMS136,0
107	074110	103025	102031	103046	EMT151: .WORD	EMS140,EMS115,EMS141,EMS70,0
108	074122	103074	102031	103111	EMT152: .WORD	EMS142,EMS115,EMS143,EMS72,0
109	074134	103137	103147	103176	EMT153: .WORD	EMS144,EMS145,EMS146,EMS115,EMS143,EMS72,0
110	074152	103137	103147	077454	EMT154: .WORD	EMS144,EMS145,EMS20,EMS115,EMS143,EMS70,0
111	074170	103232	103274	103342	EMT155: .WORD	EMS150,EMS152,EMS154,EMS153,0
112	074202	103215	103250	103342	EMT156: .WORD	EMS147,EMS151,EMS154,EMS155,0
113	074214	103215	103250	103376	EMT157: .WORD	EMS147,EMS151,EMS156,EMS157,0
114	074226	103446	103376	103431	EMT160: .WORD	EMS161,EMS156,EMS160,0

115	074236	077613	077651	103376	EMT161:	.WORD	EMS25,EMS27,EMS156,EMS160,0
116	074250	077630	077651	103376	EMT162:	.WORD	EMS26,EMS27,EMS156,EMS160,0
117	074262	100453	103476	103025	EMT163:	.WORD	EMS47,EMS163,EMS140,0
118	074272	100453	077454	000000	EMT164:	.WORD	EMS47,EMS20,0
119	074300	100652	077454	000000	EMT165:	.WORD	EMS56,EMS20,0
120	074306	100424	077454	000000	EMT166:	.WORD	EMS46,EMS20,0
121	074314	104772	077454	000000	EMT167:	.WORD	EMS224,EMS20,0
122	074322	103644	103342	103617	EMT170:	.WORD	EMS167,EMS154,EMS166,0
123	074332	103644	103342	103414	EMT171:	.WORD	EMS167,EMS154,EMS157,0
124	074342	103644	103342	103356	EMT172:	.WORD	EMS167,EMS154,EMS155,0
125	074352	103720	102567	102613	EMT173:	.WORD	EMS171,EMS132,EMS133,0
126	074362	103720	102567	102636	EMT174:	.WORD	EMS171,EMS132,EMS134,EMS123,0
127	074374	101755	104073	000000	EMT175:	.WORD	EMS113,EMS177,0
128	074402	104115	104132	000000	EMT176:	.WORD	EMS200,EMS201,0
129	074410	104230	103046	077662	EMT177:	.WORD	EMS203,EMS141,EMS30,EMS202,0
130	074422	104230	100531	077662	EMT200:	.WORD	EMS203,EMS51,EMS30,EMS202,0
131	074434	104230	104243	077662	EMT201:	.WORD	EMS203,EMS204,EMS30,EMS202,0
132	074446	104230	100502	077662	EMT202:	.WORD	EMS203,EMS50,EMS30,EMS202,0
133	074460	104230	103111	077662	EMT203:	.WORD	EMS203,EMS143,EMS30,EMS202,0
134	074472	103232	101316	104200	EMT204:	.WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
135	074510	100502	101525	101271	EMT205:	.WORD	EMS50,EMS103,EMS72,0
136	074520	101534	101301	104200	EMT206:	.WORD	EMS104,EMS73,EMS202,0
137	074530	101335	103046	102031	EMT207:	.WORD	EMS75,EMS141,EMS115,EMS140,0
138	074542	101335	103232	000000	EMT210:	.WORD	EMS75,EMS150,0
139	074550	100531	101271	102031	EMT211:	.WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
140	074564	103074	100567	102031	EMT212:	.WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
141	074600	100502	101525	100567	EMT213:	.WORD	EMS50,EMS103,EMS53,0
142	074610	100557	104271	103617	EMT214:	.WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
143	074630	100557	102064	100652	EMT215:	.WORD	EMS52,EMS117,EMS56,EMS163,0
144	074642	100652	077662	101063	EMT216:	.WORD	EMS56,EMS30,EMS64,0
145	074652	100424	077662	101063	EMT217:	.WORD	EMS46,EMS30,EMS64,0
146	074662	077672	077662	101063	EMT220:	.WORD	EMS31,EMS30,EMS64,0
147	074672	100453	077662	101063	EMT221:	.WORD	EMS47,EMS30,EMS64,0
148	074702	100557	102064	101374	EMT222:	.WORD	EMS52,EMS117,EMS77,0
149	074712	100557	102064	101035	EMT223:	.WORD	EMS52,EMS117,EMS63,0
150	074722	000000			EMT224:	.WORD	
151	074724	000000			EMT225:	.WORD	
152	074726	000000			EMT226:	.WORD	
153	074730	000000			EMT227:	.WORD	
154	074732	000000			EMT230:	.WORD	
155	074734	000000			EMT231:	.WORD	
156	074736	000000			EMT232:	.WORD	
157	074740	000000			EMT233:	.WORD	
158	074742	000000			EMT234:	.WORD	
159	074744	000000			EMT235:	.WORD	
160	074746	000000			EMT236:	.WORD	
161	074750	000000			EMT237:	.WORD	
162	074752	000000			EMT240:	.WORD	
163	074754	000000			EMT241:	.WORD	
164	074756	000000			EMT242:	.WORD	
165	074760	000000			EMT243:	.WORD	
166	074762	000000			EMT244:	.WORD	
167	074764	000000			EMT245:	.WORD	
168	074766	103644	102567	104330	EMT246:	.WORD	EMS167,EMS132,EMS207,0
169	074776	103644	102567	104355	EMT247:	.WORD	EMS167,EMS132,EMS210,EMS125,0
170	075010	103644	104366	104355	EMT250:	.WORD	EMS167,EMS211,EMS210,EMS207,EMS206,0
171	075024	104404	104427	000000	EMT251:	.WORD	EMS212,EMS213,0

172	075032	103603	104404	000000	EMT252:	.WORD	EMS165,EMS212,0
173	075040	103215	103250	103376	EMT253:	.WORD	EMS147,EMS151,EMS156,EMS210,EMS26,EMS27,0
174	075056	104230	101450	077662	EMT254:	.WORD	EMS203,EMS101,EMS30,0
175	075066	104230	103644	077662	EMT255:	.WORD	EMS203,EMS167,EMS30,0
176	075076	104230	101422	077662	EMT256:	.WORD	EMS203,EMS100,EMS30,0
177	075106	077246	100424	077662	EMT257:	.WORD	EMS11,EMS46,EMS30,EMS102,0
178	075120	100557	102064	100652	EMT260:	.WORD	EMS52,EMS117,EMS56,EMS102,0
179	075132	100557	104271	104574	EMT261:	.WORD	EMS52,EMS205,EMS220,EMS206,EMS115,EMS51,EMS72,0
180	075152	101534	101301	104200	EMT262:	.WORD	EMS104,EMS73,EMS202,0
181	075162	100502	100567	101476	EMT263:	.WORD	EMS50,EMS53,EMS102,0
182	075172	100652	100567	101476	EMT264:	.WORD	EMS56,EMS53,EMS102,EMS115,EMS150,EMS152,EMS70,0
183	075212	101256	100652	101476	EMT265:	.WORD	EMS71,EMS56,EMS102,EMS115,EMS150,EMS152,EMS72,0
184	075232	103074	103111	100567	EMT266:	.WORD	EMS142,EMS143,EMS53,EMS102,0
185	075244	100502	103176	102031	EMT267:	.WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,0
186	075262	100453	100567	101476	EMT270:	.WORD	EMS47,EMS53,EMS102,EMS115,EMS140,0
187	075276	100453	100567	101476	EMT271:	.WORD	EMS47,EMS53,EMS102,EMS115,EMS141,EMS72,0
188	075314	101335	103046	101476	EMT272:	.WORD	EMS75,EMS141,EMS102,EMS115,EMS47,EMS73,0
189	075332	100531	101316	101476	EMT273:	.WORD	EMS51,EMS74,EMS102,EMS115,EMS50,EMS70,0
190	075350	101035	100567	100727	EMT274:	.WORD	EMS63,EMS53,EMS57,EMS115,EMS41,EMS146,0
191	075366	102034	100567	100202	EMT275:	.WORD	EMS116,EMS53,EMS41,EMS57,0
192	075400	100453	100567	100727	EMT276:	.WORD	EMS47,EMS53,EMS57,EMS115,EMS140,0
193	075414	100453	100567	100727	EMT277:	.WORD	EMS47,EMS53,EMS57,EMS115,EMS141,EMS72,0
194	075432	100652	100567	100727	EMT300:	.WORD	EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS70,0
195	075452	101256	100652	100567	EMT301:	.WORD	EMS71,EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS72,0
196	075474	103603	101422	101525	EMT302:	.WORD	EMS165,EMS100,EMS103,EMS57,0
197	075506	103603	101450	101525	EMT303:	.WORD	EMS165,EMS101,EMS103,EMS57,0
198	075520	103603	101374	101525	EMT304:	.WORD	EMS165,EMS77,EMS103,EMS57,0
199	075532	100424	077662	101063	EMT305:	.WORD	EMS46,EMS30,EMS64,EMS57,0
200	075544	100502	100567	100727	EMT306:	.WORD	EMS50,EMS53,EMS57,0
201	075554	100502	103176	102031	EMT307:	.WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,EMS57,0
202	075574	103074	103111	100567	EMT310:	.WORD	EMS142,EMS143,EMS53,EMS57,0
203	075606	101534	101525	100567	EMT311:	.WORD	EMS104,EMS103,EMS53,EMS57,0
204	075620	101563	101525	100567	EMT312:	.WORD	EMS105,EMS103,EMS53,EMS57,0
205	075632	103137	103147	101525	EMT313:	.WORD	EMS144,EMS145,EMS103,EMS57,EMS115,EMS143,EMS70,0
206	075652	100230	101525	100567	EMT314:	.WORD	EMS42,EMS103,EMS53,EMS57,0
207	075664	077672	101525	100567	EMT315:	.WORD	EMS31,EMS103,EMS53,EMS57,0
208	075676	101256	077672	101525	EMT316:	.WORD	EMS71,EMS31,EMS103,EMS57,0
209	075710	100257	101525	100727	EMT317:	.WORD	EMS43,EMS103,EMS57,0
210	075720	100361	101525	100727	EMT320:	.WORD	EMS45,EMS103,EMS57,0
211	075730	100306	101525	100727	EMT321:	.WORD	EMS44,EMS103,EMS57,0
212	075740	101612	077454	000000	EMT322:	.WORD	EMS106,EMS20,0
213	075746	100076	101525	100727	EMT323:	.WORD	EMS36,EMS103,EMS57,0
214	075756	103773	100076	101525	EMT324:	.WORD	EMS173,EMS36,EMS103,EMS57,0
215	075770	103753	100076	101525	EMT325:	.WORD	EMS172,EMS36,EMS103,EMS57,0
216	076002	077320	104010	077342	EMT326:	.WORD	EMS13,EMS174,EMS15,EMS35,EMS53,EMS175,0
217	076020	103215	103250	101316	EMT327:	.WORD	EMS147,EMS151,EMS74,EMS175,0
218	076032	101147	100567	104016	EMT330:	.WORD	EMS66,EMS53,EMS175,0
219	076042	077750	101525	100567	EMT331:	.WORD	EMS33,EMS103,EMS53,EMS175,0
220	076054	100153	101525	100567	EMT332:	.WORD	EMS40,EMS103,EMS53,EMS57,0
221	076066	100531	101316	100727	EMT333:	.WORD	EMS51,EMS74,EMS57,EMS115,EMS50,EMS70,0
222	076104	101335	103046	100727	EMT334:	.WORD	EMS75,EMS141,EMS57,EMS115,EMS47,EMS73,0
223	076122	101335	103232	103274	EMT335:	.WORD	EMS75,EMS150,EMS152,EMS57,EMS115,EMS56,EMS73,0
224	076142	100755	100770	101017	EMT336:	.WORD	EMS60,EMS61,EMS62,0
225	076152	102034	102064	077776	EMT337:	.WORD	EMS116,EMS117,EMS34,0
226	076162	077776	100567	100601	EMT340:	.WORD	EMS34,EMS53,EMS54,EMS111,0
227	076174	100623	077776	000000	EMT341:	.WORD	EMS55,EMS34,0
228	076202	100557	102064	100652	EMT342:	.WORD	EMS52,EMS117,EMS56,EMS57,0

229	076214	100557	102064	100361	EMT343: .WORD	EMS52,EMS117,EMS45,EMS57,0
230	076226	100557	102064	100306	EMT344: .WORD	EMS52,EMS117,EMS44,EMS57,0
231	076240	100557	102064	104614	EMT345: .WORD	EMS52,EMS117,EMS221,0
232	076250	101612	077454	102031	EMT346: .WORD	EMS106,EMS20,EMS115,EMS223,EMS72,0
233	076264	100557	104271	104664	EMT347: .WORD	EMS52,EMS205,EMS222,EMS206,0
234	076276	101335	103232	101476	EMT350: .WORD	EMS75,EMS150,EMS102,EMS115,EMS56,EMS73,0
235	076314	103644	101301	101476	EMT351: .WORD	EMS167,EMS73,EMS102,0
236	076324	104470	000000		EMT352: .WORD	EMS215,0
237	076330	104541	104230	104511	EMT353: .WORD	EMS217,EMS203,EMS216,0
238	076340	104614	077454	000000	EMT354: .WORD	EMS221,EMS20,0
239						

1	076346	105044	105650	105725	EHT1: .WORD	EH1,STSH1,STSH2,STSH4,0
2	076360	105650	105725	106052	EHT2: .WORD	STSH1,STSH2,STSH4,0
3						
4	076370	105063	000000		EHT110: .WORD	EH110,0
5	076374	105072	000000		EHT111: .WORD	EH111,0
6						
7	076400	105111	000000		EHT114: .WORD	EH114,0
8	076404	105140	105650	105725	EHT223: .WORD	EH223,STSH1,STSH2,STSH4,0
9	076416	105166	105650	105725	EHT256: .WORD	EH256,STSH1,STSH2,STSH4,0
10						
11	076430	105242	105650	105725	EHT336: .WORD	EH336,STSH1,STSH2,STSH4,0
12	076442	105301	105650	105725	EHT337: .WORD	EH337,STSH1,STSH2,STSH4,0
13	076454	105436	105650	105725	EHT344: .WORD	EH344,STSH1,STSH2,STSH4,0
14						
15	076466	105574	000000		EHT353: .WORD	EH353,0
16						

1	076472	106110	106204	106222	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
2	076502	106204	106222	106254	EDT2:	.WORD	STSD1,STSD2,STSD4
3							
4	076510	106116			EDT110:	.WORD	ED110
5	076512	106122			EDT111:	.WORD	ED111
6							
7	076514	106130			EDT114:	.WORD	ED114
8	076516	106140	106204	106222	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
9							
10	076526	106150	106204	106222	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
11	076536	106162	106204	106222	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
12	076546	106162	106204	106222	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
13							
14	076560	106174			EDT353:	.WORD	ED353
15							

1	076562	106267	106305	106305	EFT1:	.WORD	EF111,STSF,STSF,STSF
2	076572	106305	106305	106305	EFT2:	.WORD	STSF,STSF,STSF
3							
4	076600	106266			EFT110:	.WORD	EF110
5	076602	106267			EFT111:	.WORD	EF111
6							
7	076604	106271			EFT114:	.WORD	EF114
8	076606	106271	106305	106305	EFT223:	.WORD	EF114,STSF,STSF,STSF
9							
10	076616	106274	106305	106305	EFT336:	.WORD	EF336,STSF,STSF,STSF
11	076626	106274	106305	106305	EFT337:	.WORD	EF336,STSF,STSF,STSF
12	076636	106274	106305	106305	EFT344:	.WORD	EF336,STSF,STSF,STSF
13							
14	076646	106271			EFT353:	.WORD	EF114
15							

Line	Code	Offset	Address	Label	Text
1				.SBTTL	ERROR MESSAGE STRINGS
2					
3	076650	127	122	117 EMS1:	.ASCIZ @WRONG UNIT SELECTED (RMCS2, BITS 0-2) a
4	076717	104	105	126 EMS2:	.ASCIZ @DEVICE WENT a
5	076734	125	116	101 EMS3:	.ASCIZ @UNAVAILABLE 'DVA' (RMCS1, BIT 11) a
6	076777	116	117	116 EMS4:	.ASCIZ @NONEXISTENT 'NED' (RMCS2, BIT 12) a
7	077042	103	117	115 EMS5:	.ASCIZ @COMMAND NOT COMPLETED, a
8	077072	103	117	116 EMS6:	.ASCIZ @CONTROLLER NOT READY (RMCS1, BIT 7) a
9	077137	104	122	111 EMS7:	.ASCIZ @DRIVE NOT READY 'DRY' (RMDS, BIT 7) a
10	077204	107	117	040 EMS10:	.ASCIZ @GO NOT RESET 'GO' (RMCS1, BIT 0) a
11	077246	111	116	126 EMS11:	.ASCIZ @INVALID a
12	077257	106	125	116 EMS12:	.ASCIZ @FUNCTION CODE (RMCS1, BITS 1-5) a
13	077320	115	101	123 EMS13:	.ASCIZ @MASSBUS a
14	077331	103	117	116 EMS14:	.ASCIZ @CONTROL a
15	077342	102	125	123 EMS15:	.ASCIZ @BUS PARITY ERROR a
16	077364	042	115	103 EMS16:	.ASCIZ @'MCPE' (RMCS1, BIT 13) a
17	077414	124	122	101 EMS17:	.ASCIZ @TRANSFER ERROR (RMCS1, BIT 14) a
18	077454	123	110	117 EMS20:	.ASCIZ @SHOULD NOT BE SET a
19	077477	127	117	122 EMS21:	.ASCIZ @WORD COUNT (RMWC) a
20	077522	102	125	123 EMS22:	.ASCIZ @BUS (RMBA) a
21	077536	042	114	102 EMS23:	.ASCIZ @'LBT' (RMDS, BIT 10) a
22	077564	042	101	117 EMS24:	.ASCIZ @'AOE' (RMER1, BIT 09) a
23	077613	104	111	123 EMS25:	.ASCIZ @DISK (RMDA) a
24	077630	103	131	114 EMS26:	.ASCIZ @CYLINDER (RMDC) a
25	077651	101	104	104 EMS27:	.ASCIZ @ADDRESS a
26	077662	123	124	101 EMS30:	.ASCIZ @STATUS a
27	077672	042	127	114 EMS31:	.ASCIZ @'WLE' (RMER1, BIT 11) a
28	077721	042	125	120 EMS32:	.ASCIZ @'UPE' (RMCS2, BIT 13) a
29	077750	042	127	103 EMS33:	.ASCIZ @'WCF' (RMER1, BIT 5) a
30	077776	127	122	111 EMS34:	.ASCIZ @WRITE CHECK ERROR-'WCE' (RMCS2, BIT 14) a
31	100047	042	115	104 EMS35:	.ASCIZ @'MDPE' (RMCS2, BIT 8) a
32	100076	042	104	103 EMS36:	.ASCIZ @'DCK' (RMER1, BIT 15) a
33	100125	042	105	103 EMS37:	.ASCIZ @'ECH' (RMER1, BIT 6) a
34	100153	042	104	114 EMS40:	.ASCIZ @'DLT' (RMCS2, BIT 15) a
35	100202	042	115	130 EMS41:	.ASCIZ @'MXF' (RMCS2, BIT 9) a
36	100230	042	104	124 EMS42:	.ASCIZ @'DTE' (RMER1, BIT 12) a
37	100257	042	110	103 EMS43:	.ASCIZ @'HCR' (RMER1, BIT 8) a
38	100306	110	105	101 EMS44:	.ASCIZ @HEADER COMPARE ERROR 'HCE' (RMER1, BIT 7) a
39	100361	106	117	122 EMS45:	.ASCIZ @FORMAT ERROR 'FER' (RMER1, BIT 4) a
40	100424	042	111	101 EMS46:	.ASCIZ @'IAE' (RMER1, BIT 10) a
41	100453	042	117	120 EMS47:	.ASCIZ @'DPI' (RMER1, BIT 13) a
42	100502	042	123	113 EMS50:	.ASCIZ @'SKI' (RMER2, BIT 14) a
43	100531	042	120	111 EMS51:	.ASCIZ @'PIP' (RMDS, BIT 13) a
44	100557	124	110	105 EMS52:	.ASCIZ @THE RM a
45	100567	104	105	124 EMS53:	.ASCIZ @DETECTED a
46	100601	101	124	040 EMS54:	.ASCIZ @AT AN UNEXPECTED a
47	100623	111	116	103 EMS55:	.ASCIZ @INCORRECT DATA DURING a
48	100652	111	116	126 EMS56:	.ASCIZ @INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
49	100727	104	125	122 EMS57:	.ASCIZ @DURING DATA TRANSFER a
50	100755	104	101	124 EMS60:	.ASCIZ @DATA READ a
51	100770	104	117	105 EMS61:	.ASCIZ @DOES NOT COMPARE WITH a
52	101017	104	101	124 EMS62:	.ASCIZ @DATA WRITTEN a
53	101035	042	120	101 EMS63:	.ASCIZ @'PAR' (RMER1, BIT 3) a
54	101063	111	123	040 EMS64:	.ASCIZ @IS INCORRECT a
55	101101	103	117	115 EMS65:	.ASCIZ @COMPOSITE ERROR 'ERR' (RMDS, BIT 14) a
56	101147	104	101	124 EMS66:	.ASCIZ @DATA PARITY ERROR 'DPE' (RMER2, BIT 3) a
57	101217	104	125	122 EMS67:	.ASCIZ @DURING SEEK COMMAND a

58	101244	111	123	040	EMS70:	.ASCIZ	@IS RESET @
59	101256	105	122	122	EMS71:	.ASCIZ	@ERRONEOUS @
60	101271	111	123	040	EMS72:	.ASCIZ	@IS SET @
61	101301	104	111	104	EMS73:	.ASCIZ	@DID NOT SET @
62	101316	104	111	104	EMS74:	.ASCIZ	@DID NOT RESET @
63	101335	114	117	123	EMS75:	.ASCIZ	@LOST @
64	101343	104	125	122	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
65	101374	042	122	115	EMS77:	.ASCIZ	@'RMR' (RMER1, BIT 2) @
66	101422	042	111	114	EMS100:	.ASCIZ	@'ILR' (RMER1, BIT 1) @
67	101450	042	111	114	EMS101:	.ASCIZ	@'ILF' (RMER1, BIT 0) @
68	101476	104	125	122	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
69	101525	105	122	122	EMS103:	.ASCIZ	@ERROR @
70	101534	042	114	102	EMS104:	.ASCIZ	@'LBC' (RMER2, BIT 10) @
71	101563	042	114	123	EMS105:	.ASCIZ	@'LSC' (RMER2, BIT 11) @
72	101612	110	105	101	EMS106:	.ASCIZ	@HEADER ERRORS @
73	101631	102	125	123	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
74	101660	101	104	104	EMS111:	.ASCIZ	@ADDRESS @
75	101671	127	110	105	EMS112:	.ASCIZ	@WHEN READING/WRITING RH REGISTERS @
76	101733	101	124	040		.ASCIZ	@AT THE FOLLOWING @
77	101755	124	110	105	EMS113:	.ASCIZ	@THE SELECTED DEVICE IS @
78	102005	116	117	116	EMS114:	.ASCIZ	@NONEXISTENT DEVICE @
79	102031	015	012	000	EMS115:	.ASCIZ	<CR><LF>
80	102034	124	110	105	EMS116:	.ASCIZ	@THE MASSBUS CONTROLLER @
81	102064	106	101	111	EMS117:	.ASCIZ	@FAILED TO DETECT @
82	102106	116	117	124	EMS120:	.ASCIZ	@NOT AN RM05/3/2 @
83	102127	103	117	116	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
84	102172	102	125	123	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMBA, @
85	102227	103	117	116	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
86	102272	105	122	122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMER1, @
87	102324	101	124	124	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMAS, @
88	102367	115	101	111	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMMR #1, @
89	102432	105	103	103	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
90	102471	105	103	103	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2, @
91	102527	115	101	111	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMMR2, @
92	102567	116	117	124	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
93	102613	125	116	111	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
94	102636	103	117	116	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
95	102700	122	110	057	EMS135:	.ASCIZ	@RH/RM ERROR CLEAR (RMCS1, BIT 14) @
96	102743	104	122	111	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
97	102770	104	122	111	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMDS @
98	103025	115	105	104	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
99	103046	042	115	117	EMS141:	.ASCIZ	@'MOL' (RMDS, BIT 12) @
100	103074	104	122	111	EMS142:	.ASCIZ	@DRIVE FAULT @
101	103111	042	104	126	EMS143:	.ASCIZ	@'DVC' (RMER2, BIT 7) @
102	103137	125	116	123	EMS144:	.ASCIZ	@UNSAFE @
103	103147	042	125	116	EMS145:	.ASCIZ	@'UNS' (RMER1, BIT 14) @
104	103176	123	110	117	EMS146:	.ASCIZ	@SHOULD BE SET @
105	103215	117	106	106	EMS147:	.ASCIZ	@OFFSET MODE @
106	103232	126	117	114	EMS150:	.ASCIZ	@VOLUME VALID @
107	103250	042	117	115	EMS151:	.ASCIZ	@'DM' (RMDS, BIT 0) @
108	103274	042	126	126	EMS152:	.ASCIZ	@'VV' (RMDS, BIT 6) @
109	103320	120	101	103	EMS153:	.ASCIZ	@PACK ACK COMMAND @
110	103342	116	117	124	EMS154:	.ASCIZ	@NOT SET BY @
111	103356	117	106	106	EMS155:	.ASCIZ	@OFFSET COMMAND @
112	103376	116	117	124	EMS156:	.ASCIZ	@NOT RESET BY @
113	103414	122	124	103	EMS157:	.ASCIZ	@RTC COMMAND @
114	103431	122	111	120	EMS160:	.ASCIZ	@RIP COMMAND @

ERROR MESSAGE STRINGS

115	103446	117	106	106	EMS161: .ASCIZ @OFFSET REGISTER (RMOF) @
116	103476				EMS162: ;<UNUSED>
117	103476	104	125	122	EMS163: .ASCIZ @DURING RECALIBRATE @
118	103522	111	123	040	EMS164: .ASCII @IS INTERMITTENT OR DRIVE DIDNT DROP ON @
119	103571	103	131	114	.ASCIZ @CYLINDER @
120	103603	125	116	105	EMS165: .ASCIZ @UNEXPECTED @
121	103617	122	105	103	EMS166: .ASCIZ @RECALIBRATE COMMAND @
122	103644	042	101	124	EMS167: .ASCIZ @'ATA' (RMDS, BIT15) @
123	103671	127	110	105	EMS170: .ASCIZ @WHEN READING REGISTER @
124	103720	105	122	122	EMS171: .ASCIZ @ERROR REGISTER #2, RMER2, @
125	103753	116	117	116	EMS172: .ASCIZ @NONRECOVERABLE @
126	103773	122	105	103	EMS173: .ASCIZ @RECOVERABLE @
127	104010	104	101	124	EMS174: .ASCIZ @DATA @
128	104016	104	125	122	EMS175: .ASCIZ @DURING WRITE COMMAND @
129	104044	042	117	120	EMS176: .ASCIZ @'DPE' (RMER2, BIT 13) @
130	104073	111	116	040	EMS177: .ASCIZ @IN WRITE PROTECT @
131	104115	103	101	116	EMS200: .ASCIZ @CAN NOT SET @
132	104132	104	111	101	EMS201: .ASCIZ @DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 0) @
133	104200	104	125	122	EMS202: .ASCIZ @DURING DIAGNOSTIC MODE @
134	104230	111	116	103	EMS203: .ASCIZ @INCORRECT @
135	104243	042	127	122	EMS204: .ASCIZ @'WRL' (RMDS, BIT 11) @
136	104271	105	130	105	EMS205: .ASCIZ @EXECUTED @
137	104303	127	111	124	EMS206: .ASCIZ @WITH COMP ERROR SET @
138	104330	042	107	117	EMS207: .ASCIZ @'GO' (RMCS1, BIT 0) @
139	104355	127	122	111	EMS210: .ASCIZ @WRITING @
140	104366	127	101	123	EMS211: .ASCIZ @WAS RESET BY @
141	104404	120	122	117	EMS212: .ASCIZ @PROGRAM INTERRUPT @
142	104427	127	101	123	EMS213: .ASCIZ @WAS NOT GENERATED @
143	104452	123	105	105	EMS214: .ASCIZ @SEEK COMMAND @
144	104470	120	122	117	EMS215: .ASCIZ @PROGRAM TIMEOUT @
145	104511	104	125	122	EMS216: .ASCIZ @DURING LOOK AHEAD TEST @
146	104541	114	117	117	EMS217: .ASCIZ @LOOK AHEAD REGISTER, RMLA, @
147	104574	123	105	101	EMS220: .ASCIZ @SEARCH COMMAND @
148	104614	102	101	104	EMS221: .ASCIZ @BAD SECTOR ERROR 'BSE' (RMER2, BIT 15) @
149	104664	101	040	104	EMS222: .ASCIZ @A DATA TRANSFER COMMAND @
150	104715	110	105	101	EMS223: .ASCIZ @HEADER COMPARE INHIBIT 'HCI' (RMOF, BIT 10) @
151	104772	116	117	116	EMS224: .ASCIZ @NONEXISTENT MEMORY 'NEM' (RMCS2, BIT 11) @
152					

1	105044	105	130	120	EH1:	.ASCIZ	@EXPCTD	RECEVD@			
2	105063	102	125	123	EH110:	.ASCIZ	@BUSADRA				
3	105072	040	122	115	EH111:	.ASCIZ	@ RMCS2	RMCS1@			
4											
5	105111	122	105	103	EH114:	.ASCIZ	@RECEVD	SNGPRT	DULPRT@		
6	105140	105	130	120	EH223:	.ASCIZ	@EXPCTD	RECEVD	DATA@		
7	105166	105	130	120	EH256:	.ASCII	@EXPCTD	RECEVD	RGSTR@<CRLF>		
8	105214	123	124	101		.ASCIZ	@STATUS	STATUS	INDEX@		
9											
10	105242	107	104	101	EH336:	.ASCIZ	@GDADRS	GDDATA	BDADRS	BDDATA@	
11	105301	122	115	103	EH337:	.ASCII	@RMCS2	STATUS	FAILING	DATA@<CRLF>	
12	105340	137	137	137		.ASCII	@			@<CRLF>	
13	105377	105	130	120		.ASCIZ	@EXPCTD	RECEVD	BIT	ADRESS@	
14											
15	105436	122	115	105	EH344:	.ASCII	@RMER1	STATUS	HEADER	FAILING@<CRLF>	
16	105476	137	137	137		.ASCII	@		WORD	BIT@<CRLF>	
17	105534	105	130	120		.ASCIZ	@EXPCTD	RECEVD	NUMBER	POSITON@	
18	105574	105	130	120	EH353:	.ASCII	@EXPCTD	RECEVD@<CRLF>			
19	105613	074	103	122		.ASCIZ	@<CRLF>	RMLA	RMLA	RMOF @	
20											
21	105650	040	122	115	STSH1:	.ASCII	@ RMCS1	RMCS2	RMDS	RMER1	RMER2@
22	105715	040	040	040		.ASCIZ	@	RMAS@			
23	105725	040	122	115	STSH2:	.ASCII	@ RMWC	RMBA	RMDA	RMOF	RMDC@
24	105772	040	040	040		.ASCIZ	@	RMEC1	RMEC2@		
25	106014	040	122	115	STSH3:	.ASCIZ	@ RMDA	RMDC	RMOF	RMLA@	
26	106052	040	122	115	STSH4:	.ASCIZ	@ RMMR1	RMMR2	RMDT	RMSN@	
27											
28					.EVEN						

1	106110	001140	001142	000000	ED1:	.WORD	\$GDDAT,\$BDDAT,0
2	106116	001276	000000		ED110:	.WORD	\$BASE,0
3	106122	001174	001176	000000	ED111:	.WORD	\$TMP0,\$TMP1,0
4							
5	106130	001362	001176	001200	ED114:	.WORD	RMDTI,\$TMP1,\$TMP2,0
6	106140	001140	001142	001174	ED223:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,0
7							
8	106150	001134	001140	001136	ED336:	.WORD	\$GDADR,\$GDDAT,\$BDADR,\$BDDAT,0
9							
10	106162	001140	001142	001174	ED337:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,\$TMP1,0
11	106174	001140	001142	001442	ED353:	.WORD	\$GDDAT,\$BDDAT,RMOFO,0
12							
13	106204	001334	001344	001346	STSD1:	.WORD	RMCS1I,RMCS2I,RMDSI,RMER1I,RMER2I,RMASI,0
14	106222	001336	001340	001342	STSD2:	.WORD	RMWCI,RMBAI,RMDAI,RMOFI,RMDCI,RMECI
15	106236	001402	000000			.WORD	RMEC2I,0
16	106242	001342	001370	001366	STSD3:	.WORD	RMDAI,RMDCI,RMOFI,RMLAI,0
17	106254	001360	001374	001362	STSD4:	.WORD	RMMR1I,RMMR2I,RMDTI,RMSNI,0
18							

1	106266	000			EF110:	.BYTE	0
2	106267	000	000		EF111:	.BYTE	0,0
3	106271	000	000	000	EF114:	.BYTE	0,0,0
4	106274	000	000	000	EF336:	.BYTE	0,0,0,0
5	106300	000	000	000	EF337:	.BYTE	0,0,0,0,0
6							
7	106305	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0
8					.EVEN		
9							

```
1          ;STORAGE FOR GENERAL DATA TRANSFERRS
2 106314  BUFFER:
3 106314  BUFONE: .BLKW 258.
4 107320  BUFTWO: .BLKW 258.
5
6          ;STORAGE FOR MANUFACTURES 16 BIT MODE BAD SECTOR FILE
7 110324 000000 000000  MFGFIL: .WORD 0,0          ;2 HEADER WORDS
8          .BLKW 256.          ;256. WORDS OF DATA
9 111330 177777          .WORD -1          ;TERMINATOR IF FILE IS FULL
10
11         ;STORAGE FOR USERS 16 BIT MODE BAD SECTOR FILE
12 111332 000000 000000  USRFIL: .WORD 0,0          ;2 HEADER WORDS
13          .BLKW 256.          ;256. WORDS OF DATA
14 112336 177777          .WORD -1          ;TERMINATOR IF FILE IS FULL
15
16         .=BUFFER
17
18 106314  HELP:
19 106314 200          .ASCII <CRLF>
20 106315 200          .ASCII <CRLF>
21 106316 114 111 123 .ASCII @LIST OF TESTS@<CRLF>
22 106334 055 055 055 .ASCII @-----@<CRLF>
23 106352 124 061 011 .ASCII @T1 CONTROLLER ACCESS TEST@<CRLF>
24 106404 124 062 011 .ASCII @T2 DEVICE AVAILABLE TEST@<CRLF>
25 106435 124 063 011 .ASCII @T3 DRIVE TYPE TEST@<CRLF>
26 106460 124 064 011 .ASCII @T4 WRITE, READ ZEROS@<CRLF>
27 106505 124 065 011 .ASCII @T5 WRITE, WRITE CHECK ZEROS@<CRLF>
28 106541 124 066 011 .ASCII @T6 WRITE, WRITE CHECK ZEROS W/ WCE ERROR@<CRLF>
29 106612 124 067 011 .ASCII @T7 WRITE, READ ONES@<CRLF>
30 106636 124 061 060 .ASCII @T10 WRITE, WRITE CHECK ONES@<CRLF>
31 106672 124 061 061 .ASCII @T11 WRITE, WRITE CHECK ONES W/ WCE ERROR@<CRLF>
32 106743 124 061 062 .ASCII @T12 WRITE, WRITE CHECK MULTIPLE SECTORS@<CRLF>
33 107013 124 061 063 .ASCII @T13 WRITE, READ W/ IMPLIED SEEK@<CRLF>
34 107053 124 061 064 .ASCII @T14 WRITE, WRITE CHECK W/ HEAD SWITCHING@<CRLF>
35 107124 124 061 065 .ASCII @T15 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK@<CRLF>
36 107200 124 061 066 .ASCII @T16 WRITE, READ W/ HCE ERROR@<CRLF>
37 107235 124 061 067 .ASCII @T17 WRITE, READ W/ HCIA@<CRLF>
38 107264 124 062 060 .ASCII @T20 WRITE, READ W/ IVC ERROR@<CRLF>
39 107321 124 062 061 .ASCII @T21 WRITE, READ W/ ABORT@<CRLF>
40 107352 124 062 062 .ASCII @T22 WRITE, READ EACH CURRENT LEVEL@<CRLF>
41 107415 124 062 063 .ASCII @T23 WRITE, WRITE CHECK W/ CURRENT LEVEL SWITCHING@<CRLF>
42 107477 124 062 064 .ASCII @T24 WRITE, READ EARLY PEAK SHIFT@<CRLF>
43 107540 124 062 065 .ASCII @T25 WRITE, READ MIXED PEAK SHIFT@<CRLF>
44 107601 124 062 066 .ASCII @T26 WRITE, READ EACH TRACK@<CRLF>
45 107634 124 062 067 .ASCII @T27 READ, WRITE CHECK MULTIPLE SECTORS IN OFFSET MODE@<CRLF>
46 107722 200          .ASCII <CRLF>
47 107723 117 120 105 .ASCII @OPERATIONAL SWITCH SETTINGS@<CRLF>
48 107757 055 055 055 .ASCII @-----@<CRLF>
49 110013 123 127 111 .ASCII @SWITCH USE@<CRLF>
50 110030 055 055 055 .ASCII @-----@<CRLF>
51 110065 040 040 061 .ASCII @ 15 HALT ON ERROR@<CRLF>
52 110111 040 040 061 .ASCII @ 14 LOOP ON TEST@<CRLF>
53 110134 040 040 061 .ASCII @ 13 INHIBIT ERROR TYPEOUTS@<CRLF>
54 110171 040 040 061 .ASCII @ 12 @<CRLF>
55 110200 040 040 061 .ASCII @ 11 INHIBIT ITERATIONS@<CRLF>
56 110231 040 040 061 .ASCII @ 10 BELL ON ERROR@<CRLF>
57 110255 040 040 040 .ASCII @ 9 LOOP ON ERROR@<CRLF>
```


58	110301	040	040	040	.ASCII	@	8	LOOP ON TEST IN SWR<7:0>@<CRLF>
59	110340	040	040	040	.ASCII	@	7	TN128@<CRLF>
60	110354	040	040	040	.ASCII	@	6	TN64@<CRLF>
61	110367	040	040	040	.ASCII	@	5	TN32@<CRLF>
62	110402	040	040	040	.ASCII	@	4	TN16@<CRLF>
63	110415	040	040	040	.ASCII	@	3	TN8@<CRLF>
64	110427	040	040	040	.ASCII	@	2	TN4@<CRLF>
65	110441	040	040	040	.ASCIZ	@	1	TN2@<CRLF>
66								
67	000200				.END		200	

ABASE = 176700
ACDW1 = 000000
ACDW2 = 000000
ACKSTS = 054602
ACPUOP = 000000
ADDW0 = 000000
ADDW1 = 000000
ADDW10 = 000000
ADDW11 = 000000
ADDW12 = 000000
ADDW13 = 000000
ADDW14 = 000000
ADDW15 = 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT = 000000
ADEVM = 000000
ADR = 000001
AENV = 000000
AENVM = 000000
AFATAL = 000000
ALL = 070331
AMADR1 = 000000
AMADR2 = 000000
AMADR3 = 000000
AMADR4 = 000000
AMAMS1 = 000000
AMAMS2 = 000000
AMAMS3 = 000000
AMAMS4 = 000000
AMSGAD = 000000
AMSGLG = 000000
AMSGTY = 000000
AMTYP1 = 000000
AMTYP2 = 000000
AMTYP3 = 000000
AMTYP4 = 000000
AOE = 001000
APASS = 000000
APE = 100000
APRIOR = 000000
APTC SU = 000040
APTENV = 000001
APTSIZ = 000200
APTSPO = 000100
ARGS = 000004
ASNDA = 001514
ASNDC = 001512
ASWREG = 000000
ATA = 100000
ATESTN = 000000

ATNMSK = 000377
ATNTBL = 071616
AUNIT = 000000
AUSWR = 000000
AVECT1 = 120254
AVECT2 = 000000
A16 = 000400
A17 = 001000
BACK = 000001
BADSC T = 041416
BADTMO = 005340
BAI = 000010
BB00 = 000001
BB01 = 000002
BB02 = 000004
BB03 = 000010
BB04 = 000020
BB05 = 000040
BB06 = 000100
BB07 = 000200
BB08 = 000400
BB09 = 001000
BIT0 = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1 = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2 = 000004
BIT3 = 000010
BIT4 = 000020
BIT5 = 000040
BIT6 = 000100
BIT7 = 000200
BIT8 = 000400
BIT9 = 001000
BOTADR = 040430
BOTFLG = 040432
BPTVEC = 000014
BSE = 100000
BUFFER = 106314
BUFONE = 106314
BUFTWO = 107320
CC = 004000
CH = 002000

CHRCNT = 040433
CKSWR = 104410
CLKADR = 001516
CLKVCT = 001520
CLR = 000040
CLRSTS = 053722
CMNSTA = 007612
CMPBUF = 043602
CMPERR = 051720
CNSL00 = 070755
CNSL01 = 071010
CNSL02 = 071026
CNSL03 = 071070
CNSL04 = 071110
CNSL05 = 071144
CNSL06 = 071156
CNSL07 = 071207
CNSL08 = 071352
CNSL09 = 071353
CNTCLR = 053604
COMMA = 070342
CONT = 000100
CR = 000015
CRLF = 000200
CTLFG = 001326
CYLSK = 001777
DBCK = 100000
DBEN = 040000
DBL = 002000
DCK = 100000
DDISP = 177570
DEBL = 020000
DEVSEL = 052132
DISPLA = 001156
DISPRE = 000174
DLT = 100000
DMD = 000001
DPE = 000010
DPEHI = 040000
DPELO = 020000
DPR = 000400
DRQ = 004000
DRVCLR = 000010
DRVSTS = 057140
DRY = 000200
DSWR = 177570
DTASTS = 057742
DTE = 010000
DTO = 010000
DULPRT = 024024
DVA = 004000
DVC = 000200
EARLY = 072142
EBL = 020000
ECH = 000100
ECI = 004000
ECRC = 001000

EDT1 = 076472
EDT110 = 076510
EDT111 = 076512
EDT114 = 076514
EDT2 = 076502
EDT223 = 076516
EDT336 = 076526
EDT337 = 076536
EDT344 = 076546
EDT353 = 076560
ED1 = 106110
ED110 = 106116
ED111 = 106122
ED114 = 106130
ED223 = 106140
ED336 = 106150
ED337 = 106162
ED353 = 106174
EECC = 000020
EFT1 = 076562
EFT110 = 076600
EFT111 = 076602
EFT114 = 076604
EFT2 = 076572
EFT223 = 076606
EFT336 = 076616
EFT337 = 076626
EFT344 = 076636
EFT353 = 076646
EF110 = 106266
EF111 = 106267
EF114 = 106271
EF336 = 106274
EF337 = 106300
EHT1 = 076346
EHT110 = 076370
EHT111 = 076374
EHT114 = 076400
EHT2 = 076360
EHT223 = 076404
EHT256 = 076416
EHT336 = 076430
EHT337 = 076442
EHT344 = 076454
EHT353 = 076466
EH1 = 105044
EH110 = 105063
EH111 = 105072
EH114 = 105111
EH223 = 105140
EH256 = 105166
EH336 = 105242
EH337 = 105301
EH344 = 105436
EH353 = 105574
EMS1 = 076650
EMS10 = 077204

EMS100 = 101422
EMS101 = 101450
EMS102 = 101476
EMS103 = 101525
EMS104 = 101534
EMS105 = 101563
EMS106 = 101612
EMS11 = 077246
EMS110 = 101631
EMS111 = 101660
EMS112 = 101671
EMS113 = 101755
EMS114 = 102005
EMS115 = 102031
EMS116 = 102034
EMS117 = 102064
EMS12 = 077257
EMS120 = 102106
EMS121 = 102127
EMS122 = 102172
EMS123 = 102227
EMS124 = 102272
EMS125 = 102324
EMS126 = 102367
EMS127 = 102432
EMS13 = 077320
EMS130 = 102471
EMS131 = 102527
EMS132 = 102567
EMS133 = 102613
EMS134 = 102636
EMS135 = 102700
EMS136 = 102743
EMS137 = 102770
EMS14 = 077331
EMS140 = 103025
EMS141 = 103046
EMS142 = 103074
EMS143 = 103111
EMS144 = 103137
EMS145 = 103147
EMS146 = 103176
EMS147 = 103215
EMS15 = 077342
EMS150 = 103232
EMS151 = 103250
EMS152 = 103274
EMS153 = 103320
EMS154 = 103342
EMS155 = 103356
EMS156 = 103376
EMS157 = 103414
EMS16 = 077364
EMS160 = 103431
EMS161 = 103446
EMS162 = 103476
EMS163 = 103476

EMT353 076330	FNCMSK= 000077	IR = 000100	PHA = 000200	RMDAI 001342
EMT354 076340	F0 = 000002	IVC = 010000	PIP = 020000	RMDAO 001416
EMT36 072570	F1 = 000004	LBC = 002000	PIRQ = 177772	RMDB = 000022
EMT37 072600	F2 = 000010	LBT = 002000	PIRQVE= 000240	RMDBI 001356
EMT4 072274	F3 = 000020	LF = 000012	PLFS = 002000	RMDBO 001432
EMT40 072610	F4 = 000040	LODEV 071416	PRIERR 045226	RMDC = 000034
EMT41 072616	GENBUF 043344	LS = 000004	PRO = 000000	RMDCI 001370
EMT42 072626	GET 044230	LSC = 004000	PR1 = 000040	RMDCO 001444
EMT43 072640	GETBUF 001334	LST = 000002	PR2 = 000100	RMDS = 000012
EMT44 072650	GETINX 001522	LSTRK 001332	PR3 = 000140	RMDSI 001346
EMT45 072660	GETSTS 044144	MCLK = 004000	PR4 = 000200	RMDSO 001422
EMT46 072670	GO = 000001	MCPE = 020000	PR5 = 000240	RMDT = 000026
EMT47 072700	GTSWR = 104407	MDF = 000100	PR6 = 000300	RMDTI 001362
EMT5 072302	HCE = 000200	MDPE = 000400	PR7 = 000340	RMDTO 001436
EMT50 072706	HCI = 002000	MEDENB 001510	PS = 177776	RMEC1 = 000044
EMT51 072716	HCRC = 000400	MFGFIL 110324	PSEL = 002000	RMEC1I 001400
EMT52 072730	HELP 106314	MI = 000004	PSW = 177776	RMEC10 001454
EMT53 072746	HT = 000011	MIXED 071626	PUT 044500	RMEC2 = 000046
EMT54 072764	IAE = 002000	MOC = 000400	PUTBUF 001410	RMEC2I 001402
EMT55 072774	IDXMSK= 000077	MOH = 020000	PUTINX 001551	RMEC20 001456
EMT56 073006	IE = 000100	MOL = 010000	PWRVEC= 000024	RMER1 = 000014
EMT57 073024	ILF = 000001	MRD = 002000	QUES 070336	RMER1I 001350
EMT6 072310	ILF02 = 000002	MS = 000040	RCLSTS 055376	RMER10 001424
EMT60 073034	ILF24 = 000024	MSC = 000002	RD = 000070	RMER2 = 000042
EMT61 073052	ILF26 = 000026	MSDRVS 071374	RDCHR = 104411	RMER2I 001376
EMT62 073072	ILF30 = 000030	MSE = 100000	RDLIN = 104412	RMER20 001452
EMT63 073106	ILF32 = 000032	MSER = 000200	RDICT = 104413	RMHR = 000036
EMT64 073110	ILF34 = 000034	MSGDRV 071410	RDY = 000200	RMHRI 001372
EMT65 073130	ILF36 = 000036	MSHELP 070345	READY 007764	RMHRO 001446
EMT66 073150	ILF40 = 000040	MUR = 001000	RECAL = 000006	RMLA = 000020
EMT67 073162	ILF42 = 000042	MWD = 000010	RESREG= 104415	RMLAI 001354
EMT7 072316	ILF44 = 000044	MWP = 000010	RESVEC= 000010	RMLAO 001430
EMT70 073174	ILF46 = 000046	MXF = 001000	REX = 010000	RMMR1 = 000024
EMT71 073204	ILF54 = 000054	NDTMSK= 115760	RG = 040000	RMMR1I 001360
EMT72 073214	ILF56 = 000056	NED = 010000	RGDTPT 071626	RMMR10 001434
EMT73 073232	ILF64 = 000064	NEM = 004000	RH = 000072	RMMR2 = 000040
EMT74 073250	ILF66 = 000066	NOP = 000000	RIP = 000020	RMMR2I 001374
EMT75 073260	ILF74 = 000074	NOTAVL 071453	RELEASE= 000012	RMMR20 001450
EMT76 073300	ILF76 = 000076	NOTPRS 071436	RMAS = 000016	RMOF = 000032
EMT77 073310	ILR = 000002	NSA = 100000	RMASI 001352	RMOFI 001366
ENRGDT 072254	ILRG50= 000050	OCC = 100000	RMASO 001426	RMOFO 001442
EQUALS 070327	ILRG52= 000052	OFD = 000200	RMBA = 000004	RMR = 000004
ERR = 040000	ILRG54= 000054	OFFLIN 071472	RMBAE = 000050	RMSN = 000030
ERRMB 040426	ILRG56= 000056	OFFSET= 000014	RMBAEI 001404	RMSNI 001364
ERROR = 104000	ILRG60= 000060	OM = 000001	RMBAEO 001460	RMSNO 001440
ERRTYP 037670	ILRG62= 000062	ONES 071666	RMBAI 001340	RMWC = 000002
ERRVEC= 000004	ILRG64= 000064	ONLINE 071504	RMBAO 001414	RMWCI 001336
ERTY00 040434	ILRG66= 000066	OPE = 020000	RMCS1 = 000000	RMWCO 001412
ERTY01 040442	ILRG70= 000070	OPI = 020000	RMCS1I 001334	RQA = 100000
ERTY02 040452	ILRG72= 000072	OR = 000200	RMCS10 001410	RQB = 040000
ERTY03 040461	ILRG74= 000074	PACACK= 000022	RMCS2 = 000010	RTC = 000016
ERTY04 040467	ILRG76= 000076	PAKACK= 000022	RMCS2I 001344	R6 = 2000006
ESRC = 004000	IOTVEC= 000020	PAR = 000010	RMCS20 001420	R7 = 2000007
FER = 000020	IPCK0 = 000001	PAT = 000020	RMCS3 = 000052	SADMSK= 000377
FIND = 000001	IPCK1 = 000002	PDA = 000400	RMCS3I 001406	SAVREG= 104414
FMT16 = 010000	IPCK2 = 000004	PGE = 002000	RMCS30 001462	SA1 = 000001
FNCDTB 071516	IPCK3 = 000010	PGM = 001000	RMDA = 000006	SA16 = 000020

SA2 = 000002	SW3 = 000010	UNS = 040000	\$DOAGN 037660	\$MTYP4 001267
SA4 = 000004	SW4 = 000020	UNTMSK= 000007	\$DTBL 064540	\$MXCNT 065656
SA8 = 000010	SW5 = 000040	UPE = 020000	\$ENDAD 037650	\$NULL 001170
SC = 100000	SW6 = 000100	USE = 040000	\$ENDCT 037514	\$NWTST= 000001
SCOPE = 000004	SW7 = 000200	USRFIL 111332	\$ENULL 037664	\$OCNT 065002
SCTMSG 070230	SW8 = 000400	UO = 000001	\$ENV 001242	\$OMODE 065004
SCTMSK= 003700	SW9 = 001000	U1 = 000002	\$ENVM 001243	\$OVER 065642
SCO = 000100	TADMSK= 177400	U2 = 000004	\$EOP 037460	\$PASS 001230
SC1 = 000200	TAG = 020000	VV = 000100	\$EOPCT 037506	\$PASTM 001106
SC2 = 000400	TAGADR= 001114	WC = 000040	\$EOSP 037422	\$POWER 067752
SC3 = 001000	TAP = 040000	WCD = 000050	\$ERFLG 001117	\$PWRDN 067604
SC4 = 002000	TA1 = 000400	WCE = 040000	\$ERMAX 001131	\$PWRMG 067740
SLARCH= 000030	TA16 = 010000	WCEHI = 010000	\$ERROR 065736	\$PWRUP 067656
SECERR 046060	TA2 = 001000	WCELO = 004000	\$ERRPC 001132	\$QUES 001216
SEEK = 000004	TA4 = 002000	WCF = 000040	\$ERRTB 001600	\$RDCHR 066776
SEKSTS 052344	TAB = 004000	WCH = 000052	\$ERTTL 001126	\$RDLIN 067066
SHUT 064076	TBITVE= 000014	WD = 000060	\$ESCAP 001210	\$RDOCT 067374
SHUT2 064154	TIMOUT 045042	WH = 000062	\$ETABL 001242	\$RDSZ = 000010
SIZCLK 044720	TKVEC = 000060	WLE = 004000	\$ETEND 001326	\$RESRE 064222
SKI = 040000	TPVEC = 000064	WRL = 004000	\$FATAL 001224	\$RTNAD 037662
SNGPRT= 020024	TRAPVE= 000034	XSIZ 006462	\$FFLG 070226	\$SAVRE 064164
STACK = 001100	TRE = 040000	XXDP 001330	\$FILLC 001172	\$SAVR6 067750
STANDA 006706	TRTVEC= 000014	ZEROS 071730	\$FILLS 001171	\$SCOPE 065342
START 005420	TST = 010000	\$APTHD 001100	\$GDADR 001134	\$SETUP= 000137
STCDRV 063342	TSTNMB 040424	\$ATYC 070006	\$GDDAT 001140	\$STUP = 177777
STKLMT= 177774	TSTPRP 040472	\$ATY1 067762	\$GET42 037640	\$SVLAD 065606
STOP 064046	TSTQUE 001464	\$ATY3 067770	\$GTSWR 066524	\$SVPC = 000204
STSD1 106204	TST1 010130	\$ATY4 070000	\$HD = 000000	\$SWR = 167400
STSD2 106222	TST10 015070	\$AUTOB 001150	\$HIBTS 001100	\$SWREG 001244
STSD3 106242	TST11 016046	\$BASE 001276	\$HIOCT 067474	\$SWRMK= 000000
STSD4 106254	TST12 017244	\$BDADR 001136	\$ICNT 001120	\$SW08 065660
STSF 106305	TST13 020250	\$BDDAT 001142	\$ILLUP 067744	\$TESTN 001226
STSH1 105650	TST14 021474	\$BELL 001212	\$INTAG 001151	\$TIMES 001206
STSH2 105725	TST15 022702	\$BIN 064332	\$ITEMB 001130	\$TKB 001162
STSH3 106014	TST16 023724	\$CDW1 001302	\$LF 001220	\$TKCNT 066136
STSH4 106052	TST17 025572	\$CDW2 001304	\$LFLG 070225	\$TKINT 066146
\$WR 001154	TST2 010314	\$CHARC 065336	\$LPADR 001122	\$TKQEN= 066145
\$WREG 000176	TST20 027176	\$CKSWR 066434	\$LPERR 001124	\$TKQIN 066140
\$W0 = 000001	TST21 030216	\$CMTAG 001114	\$MADR1 001254	\$TKQOU 066142
\$W00 = 000001	TST22 031134	\$CM3 = 000000	\$MADR2 001260	\$TKQSR 066144
\$W01 = 000002	TST23 032172	\$CM4 = 000005	\$MADR3 001264	\$TKS 001160
\$W02 = 000004	TST24 033176	\$CNTLC 067332	\$MADR4 001270	\$TKSRV 066216
\$W03 = 000010	TST25 034204	\$CNTLG 067344	\$MAIL 001222	\$TMP0 001174
\$W04 = 000020	TST26 035212	\$CNTLU 067337	\$MAMS1 001252	\$TMP1 001176
\$W05 = 000040	TST27 036240	\$CPUOP 001250	\$MAMS2 001256	\$TMP2 001200
\$W06 = 000100	TST3 010500	\$CRLF 001217	\$MAMS3 001262	\$TMP3 001202
\$W07 = 000200	TST4 010702	\$DBLK 064550	\$MAMS4 001266	\$TMP4 001204
\$W08 = 000400	TST5 011710	\$DDW0 001306	\$MBADR 001102	\$TN = 000030
\$W09 = 001000	TST6 012666	\$DDW1 001310	\$MFLG 070224	\$TPB 001166
\$W1 = 000002	TST7 014062	\$DDW2 001312	\$MNEW 067362	\$TPFLG 001173
\$W10 = 002000	TYPBN = 104406	\$DDW3 001314	\$MSGAD 001236	\$TPS 001164
\$W11 = 004000	TYPDS = 104405	\$DDW4 001316	\$MSGLG 001240	\$TRAP 067476
\$W12 = 010000	TYPE = 104401	\$DDW5 001320	\$MSGTY 001222	\$TRAP2 067536
\$W13 = 020000	TYPOC = 104402	\$DDW6 001322	\$MSWR 067351	\$TRP = 000016
\$W14 = 040000	TYPON = 104404	\$DDW7 001324	\$MTYP1 001253	\$TRPAD 067550
\$W15 = 100000	TYPOS = 104403	\$DEVCT 001232	\$MTYP2 001257	\$STM 001104
\$W2 = 000004	UBUSQS 070722	\$DEVN 001300	\$MTYP3 001263	\$STNM 001116

STYIN 067322
STYPBN 064260
STYPDS 064334
STYPE 065006
STYPEC 065220

STYPEX 065340
STYPOC 064604
STYPON 064620
STYPOS 064560

SUNIT 001234
SUNITM 001110
SUSWR 001246
SVECT1 001272

SVECT2 001274
\$XOFF = 000023
\$XON = 000021
\$XTSTR 065360

\$\$GET4= 000000
\$\$SW08= 000030
\$OFILL 065003
\$.SX = 001100

. ABS. 112340 000
 000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 55976 WORDS (219 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
CZRMOA.BIC,CZRMOA/C=CZRMOA.DOC,CZRMOA,SYSMAC/M

SITEME 5-0# 14-29 38-7 38-7 38-7 38-7*

G 8

SEQ 0303

SCOPE 9-19 38-6#

1 8

SEQ 0305

38-10 38-10 38-10 38-10 38-10 38-10 38-10^k 38-10⁸ 38-10 38-10 38-10 38-10# 38-10# 38-10# 38-10#
SEQ 0307

ALL 10-121 39-6#

M 8

SEQ 0309

RECEIVED
FBI
WASHINGTON
D.C.
OCT 12 1964

AMADR1	5-0	5-0															
AMADR2	5-0	5-0															
AMADR3	5-0	5-0															
AMADR4	5-0	5-0															
AMAMS1	5-0	5-0															
AMAMS2	5-0	5-0															
AMAMS3	5-0	5-0															
AMAMS4	5-0	5-0															
AMSGAD	5-0	5-0															
AMSLG	5-0	5-0															
AMSGTY	5-0	5-0															
AMTYP1	5-0	5-0															
AMTYP2	5-0	5-0															
AMTYP3	5-0	5-0															
AMTYP4	5-0	5-0															
AOE	4-579#	4-590	25-311	25-326	25-341	25-356	25-360	26-117	26-121	40-77	40-78	40-81	40-82	40-85			
APASS	40-86																
APE	5-0	5-0															
APRIOR	4-757#																
APTCSU	5-0																
APTENV	38-5	38-12#															
APTSIZ	38-5	38-7	38-12	38-12#													
APTSPD	9-19	38-12#															
ARGS	38-5	38-12	38-12#														
	12-99	12-99#	12-110#	12-126#	12-129#	12-131#	12-134#	12-138#	12-151	12-151#	12-168#	12-171#	12-173#	12-176#			
	12-180#	12-182#	12-184#	12-197	12-197#	12-213#	12-216#	12-218#	12-221#	12-225#	12-227#	12-229#	12-233#	12-246			
	12-246#	12-257#	12-273#	12-276#	12-278#	12-281#	12-285#	12-298	12-298#	12-315#	12-318#	12-320#	12-323#	12-327#			
	12-329#	12-331#	12-344	12-344#	12-359#	12-362#	12-364#	12-367#	12-371#	12-373#	12-375#	12-388	12-388#	12-399#			
	12-414#	12-417#	12-419#	12-422#	12-426#	12-439	12-439#	12-456#	12-459#	12-461#	12-464#	12-468#	12-470#	12-472#			
	12-487	12-487#	12-502#	12-505#	12-507#	12-510#	12-514#	12-519#	12-547#	12-563	12-563#	12-574#	12-592#	12-595#			
	12-597#	12-600#	12-604#	12-618	12-618#	12-635#	12-638#	12-640#	12-643#	12-647#	12-649#	12-651#	12-664	12-664#			
	12-680#	12-683#	12-685#	12-688#	12-692#	12-694#	12-696#	12-698#	12-711	12-711#	12-722#	12-740#	12-743#	12-745#			
	12-748#	12-752#	12-766	12-766#	12-783#	12-786#	12-788#	12-791#	12-795#	12-797#	12-799#	12-812	12-812#	12-827#			
	12-830#	12-832#	12-835#	12-839#	12-841#	12-843#	12-856	12-856#	12-867#	12-885#	12-888#	12-890#	12-893#	12-897#			
	12-910	12-910#	12-927#	12-930#	12-932#	12-935#	12-939#	12-941#	12-943#	12-958	12-958#	12-973#	12-976#	12-978#			
	12-981#	12-985#	12-989#	12-:19#	12-:35	12-:35#	12-:46#	12-:64#	12-:67#	12-:69#	12-:72#	12-:76#	12-:98	12-:98#			
	12-:13#	12-:16#	12-:18#	12-:21#	12-:25#	12-:27#	12-:29#	12-:42	12-:42#	12-:57#	12-:60#	12-:62#	12-:65#	12-:69#			
	12-:71#	12-:73#	12-:86	12-:86#	12-:98#	12-<16#	12-<19#	12-<21#	12-<24#	12-<28#	12-<41	12-<41#	12-<46#	12-<49#			
	12-<51#	12-<54#	12-<58#	12-<76#	12-<79#	12-<82#	12-<86#	12-<88#	12-<90#	12-=03	12-=03#	12-=08#	12-=11#	12-=13#			
	12-=16#	12-=20#	12-=37#	12-=40#	12-=43#	12-=47#	12-=49#	12-=51#	12-=53#	12-=78#	12-=87	12-=87#	12->01#	12->04#			
	12->06#	12->09#	12->13#	12->34	12->34#	12->38#	12->41#	12->43#	12->46#	12->50#	12->65#	12->68#	12->70#	12->73#			
	12->77#	12->79#	12->81#	12->94	12->94#	12->98#	12-?01#	12-?03#	12-?06#	12-?10#	12-?25#	12-?28#	12-?30#	12-?33#			
	12-?37#	12-?39#	12-?41#	12-?61#	12-?70	12-?70#	12-?83#	12-?86#	12-?88#	12-?91#	12-?95#	12-@16	12-@16#	12-@31#			
	12-@34#	12-@36#	12-@39#	12-@43#	12-@45#	12-@47#	12-@60	12-@60#	12-@75#	12-@78#	12-@80#	12-@83#	12-@87#	12-@89#			
	12-@91#	12-A06	12-A06#	12-A17#	12-A41#	12-A44#	12-A46#	12-A49#	12-A53#	12-A66	12-A66#	12-A83#	12-A86#	12-A88#			
	12-A91#	12-A95#	12-B12#	12-B28#	12-B40#	12-B49#	12-B61	12-B61#	12-B78#	12-B81#	12-B83#	12-B86#	12-B90#	12-C05#			
	12-C21#	12-C33#	12-C42#	12-C65	12-C65#	12-C76#	12-C92	12-C92#	12-D03#	12-D29#	12-D32#	12-D34#	12-D37#	12-D41#			
	12-D54	12-D54#	12-D71#	12-D74#	12-D76#	12-D79#	12-D83#	12-E02#	12-E04#	12-E17	12-E17#	12-E33#	12-E36#	12-E38#			
	12-E41#	12-E45#	12-E64#	12-E66#	12-E68#	12-E90	12-E90#	12-F01#	12-F21#	12-F28	12-F28#	12-F49#	12-F52#	12-F54#			
	12-F57#	12-F61#	12-F66#	12-F74#	12-F87	12-F87#	12-G09#	12-G12#	12-G14#	12-G17#	12-G21#	12-G26#	12-G34#	12-G54			
	12-G54#	12-G73#	12-G76#	12-G79#	12-G89#	12-G92#	12-G94#	12-G96#	12-H09	12-H09#	12-H28#	12-H31#	12-H34#	12-H45#			
	12-H48#	12-H50#	12-H52#	12-H64	12-H64#	12-H75#	12-H91#	12-H94#	12-H96#	12-H99#	12-I03#	12-I17	12-I17#	12-I35#			
	12-I38#	12-I40#	12-I43#	12-I47#	12-I49#	12-I53#	12-I66	12-I66#	12-I84#	12-I87#	12-I89#	12-I92#	12-I96#	12-I98#			
	12-J02#	12-J04#	12-J18	12-J18#	12-J30#	12-J46#	12-J49#	12-J51#	12-J54#	12-J58#	12-J72	12-J72#	12-J93#	12-J96#			
	12-J98#	12-K01#	12-K05#	12-K07#	12-K11#	12-K24	12-K24#	12-K40#	12-K42#	12-K45#	12-K48#	12-K52#	12-K54#	12-K58#			
	12-K71	12-K71#	12-K82#	12-K98#	12-L01#	12-L03#	12-L06#	12-L10#	12-L25	12-L23#	12-L40#	12-L43#	12-L45#	12-L48#			

	12-M16#	12-M27#	12-M43#	12-M46#	12-M48#	12-M51#	12-M55#	12-M68	12-M68#	12-M85#	12-M88#	12-M90#	12-M93#	12-M97#
	12-M99#	12-N01#	12-N14	12-N14#	12-N30#	12-N33#	12-N35#	12-N38#	12-N42#	12-N44#	12-N46#	12-N48#	12-N61	12-N61#
	12-N73#	12-N91#	12-N94#	12-N96#	12-N99#	12-003#	12-016	12-016#	12-033#	12-036#	12-038#	12-041#	12-045#	12-047#
	12-049#	12-062	12-062#	12-078#	12-081#	12-083#	12-086#	12-090#	12-092#	12-094#	12-096#	12-P11	12-P11#	12-P22#
	12-P38#	12-P41#	12-P43#	12-P46#	12-P50#	12-P63	12-P63#	12-P75#	12-P81#	12-P84#	12-P91#	12-P94#	12-P96#	12-P99#
	12-Q03#	12-Q05#	12-Q07#	12-Q19	12-Q19#	12-Q32#	12-Q41#	12-Q44#	12-Q52#	12-Q55#	12-Q57#	12-Q60#	12-Q64#	12-Q66#
ASDA	6-0#	16-300*	16-332	16-359*	16-360	16-362*	16-363*	16-364	16-366*	16-398				
ASDC	6-0#	16-299*	16-331	16-367*	16-368	16-370*	16-397							
ASWREG	5-0	5-0												
ATA	4-559#	25-138	25-139	25-141	29-156	29-157	29-192	29-195	33-145	33-146	33-185	33-188	40-58	40-59
	40-60	40-63	40-64	40-67	40-68	40-69	40-70	40-71	40-72	40-73	40-74	40-75	40-76	40-79
	40-80	40-83	40-84	40-87	40-88									
ATESTN	5-0	5-0												
ATMSK	4-596#	34-55												
ATNTBL	9-74	9-96	10-149	11-11	41-3#									
AUNIT	5-0	5-0												
AUSWR	5-0	5-0												
AVECT1	4-780#	5-0	5-0											
AVECT2	5-0	5-0												
BACK	12-99	12-99#	12-110	12-110#	12-126	12-126#	12-129#	12-131#	12-134	12-134#	12-138	12-138#	12-138#	12-151
	12-151#	12-168	12-168#	12-171#	12-173#	12-176	12-176#	12-180	12-180#	12-180#	12-182	12-182#	12-182#	12-184
	12-184#	12-184#	12-197	12-197#	12-213	12-213#	12-216#	12-218#	12-221	12-221#	12-225	12-225#	12-225#	12-227
	12-227#	12-227#	12-229	12-229#	12-229#	12-233	12-233#	12-246	12-246#	12-257	12-257#	12-273	12-273#	12-276#
	12-278#	12-281	12-281#	12-285	12-285#	12-285#	12-298	12-298#	12-315	12-315#	12-318#	12-320#	12-323	12-323#
	12-327	12-327#	12-327#	12-329	12-329#	12-329#	12-331	12-331#	12-331#	12-344	12-344#	12-359	12-359#	12-362#
	12-364#	12-367	12-367#	12-371	12-371#	12-371#	12-373	12-373#	12-373#	12-375	12-375#	12-375#	12-388	12-388#
	12-399	12-399#	12-414	12-414#	12-417#	12-419#	12-422	12-422#	12-426	12-426#	12-426#	12-439	12-439#	12-456
	12-456#	12-459#	12-461#	12-464	12-464#	12-468	12-468#	12-468#	12-470	12-470#	12-470#	12-472	12-472#	12-472#
	12-487	12-487#	12-502	12-502#	12-505#	12-507#	12-510	12-510#	12-514	12-514#	12-514#	12-519	12-519#	12-519#
	12-547	12-547#	12-547#	12-563	12-563#	12-574	12-574#	12-592	12-592#	12-595#	12-597#	12-600	12-600#	12-604
	12-604#	12-604#	12-618	12-618#	12-635	12-635#	12-638#	12-640#	12-643	12-643#	12-647	12-647#	12-647#	12-649
	12-649#	12-649#	12-651	12-651#	12-651#	12-664	12-664#	12-680	12-680#	12-683#	12-685#	12-688	12-688#	12-692
	12-692#	12-692#	12-694	12-694#	12-694#	12-696	12-696#	12-696#	12-698	12-698#	12-711	12-711#	12-722	12-722#
	12-740	12-740#	12-743#	12-745#	12-748	12-748#	12-752	12-752#	12-752#	12-766	12-766#	12-783	12-783#	12-786#
	12-788#	12-791	12-791#	12-795	12-795#	12-795#	12-797	12-797#	12-797#	12-799	12-799#	12-799#	12-812	12-812#
	12-827	12-827#	12-830#	12-832#	12-835	12-835#	12-839	12-839#	12-839#	12-841	12-841#	12-841#	12-843	12-843#
	12-843#	12-856	12-856#	12-867	12-867#	12-885	12-885#	12-888#	12-890#	12-893	12-893#	12-897	12-897#	12-897#
	12-910	12-910#	12-927	12-927#	12-930#	12-932#	12-935	12-935#	12-939	12-939#	12-939#	12-941	12-941#	12-941#
	12-943	12-943#	12-943#	12-958	12-958#	12-973	12-973#	12-976#	12-978#	12-981	12-981#	12-985	12-985#	12-985#
	12-989	12-989#	12-989#	12-:19	12-:19#	12-:19#	12-:35	12-:35#	12-:46	12-:46#	12-:64	12-:64#	12-:67#	12-:69#
	12-:72	12-:72#	12-:76	12-:76#	12-:76#	12-:98	12-:98#	12-:13	12-:13#	12-:16#	12-:18#	12-:21	12-:21#	12-:25
	12-:25#	12-:25#	12-:27	12-:27#	12-:27#	12-:29	12-:29#	12-:29#	12-:42	12-:42#	12-:57	12-:57#	12-:60#	12-:62#
	12-:65	12-:65#	12-:69	12-:69#	12-:69#	12-:71	12-:71#	12-:71#	12-:73	12-:73#	12-:73#	12-:86	12-:86#	12-:98
	12-:98#	12-<16	12-<16#	12-<19#	12-<21#	12-<24	12-<24#	12-<28	12-<28#	12-<28#	12-<41	12-<41#	12-<46	12-<46#
	12-<49#	12-<51#	12-<54	12-<54#	12-<58	12-<58#	12-<76	12-<76#	12-<79#	12-<82	12-<82#	12-<86	12-<86#	12-<86#
	12-<86#	12-<88	12-<88#	12-<88#	12-<90	12-<90#	12-<90#	12-<93	12-<93#	12-<93#	12-<98	12-<98#	12-<98#	12-<98#
	12-=16#	12-=20	12-=20#	12-=20#	12-=37	12-=37#	12-=40#	12-=43	12-=43#	12-=47	12-=47#	12-=47#	12-=49	12-=49#
	12-=49#	12-=51	12-=51#	12-=51#	12-=53	12-=53#	12-=78	12-=78#	12-=87	12-=87#	12->01	12->01#	12->04#	12->06#
	12->09	12->09#	12->13	12->13#	12->13#	12->34	12->34#	12->38	12->38#	12->41#	12->43#	12->46	12->46#	12->50
	12->50#	12->50#	12->65	12->65#	12->68#	12->70#	12->73	12->73#	12->77	12->77#	12->77#	12->79	12->79#	12->79#
	12->81	12->81#	12->81#	12->94	12->94#	12->98	12->98#	12-?01#	12-?03#	12-?06	12-?06#	12-?10	12-?10#	12-?10#
	12-?25	12-?25#	12-?28#	12-?30#	12-?33	12-?33#	12-?37	12-?37#	12-?37#	12-?39	12-?39#	12-?39#	12-?41	12-?41#
	12-?41#	12-?61	12-?61#	12-?70	12-?70#	12-?83	12-?83#	12-?86#	12-?88#	12-?91	12-?91#	12-?95	12-?95#	12-?95#
	12-a16	12-a16#	12-a31	12-a31#	12-a34#	12-a36#	12-a39	12-a39#	12-a43	12-a43#	12-a43#	12-a45	12-a45#	12-a45#
	12-a47	12-a47#	12-a47#	12-a60	12-a60#	12-a75	12-a75#	12-a78#	12-a80#	12-a83	12-a83#	12-a87	12-a87#	12-a87#

12-a89 12-a89# 12-a89# 12-a91 12-a91# 12-a91# 12-A06⁰ 12-A06⁹# 12-A17 12-A17# 12-A41 12-A41# 12-A44# 12-A46#
SEQ 0313

=====

	12-A49	12-A95#	12-A53	12-A53#	12-A53#	12-A66	12-A66#	12-A83	12-A83#	12-A86#	12-A88#	12-A91	12-A91#	12-A95	
	12-A95#	12-A95#	12-B12	12-B12#	12-B12#	12-B28	12-B28#	12-B28#	12-B28#	12-B40	12-B40#	12-B40#	12-B49	12-B49#	12-B49#
	12-B61	12-B61#	12-B78	12-B78#	12-B81#	12-B83#	12-B86	12-B86#	12-B86#	12-B90	12-B90#	12-B90#	12-C05	12-C05#	12-C05#
	12-C21	12-C21#	12-C21#	12-C33	12-C33#	12-C33#	12-C42	12-C42#	12-C42#	12-C65	12-C65#	12-C76	12-C76#	12-C92	
	12-C92#	12-D03	12-D03#	12-D29	12-D29#	12-D32#	12-D34#	12-D37	12-D37#	12-D41	12-D41#	12-D41#	12-D54	12-D54#	12-D54#
	12-D71	12-D71#	12-D74#	12-D76#	12-D79	12-D79#	12-D83	12-D83#	12-D83#	12-E02	12-E02#	12-E02#	12-E04	12-E04#	12-E04#
	12-E04#	12-E17	12-E17#	12-E33	12-E33#	12-E36#	12-E38#	12-E41	12-E41#	12-E45	12-E45#	12-E45#	12-E64	12-E64#	12-E64#
	12-E64#	12-E66	12-E66#	12-E66#	12-E68	12-E68#	12-E90	12-E90#	12-E90#	12-F01	12-F01#	12-F21	12-F28	12-F28#	12-F28#
	12-F49	12-F49#	12-F52#	12-F54#	12-F57	12-F57#	12-F61	12-F61#	12-F61#	12-F66	12-F66#	12-F66#	12-F74	12-F74#	12-F74#
	12-F74#	12-F87	12-F87#	12-G09	12-G09#	12-G12#	12-G14#	12-G17	12-G17#	12-G21	12-G21#	12-G21#	12-G26	12-G26#	12-G26#
	12-G26#	12-G34	12-G34#	12-G34#	12-G54	12-G54#	12-G73	12-G73#	12-G73#	12-G76#	12-G79	12-G79#	12-G89#	12-G92	12-G92#
	12-G94	12-G94#	12-G94#	12-G96	12-G96#	12-G96#	12-H09	12-H09#	12-H09#	12-H28	12-H28#	12-H31#	12-H34	12-H34#	12-H34#
	12-H48	12-H48#	12-H50	12-H50#	12-H50#	12-H52	12-H52#	12-H52#	12-H52#	12-H64	12-H64#	12-H75	12-H75#	12-H91	12-H91#
	12-H94#	12-H96#	12-H99	12-H99#	12-I03	12-I03#	12-I03#	12-I17	12-I17#	12-I35	12-I35#	12-I38#	12-I40#	12-I43	12-I43#
	12-I43#	12-I47	12-I47#	12-I47#	12-I49	12-I49#	12-I49#	12-I53	12-I53#	12-I53#	12-I66	12-I66#	12-I84	12-I84#	12-I84#
	12-I87#	12-I89#	12-I92	12-I92#	12-I96	12-I96#	12-I96#	12-I98	12-I98#	12-I98#	12-J02	12-J02#	12-J02#	12-J04	12-J04#
	12-J04#	12-J18	12-J18#	12-J30	12-J30#	12-J46	12-J46#	12-J49#	12-J51#	12-J54	12-J54#	12-J58	12-J58#	12-J58#	12-J58#
	12-J72	12-J72#	12-J93	12-J93#	12-J96#	12-J98#	12-K01	12-K01#	12-K05	12-K05#	12-K05#	12-K07	12-K07#	12-K07#	12-K07#
	12-K11	12-K11#	12-K11#	12-K24	12-K24#	12-K40	12-K40#	12-K43#	12-K45#	12-K48	12-K48#	12-K52	12-K52#	12-K52#	12-K52#
	12-K54	12-K54#	12-K54#	12-K58	12-K58#	12-K58#	12-K71	12-K71#	12-K82	12-K82#	12-K98	12-K98#	12-L01#	12-L03#	12-L03#
	12-L06	12-L06#	12-L10	12-L10#	12-L10#	12-L23	12-L23#	12-L40	12-L40#	12-L43#	12-L45#	12-L48	12-L48#	12-L52	12-L52#
	12-L52#	12-L52#	12-L54	12-L54#	12-L54#	12-L56	12-L56#	12-L56#	12-L69	12-L69#	12-L85	12-L85#	12-L88#	12-L90#	12-L90#
	12-L93	12-L93#	12-L97	12-L97#	12-L97#	12-L99	12-L99#	12-L99#	12-M01	12-M01#	12-M01#	12-M03	12-M03#	12-M16	12-M16#
	12-M16#	12-M27	12-M27#	12-M43	12-M43#	12-M46#	12-M48#	12-M51	12-M51#	12-M55	12-M55#	12-M55#	12-M68	12-M68#	12-M68#
	12-M85	12-M85#	12-M88#	12-M90#	12-M93	12-M93#	12-M97	12-M97#	12-M97#	12-M99	12-M99#	12-M99#	12-M99#	12-N01	12-N01#
	12-N01#	12-N14	12-N14#	12-N30	12-N30#	12-N33#	12-N35#	12-N38	12-N38#	12-N42	12-N42#	12-N42#	12-N44	12-N44#	12-N44#
	12-N44#	12-N46	12-N46#	12-N46#	12-N48	12-N48#	12-N61	12-N61#	12-N73	12-N73#	12-N91	12-N91#	12-N94#	12-N96#	12-N96#
	12-N99	12-N99#	12-003	12-003#	12-003#	12-016	12-016#	12-033	12-033#	12-036#	12-038#	12-041	12-041#	12-045	12-045#
	12-045#	12-045#	12-047	12-047#	12-047#	12-049	12-049#	12-049#	12-062	12-062#	12-078	12-078#	12-081#	12-083#	12-083#
	12-086	12-086#	12-090	12-090#	12-090#	12-092	12-092#	12-092#	12-094	12-094#	12-094#	12-096	12-096#	12-P11	12-P11#
	12-P11#	12-P22	12-P22#	12-P38	12-P38#	12-P41#	12-P43#	12-P46	12-P46#	12-P50	12-P50#	12-P50#	12-P63	12-P63#	12-P63#
	12-P75	12-P75#	12-P81	12-P81#	12-P84#	12-P91	12-P91#	12-P94#	12-P96#	12-P99	12-P99#	12-Q03	12-Q03#	12-Q03#	12-Q03#
	12-Q05	12-Q05#	12-Q05#	12-Q07	12-Q07#	12-Q07#	12-Q19	12-Q19#	12-Q32	12-Q32#	12-Q41	12-Q41#	12-Q44#	12-Q52	12-Q52#
	12-Q52#	12-Q55#	12-Q57#	12-Q60	12-Q60#	12-Q64	12-Q64#	12-Q64#	12-Q66	12-Q66#	12-Q66#	12-Q68	12-Q68#	12-Q68#	12-Q68#
BADSC	12-110	12-257	12-399	12-574	12-722	12-867	12-:46	12-:98	12-=:78	12-?61	12-A17	12-D03	12-F21	12-H75	12-H75#
	12-J30	12-K82	12-M27	12-N73	12-P22	16-33#									
BADTMD	9-3#	9-21													
BAI	4-746#	26-28													
BB00	4-677#														
BB01	4-677#														
BB02	4-677#														
BB03	4-677#														
BB04	4-677#														
BB05	4-677#														
BB06	4-677#														
BB07	4-677#														
BB08	4-677#														
BB09	4-677#														
BIT0	4-491#	12-74	12-76	12-79	12-81	23-18									
BIT00	4-491	4-491#	4-503	4-551	4-569	4-588	4-625	4-643	4-677	4-749	4-767				
BIT01	4-491	4-491#	4-502	4-550	4-587	4-624	4-642	4-677	4-749	4-766					
BIT02	4-491	4-491#	4-501	4-549	4-586	4-623	4-641	4-677	4-749	4-765					
BIT03	4-491	4-491#	4-500	4-548	4-585	4-622	4-640	4-677	4-688	4-746	4-764				
BIT04	4-491	4-491#	4-499	4-547	4-584	4-639	4-677	4-745							
BIT05	4-491	4-491#	4-498	4-583	4-621	4-638	4-677	4-744							
BIT06	4-491	4-491#	4-568	4-582	4-604	4-620	4-637	4-677	4-730	4-743	4-763				

BIT08	4-491	4-491#	4-544	4-566	4-580	4-602	4-618	4-635	4-677	4-728	4-741	38-6		
BIT09	4-491	4-491#	4-543	4-565	4-579	4-601	4-617	4-634	4-677	4-727	4-740	38-6	38-7	
BIT1	4-491#	12-79	12-81	16-305	17-26	26-48								
BIT10	4-491#	4-542	4-564	4-578	4-600	4-616	4-633	4-659	4-674	4-686	4-726	4-739	4-762	38-7
BIT11	4-491#	4-497	4-541	4-563	4-577	4-615	4-632	4-650	4-658	4-673	4-685	4-738	4-761	15-165
	38-6													
BIT12	4-491#	4-540	4-562	4-576	4-614	4-631	4-657	4-672	4-684	4-737	4-760	15-111		
BIT13	4-491#	4-561	4-575	4-613	4-630	4-649	4-671	4-683	4-725	4-736	4-759	38-7		
BIT14	4-491#	4-560	4-574	4-612	4-629	4-648	4-670	4-682	4-693	4-724	4-735	4-758	15-73	38-6
BIT15	4-491#	4-559	4-573	4-611	4-628	4-647	4-669	4-681	4-692	4-723	4-734	4-757	18-33	18-90
BIT2	4-491#	23-18												
BIT3	4-491#	15-196	35-320	35-407										
BIT4	4-491#	15-141	35-426											
BIT5	4-491#													
BIT6	4-491#	15-87												
BIT7	4-491#	9-68	23-24											
BIT8	4-491#													
BIT9	4-491#													
BOTADR	14-53*	14-71*	14-74	14-89	14-134#									
BOTFLG	14-39*	14-81*	14-84	14-87*	14-135#									
BPTVEC	4-491#													
BSE	4-681#	12-B37	12-B43	12-C30	12-C36	12-D93	12-D95	12-E55	12-E57	16-221	25-431			
BUFFER	16-49*	16-285	51-2#	51-16										
BUFONE	12-105	12-156	12-233	12-252	12-303	12-394	12-445	12-569	12-623	12-698	12-717	12-771	12-862	12-916
	12-:41	12-:84	12-:93	12-<65	12-=53	12-=74	12->21	12-?57	12-a04	12-A12	12-A24	12-A26*	12-A28*	12-A71
	12-C61	12-C98	12-D10	12-D12*	12-D14*	12-D61	12-E68	12-E86	12-F16	12-G47	12-H70	12-I24	12-J04	12-J25
BUF TWO	12-J79	12-K77	12-L29	12-M03	12-M22	12-M74	12-N48	12-N68	12-O21	12-O96	12-P17	12-P58	51-3#	
	12-202	12-233	12-484*	12-525	12-532	12-534	12-549*	12-669	12-698	12-955*	12-995	12-:04	12-:06	12-:21*
	12-=27	12-=53	12-B66	12-E23	12-E68	12-F97	12-I72	12-J04	12-L75	12-M03	12-N19	12-N48	12-067	12-096
	51-4#													
CC	4-673#													
CH	4-674#													
CHRCNT	14-40*	14-58*	14-64*	14-65	14-68*	14-72	14-77*	14-88*	14-136#					
CKSWR	38-6	38-7	38-7	38-10#										
CLKADR	6-0#	22-8*	22-10	22-14*	22-16	23-13								
CLKVCT	6-0#	22-9*	22-15*											
CLR	4-744#	9-82	11-49	30-16										
CLRSTS	15-98	31-22#												
CMNSTA	9-114	10-18	11-2#											
CMPBUF	12-233	12-698	12-=53	12-E68	12-J04	12-M03	12-N48	12-096	18-17#					
CMPERR	27-21#													
CNSL00	10-11	39-16#												
CNSL01	10-55	39-17#												
CNSL02	10-63	39-18#												
CNSL03	10-67	39-19#												
CNSL04	10-79	39-20#												
CNSL05	10-83	39-21#												
CNSL06	10-100	39-22#												
CNSL07	10-113	39-23#												
CNSL08	10-22	10-35	10-50	10-144	39-26#									
CNSL09	39-27#													
CNTCLR	12-36	12-65	15-76	30-12#										
COMMA	10-138	39-8#												
CONT	4-637#													
CR	4-491#	10-125	10-136	14-56	38-5	38-5	47-79							
CRLF	4-491#	9-6	9-25	9-25	9-39	9-50	9-56	9-57	11-24	37-18	37-18	38-5	38-5	39-3

39-4

39-6

39-9

39-10

39-11

39-12

39-12^{H 9}

39-13

39-15

39-16

39-17

39-18

39-20

39-22
SEQ 0317

	39-23	39-24	39-25	39-26	39-27	39-28	48-7	48-11	48-12	48-15	48-16	48-18	51-19	51-20
	51-21	51-22	51-23	51-24	51-25	51-26	51-27	51-28	51-29	51-30	51-31	51-32	51-33	51-34
	51-35	51-36	51-37	51-38	51-39	51-40	51-41	51-42	51-43	51-44	51-45	51-46	51-47	51-48
	51-49	51-50	51-51	51-52	51-53	51-54	51-55	51-56	51-57	51-58	51-59	51-60	51-61	51-62
	51-63	51-64	51-65											
CTLFG	6-0#	11-39*	37-13	37-22*										
CYLSK	4-664#													
DBCK	4-611#													
DBEN	4-612#													
DBL	4-762#													
DCK	4-573#	4-590	16-225	18-23	25-469	25-489	35-413	35-416						
DDISP	4-491#	5-0	9-19											
DEBL	4-613#													
DEVSEL	15-62	28-12#												
DISPLA	5-0#	9-19*	9-19*	38-6*	38-7*									
DISPRE	4-784#	9-19												
DLT	4-734#	25-313	35-492	35-495										
DMD	4-625#	4-643#	12-F32	12-F91										
DPE	4-688#	24-137	25-255	29-32	33-31	35-43	35-466	35-469						
DPEHI	4-758#													
DPELO	4-759#													
DPR	4-566#	29-195	31-125	31-126	34-30									
DRQ	4-650#													
DRVCLR	4-511#	16-87	34-18											
DRVSTS	16-105	34-9#												
DRY	4-567#	24-104	25-38	25-41	25-42	29-195	31-125	31-126	34-30					
DSWR	4-491#	5-0	9-19											
DTASTS	12-138	12-182	12-227	12-285	12-329	12-373	12-426	12-470	12-519	12-604	12-649	12-694	12-752	12-797
	12-841	12-897	12-941	12-989	12-:76	12-:27	12-:71	12-<28	12-<88	12-=49	12->13	12->79	12-?39	12-?95
	12-a45	12-a89	12-A53	12-B12	12-B28	12-B40	12-C05	12-C21	12-C33	12-D41	12-E02	12-E64	12-F66	12-G26
	12-103	12-I49	12-198	12-J58	12-K07	12-K54	12-L10	12-L54	12-L99	12-M55	12-M99	12-N44	12-003	12-047
	12-092	12-P50	12-Q05	12-Q66	16-200	35-17#								
DTE	4-576#	4-590	35-279	35-297	35-300									
DTO	4-614#													
DULPRT	4-653#	12-71	12-76	12-81	12-85									
DVA	4-497#	9-89	12-51	20-43	21-37	24-65	24-68	28-37	31-31	34-17	34-18			
DVC	4-687#	29-127	31-115	33-92	33-130	33-133	33-266	35-232	35-236	35-239	35-284	36-89	36-94	36-97
EARLY	12-K84	42-107#												
EBL	4-630#													
ECH	4-582#	4-590	16-225	18-25	25-452	25-467	25-485	25-491	35-423	40-77	40-78	40-85	40-86	
ECI	4-658#	18-21	25-487	35-420										
ECRC	4-634#													
ED1	45-1	49-1#												
ED110	45-4	49-2#												
ED111	45-5	49-3#												
ED114	45-7	49-5#												
ED223	45-8	49-6#												
ED336	45-10	49-8#												
ED337	45-11	45-12	49-10#											
ED353	45-14	49-11#												
EDT1	7-4	7-7	7-10	7-13	7-16	7-19	7-25	7-28	7-31	7-34	7-37	7-40	7-43	7-46
	7-49	7-52	7-55	7-58	7-61	7-64	7-67	7-70	7-73	7-76	7-79	7-82	7-85	7-88
	7-91	7-94	7-97	7-100	7-103	7-106	7-109	7-112	7-115	7-118	7-121	7-124	7-127	7-131
	7-134	7-137	7-140	7-143	7-147	7-151	7-155	7-161	7-164	7-167	7-170	7-173	7-176	7-179
	7-183	7-186	7-189	7-192	7-195	7-198	7-201	7-204	7-207	7-210	7-213	7-216	7-219	7-237
	7-240	7-243	7-246	7-249	7-252	7-255	7-258	7-261	7-264	7-267	7-270	7-273	7-276	7-279

7-282 7-285 7-288 7-291 7-294 7-297 7-300^J 9 7-303 7-306 7-309 7-312 7-315 7-318 7-321
SEQ 0319

EH353 44-15 48-18#

L 9

SEQ 0321



EMS205 43-55 43-142 43-179 43-233 47-136#

C 10

SEQ 0325

EMS57 43-190 43-191 43-192 43-193 43-194 43-195 43-196^{E 10} 43-197 43-198 43-199 43-200 43-201 43-202 43-203
SEQ 0327

F
F
G
G
G
G

EMT136	7-288	43-96#
EMT137	7-291	43-97#
EMT14	7-37	43-14#
EMT140	7-294	43-98#
EMT141	7-297	43-99#
EMT142	7-300	43-100#
EMT143	7-303	43-101#
EMT144	7-306	43-102#
EMT145	7-309	43-103#
EMT146	7-312	43-104#
EMT147	7-315	43-105#
EMT15	7-40	43-15#
EMT150	7-318	43-106#
EMT151	7-321	43-107#
EMT152	7-324	43-108#
EMT153	7-327	43-109#
EMT154	7-330	43-110#
EMT155	7-333	43-111#
EMT156	7-336	43-112#
EMT157	7-339	43-113#
EMT16	7-43	43-16#
EMT160	7-342	43-114#
EMT161	7-345	43-115#
EMT162	7-348	43-116#
EMT163	43-117#	
EMT164	7-355	43-118#
EMT165	7-358	43-119#
EMT166	7-361	43-120#
EMT167	7-364	43-121#
EMT17	7-46	43-17#
EMT170	43-122#	
EMT171	7-370	43-123#
EMT172	7-373	43-124#
EMT173	7-376	43-125#
EMT174	7-379	43-126#
EMT175	7-382	43-127#
EMT176	7-385	43-128#
EMT177	7-388	43-129#
EMT2	7-7	43-4#
EMT20	7-49	43-18#
EMT200	7-391	43-130#
EMT201	7-394	43-131#
EMT202	7-397	43-132#
EMT203	7-400	43-133#
EMT204	7-403	43-134#
EMT205	7-406	43-135#
EMT206	7-409	43-136#
EMT207	7-412	43-137#
EMT21	7-52	43-19#
EMT210	7-415	43-138#
EMT211	7-418	43-139#
EMT212	7-421	43-140#
EMT213	7-424	43-141#
EMT214	7-427	43-142#
EMT215	7-430	43-143#
EMT216	7-433	43-144#

EMT22	7-55	43-20#
EMT220	7-439	43-146#
EMT221	7-442	43-147#
EMT222	7-445	43-148#
EMT223	7-448	43-149#
EMT224	43-150#	
EMT225	43-151#	
EMT226	43-152#	
EMT227	43-153#	
EMT23	7-58	43-21#
EMT230	43-154#	
EMT231	43-155#	
EMT232	43-156#	
EMT233	43-157#	
EMT234	43-158#	
EMT235	43-159#	
EMT236	43-160#	
EMT237	43-161#	
EMT24	7-61	43-22#
EMT240	43-162#	
EMT241	43-163#	
EMT242	43-164#	
EMT243	43-165#	
EMT244	43-166#	
EMT245	43-167#	
EMT246	7-505	43-168#
EMT247	7-508	43-169#
EMT25	7-64	43-23#
EMT250	7-511	43-170#
EMT251	7-514	43-171#
EMT252	7-517	43-172#
EMT253	7-520	43-173#
EMT254	7-523	43-174#
EMT255	7-526	43-175#
EMT256	7-529	43-176#
EMT257	7-532	43-177#
EMT26	7-67	43-24#
EMT260	7-535	43-178#
EMT261	7-538	43-179#
EMT262	7-541	43-180#
EMT263	7-544	43-181#
EMT264	7-547	43-182#
EMT265	7-550	43-183#
EMT266	7-553	43-184#
EMT267	7-557	43-185#
EMT27	7-70	43-25#
EMT270	7-560	43-186#
EMT271	7-564	43-187#
EMT272	7-567	43-188#
EMT273	7-570	43-189#
EMT274	7-574	43-190#
EMT275	7-578	43-191#
EMT276	7-582	43-192#
EMT277	7-587	43-193#
EMT3	7-10	43-5#
EMT30	7-73	43-26#

EMT300 7-591 43-194#

K 10

SEQ 0333

EMT301	7-595	43-195#	
EMT302	7-598	43-196#	
EMT303	7-601	43-197#	
EMT304	7-604	43-198#	
EMT305	7-607	43-199#	
EMT306	7-610	43-200#	
EMT307	7-613	43-201#	
EMT31	7-76	43-27#	
EMT310	7-616	43-202#	
EMT311	7-619	43-203#	
EMT312	7-622	43-204#	
EMT313	7-625	43-205#	
EMT314	7-628	43-206#	
EMT315	7-631	43-207#	
EMT316	7-634	43-208#	
EMT317	7-637	43-209#	
EMT32	7-79	43-28#	
EMT320	7-640	43-210#	
EMT321	7-643	43-211#	
EMT322	7-646	43-212#	
EMT323	7-649	43-213#	
EMT324	7-652	43-214#	
EMT325	7-655	43-215#	
EMT326	7-658	43-216#	
EMT327	7-661	43-217#	
EMT33	7-82	43-29#	
EMT330	7-664	43-218#	
EMT331	7-667	43-219#	
EMT332	7-670	43-220#	
EMT333	7-673	43-221#	
EMT334	7-676	43-222#	
EMT335	7-679	43-223#	
EMT336	7-352	7-682	43-224#
EMT337	7-685	43-225#	
EMT34	7-85	43-30#	
EMT340	7-688	43-226#	
EMT341	7-691	43-227#	
EMT342	7-694	43-228#	
EMT343	7-697	43-229#	
EMT344	7-700	43-230#	
EMT345	7-703	43-231#	
EMT346	7-706	43-232#	
EMT347	7-709	43-233#	
EMT35	7-88	43-31#	
EMT350	7-712	43-234#	
EMT351	7-715	43-235#	
EMT352	7-718	43-236#	
EMT353	7-721	43-237#	
EMT354	7-724	43-238#	
EMT36	7-91	43-32#	
EMT37	7-94	43-33#	
EMT4	7-13	43-6#	
EMT40	7-97	43-34#	
EMT41	7-100	43-35#	
EMT42	7-103	43-36#	
EMT43	7-106	43-37#	

EMT44

7-109

43-38#

M 10

SEQ 0335

EMT45	7-112	43-39#														
EMT46	7-115	43-40#														
EMT47	7-118	43-41#														
EMT5	7-16	43-7#														
EMT50	7-121	43-42#														
EMT51	7-124	43-43#														
EMT52	7-127	43-44#														
EMT53	7-131	43-45#														
EMT54	7-134	43-46#														
EMT55	7-137	43-47#														
EMT56	7-140	43-48#														
EMT57	7-143	43-49#														
EMT6	7-19	43-8#														
EMT60	7-147	43-50#														
EMT61	7-151	43-51#														
EMT62	7-155	43-52#														
EMT63	43-53#															
EMT64	7-161	43-54#														
EMT65	7-164	43-55#														
EMT66	7-167	43-56#														
EMT67	7-170	43-57#														
EMT7	7-22	43-9#														
EMT70	7-173	43-58#														
EMT71	7-176	43-59#														
EMT72	7-179	43-60#														
EMT73	7-183	43-61#														
EMT74	7-186	43-62#														
EMT75	7-189	43-63#														
EMT76	7-192	43-64#														
EMT77	7-195	43-65#														
EMTVEC	4-491#	9-19*	9-19*													
ENRGDT	42-144#															
EQUALS	39-5#															
ERR	4-560#	25-84	25-86	25-102	25-136											
ERRNMB	14-28*	14-29*	14-32	14-36	14-41	14-133#										
ERROR	4-491#															
ERRTYP	14-15#	38-7														
ERRVEC	4-491#	9-19	9-19*	9-19*	9-21*	9-22*	12-5	12-6	12-7*	12-8*	12-19*	12-20*	12-24*	12-25*		
	12-38	12-39	12-40*	12-41*	12-45*	12-46*	12-57*	12-58*	20-35	20-35	20-39*	20-40*	20-70*	20-70*		
	21-29	21-29	21-33*	21-34*	21-59*	21-59*	22-4	22-5	22-6*	22-7*	22-13*	22-20*	22-21*	23-9		
	23-9	23-10*	23-11*	23-35*	23-35*	28-20	28-20	28-21*	28-22*	28-51*	28-51*	30-12	30-12	30-13*		
	30-14*	30-25*	30-25*	38-6	38-6*	38-6*	38-6*									
ERTY00	14-22	14-138#														
ERTY01	14-26	14-139#														
ERTY02	14-31	14-140#														
ERTY03	14-33	14-141#														
ERTY04	14-125	14-142#														
ESRC	4-632#															
F0	4-502#	25-29														
F1	4-501#	25-29														
F2	4-500#	25-29	26-113													
F3	4-499#	25-29														
F4	4-498#	25-29														
FER	4-584#	4-590	12-B25	12-B30	12-C18	12-C23	12-D86	12-D89	12-E48	12-E51	16-225	25-416	35-348	35-365		
	35-368	35-391	35-394													
FIND	12-99	12-99#	12-99#	12-110	12-110#	12-110#	12-126	12-126#	12-126#	12-129	12-129#	12-131	12-131#	12-134		

12-173#	12-176	12-176#	12-176#	12-180	12-180#	12-180#	12-182	12-182#	12-182#	12-184	12-184#	12-184#	12-197
12-197#	12-197#	12-213	12-213#	12-213#	12-216	12-216#	12-218	12-218#	12-221	12-221#	12-221#	12-225	12-225#
12-225#	12-227	12-227#	12-227#	12-229	12-229#	12-229#	12-233	12-233#	12-233#	12-246	12-246#	12-246#	12-257
12-257#	12-257#	12-273	12-273#	12-273#	12-276	12-276#	12-278	12-278#	12-281	12-281#	12-281#	12-285	12-285#
12-285#	12-298	12-298#	12-298#	12-315	12-315#	12-315#	12-318	12-318#	12-320	12-320#	12-323	12-323#	12-323#
12-327	12-327#	12-327#	12-329	12-329#	12-329#	12-331	12-331#	12-331#	12-344	12-344#	12-344#	12-359	12-359#
12-359#	12-362	12-362#	12-364	12-364#	12-367	12-367#	12-367#	12-371	12-371#	12-371#	12-373	12-373#	12-373#
12-375	12-375#	12-375#	12-388	12-388#	12-388#	12-399	12-399#	12-399#	12-414	12-414#	12-414#	12-417	12-417#
12-419	12-419#	12-422	12-422#	12-422#	12-426	12-426#	12-426#	12-439	12-439#	12-439#	12-456	12-456#	12-456#
12-459	12-459#	12-461	12-461#	12-464	12-464#	12-464#	12-468	12-468#	12-468#	12-470	12-470#	12-470#	12-472
12-472#	12-472#	12-487	12-487#	12-487#	12-502	12-502#	12-502#	12-505	12-505#	12-507	12-507#	12-510	12-510#
12-510#	12-514	12-514#	12-514#	12-519	12-519#	12-519#	12-547	12-547#	12-547#	12-563	12-563#	12-563#	12-574
12-574#	12-574#	12-592	12-592#	12-592#	12-595	12-595#	12-597	12-597#	12-600	12-600#	12-600#	12-604	12-604#
12-604#	12-618	12-618#	12-618#	12-635	12-635#	12-635#	12-638	12-638#	12-640	12-640#	12-643	12-643#	12-643#
12-647	12-647#	12-647#	12-649	12-649#	12-649#	12-651	12-651#	12-651#	12-664	12-664#	12-664#	12-680	12-680#
12-680#	12-683	12-683#	12-685	12-685#	12-688	12-688#	12-688#	12-692	12-692#	12-692#	12-694	12-694#	12-694#
12-696	12-696#	12-696#	12-698	12-698#	12-698#	12-711	12-711#	12-711#	12-722	12-722#	12-722#	12-740	12-740#
12-740#	12-743	12-743#	12-745	12-745#	12-748	12-748#	12-748#	12-752	12-752#	12-752#	12-766	12-766#	12-766#
12-783	12-783#	12-783#	12-786	12-786#	12-788	12-788#	12-791	12-791#	12-791#	12-795	12-795#	12-795#	12-797
12-797#	12-797#	12-799	12-799#	12-799#	12-812	12-812#	12-812#	12-827	12-827#	12-827#	12-830	12-830#	12-832
12-832#	12-835	12-835#	12-835#	12-839	12-839#	12-839#	12-841	12-841#	12-841#	12-843	12-843#	12-843#	12-856
12-856#	12-856#	12-867	12-867#	12-867#	12-885	12-885#	12-885#	12-888	12-888#	12-890	12-890#	12-893	12-893#
12-893#	12-897	12-897#	12-897#	12-910	12-910#	12-910#	12-927	12-927#	12-927#	12-930	12-930#	12-932	12-932#
12-935	12-935#	12-935#	12-939	12-939#	12-939#	12-941	12-941#	12-941#	12-943	12-943#	12-943#	12-958	12-958#
12-958#	12-973	12-973#	12-973#	12-976	12-976#	12-978	12-978#	12-981	12-981#	12-981#	12-985	12-985#	12-985#
12-989	12-989#	12-989#	12-:19	12-:19#	12-:19#	12-:35	12-:35#	12-:35#	12-:46	12-:46#	12-:46#	12-:64	12-:64#
12-:64#	12-:67	12-:67#	12-:69	12-:69#	12-:69#	12-:72	12-:72#	12-:72#	12-:76	12-:76#	12-:76#	12-:98	12-:98#
12-:13	12-:13#	12-:13#	12-:16	12-:16#	12-:16#	12-:18	12-:18#	12-:21	12-:21#	12-:21#	12-:25	12-:25#	12-:27
12-:27#	12-:27#	12-:29	12-:29#	12-:29#	12-:42	12-:42#	12-:42#	12-:57	12-:57#	12-:57#	12-:60	12-:60#	12-:62
12-:62#	12-:65	12-:65#	12-:65#	12-:69	12-:69#	12-:69#	12-:71	12-:71#	12-:71#	12-:73	12-:73#	12-:73#	12-:86
12-:86#	12-:86#	12-:98	12-:98#	12-:98#	12-<16	12-<16#	12-<16#	12-<19	12-<19#	12-<21	12-<21#	12-<24	12-<24#
12-<24#	12-<28	12-<28#	12-<28#	12-<41	12-<41#	12-<41#	12-<46	12-<46#	12-<46#	12-<49	12-<49#	12-<51	12-<51#
12-<54	12-<54#	12-<54#	12-<58	12-<58#	12-<58#	12-<76	12-<76#	12-<76#	12-<79	12-<79#	12-<82	12-<82#	12-<82#
12-<86	12-<86#	12-<86#	12-<88	12-<88#	12-<88#	12-<90	12-<90#	12-<90#	12=-03	12=-03#	12=-03#	12=-08	12=-08#
12=-08#	12=-11	12=-11#	12=-13	12=-13#	12=-16	12=-16#	12=-16#	12=-16#	12=-20	12=-20#	12=-20#	12=-37	12=-37#
12=-40	12=-40#	12=-43	12=-43#	12=-43#	12=-47	12=-47#	12=-47#	12=-47#	12=-49	12=-49#	12=-49#	12=-51	12=-51#
12=-53	12=-53#	12=-53#	12=-78	12=-78#	12=-78#	12=-87	12=-87#	12=-87#	12->01	12->01#	12->01#	12->04	12->04#
12->06	12->06#	12->09	12->09#	12->09#	12->13	12->13#	12->13#	12->13#	12->34	12->34#	12->34#	12->38	12->38#
12->41	12->41#	12->43	12->43#	12->46	12->46#	12->46#	12->50	12->50#	12->50#	12->65	12->65#	12->65#	12->68
12->68#	12->70	12->70#	12->73	12->73#	12->73#	12->77	12->77#	12->77#	12->79	12->79#	12->79#	12->81	12->81#
12->81#	12->94	12->94#	12->94#	12->98	12->98#	12->98#	12-?01	12-?01#	12-?03	12-?03#	12-?06	12-?06#	12-?06#
12-?10	12-?10#	12-?10#	12-?25	12-?25#	12-?25#	12-?28	12-?28#	12-?30	12-?30#	12-?33	12-?33#	12-?33#	12-?37
12-?37#	12-?37#	12-?39	12-?39#	12-?39#	12-?41	12-?41#	12-?41#	12-?61	12-?61#	12-?61#	12-?70	12-?70#	12-?70#
12-?83	12-?83#	12-?83#	12-?86	12-?86#	12-?88	12-?88#	12-?91	12-?91#	12-?91#	12-?95	12-?95#	12-?95#	12-@16
12-@16#	12-@16#	12-@31	12-@31#	12-@31#	12-@34	12-@34#	12-@36	12-@36#	12-@39	12-@39#	12-@39#	12-@43	12-@43#
12-@43#	12-@45	12-@45#	12-@45#	12-@47	12-@47#	12-@47#	12-@60	12-@60#	12-@60#	12-@75	12-@75#	12-@75#	12-@78
12-@78#	12-@80	12-@80#	12-@83	12-@83#	12-@83#	12-@87	12-@87#	12-@87#	12-@89	12-@89#	12-@89#	12-@91	12-@91#
12-@91#	12-A06	12-A06#	12-A06#	12-A17	12-A17#	12-A17#	12-A41	12-A41#	12-A41#	12-A44	12-A44#	12-A46	12-A46#
12-A49	12-A49#	12-A49#	12-A53	12-A53#	12-A53#	12-A66	12-A66#	12-A66#	12-A83	12-A83#	12-A83#	12-A86	12-A86#
12-A88	12-A88#	12-A91	12-A91#	12-A91#	12-A95	12-A95#	12-A95#	12-B12	12-B12#	12-B12#	12-B28	12-B28#	12-B28#
12-B40	12-B40#	12-B40#	12-B49	12-B49#	12-B49#	12-B61	12-B61#	12-B61#	12-B78	12-B78#	12-B78#	12-B81	12-B81#
12-B83	12-B83#	12-B86	12-B86#	12-B86#	12-B90	12-B90#	12-B90#	12-C05	12-C05#	12-C05#	12-C21	12-C21#	12-C21#
12-C33	12-C33#	12-C33#	12-C42	12-C42#	12-C42#	12-C65	12-C65#	12-C65#	12-C76	12-C76#	12-C76#	12-C92	12-C92#
12-C92#	12-D03	12-D03#	12-D03#	12-D29	12-D29#	12-D29#	12-D32	12-D32#	12-D34	12-D34#	12-D37	12-D37#	12-D37#
12-D41	12-D41#	12-D41#	12-D54	12-D54#	12-D54#	12-D71	12-D71#	12-D71#	12-D74	12-D74#	12-D76	12-D76#	12-D79
12-D79#	12-D79#	12-D83	12-D83#	12-D83#	12-E02	12-E02#	12-E02#	12-E04	12-E04#	12-E04#	12-E17	12-E17#	12-E17#

12-E64#	12-E64#	12-E66	12-E66#	12-E66#	12-E68	12-E68#	12-E68#	12-E90	12-E90#	12-E90#	12-F01	12-F01#	12-F01#	
12-F21	12-F21#	12-F21#	12-F28	12-F28#	12-F28#	12-F49	12-F49#	12-F49#	12-F52	12-F52#	12-F54	12-F54#	12-F57	
12-F57#	12-F57#	12-F61	12-F61#	12-F61#	12-F66	12-F66#	12-F66#	12-F74	12-F74#	12-F74#	12-F87	12-F87#	12-F87#	
12-G09	12-G09#	12-G09#	12-G12	12-G12#	12-G12#	12-G14	12-G14#	12-G17	12-G17#	12-G17#	12-G21	12-G21#	12-G21#	
12-G26#	12-G26#	12-G34	12-G34#	12-G34#	12-G54	12-G54#	12-G54#	12-G73	12-G73#	12-G73#	12-G76	12-G76#	12-G79	
12-G79#	12-G79#	12-G89	12-G89#	12-G89#	12-G92	12-G92#	12-G92#	12-G94	12-G94#	12-G94#	12-G96	12-G96#	12-H09	
12-H09#	12-H09#	12-H28	12-H28#	12-H28#	12-H31	12-H31#	12-H31#	12-H34	12-H34#	12-H34#	12-H45	12-H45#	12-H48#	
12-H48#	12-H50	12-H50#	12-H50#	12-H52	12-H52#	12-H52#	12-H64	12-H64#	12-H64#	12-H75	12-H75#	12-H75#	12-H91	
12-H91#	12-H91#	12-H94	12-H94#	12-H96	12-H96#	12-H99	12-H99#	12-H99#	12-I03	12-I03#	12-I03#	12-I17	12-I17#	
12-I17#	12-I35	12-I35#	12-I35#	12-I38	12-I38#	12-I40	12-I40#	12-I43	12-I43#	12-I43#	12-I47	12-I47#	12-I47#	
12-I49	12-I49#	12-I49#	12-I53	12-I53#	12-I53#	12-I66	12-I66#	12-I66#	12-I84	12-I84#	12-I84#	12-I87	12-I87#	
12-I89	12-I89#	12-I92	12-I92#	12-I92#	12-I96	12-I96#	12-I96#	12-I98	12-I98#	12-I98#	12-J02	12-J02#	12-J02#	
12-J04	12-J04#	12-J04#	12-J18	12-J18#	12-J18#	12-J30	12-J30#	12-J30#	12-J46	12-J46#	12-J46#	12-J49	12-J49#	
12-J51	12-J51#	12-J54	12-J54#	12-J54#	12-J58	12-J58#	12-J58#	12-J72	12-J72#	12-J72#	12-J93	12-J93#	12-J93#	
12-J96	12-J96#	12-J98	12-J98#	12-K01	12-K01#	12-K01#	12-K05	12-K05#	12-K05#	12-K07	12-K07#	12-K07#	12-K11	
12-K11#	12-K11#	12-K24	12-K24#	12-K24#	12-K40	12-K40#	12-K40#	12-K43	12-K43#	12-K45	12-K45#	12-K48	12-K48#	
12-K48#	12-K52	12-K52#	12-K52#	12-K54	12-K54#	12-K54#	12-K58	12-K58#	12-K58#	12-K71	12-K71#	12-K71#	12-K82	
12-K82#	12-K82#	12-K98	12-K98#	12-K98#	12-L01	12-L01#	12-L03	12-L03#	12-L06	12-L06#	12-L06#	12-L10	12-L10#	
12-L10#	12-L23	12-L23#	12-L23#	12-L40	12-L40#	12-L40#	12-L43	12-L43#	12-L45	12-L45#	12-L48	12-L48#	12-L48#	
12-L52	12-L52#	12-L52#	12-L54	12-L54#	12-L54#	12-L56	12-L56#	12-L56#	12-L69	12-L69#	12-L69#	12-L85	12-L85#	
12-L85#	12-L88	12-L88#	12-L90	12-L90#	12-L93	12-L93#	12-L93#	12-L97	12-L97#	12-L97#	12-L99	12-L99#	12-L99#	
12-M01	12-M01#	12-M01#	12-M03	12-M03#	12-M03#	12-M16	12-M16#	12-M16#	12-M27	12-M27#	12-M27#	12-M43	12-M43#	
12-M43#	12-M46	12-M46#	12-M48	12-M48#	12-M51	12-M51#	12-M51#	12-M55	12-M55#	12-M55#	12-M68	12-M68#	12-M68#	
12-M85	12-M85#	12-M85#	12-M88	12-M88#	12-M90	12-M90#	12-M93	12-M93#	12-M93#	12-M97	12-M97#	12-M97#	12-M99	
12-M99#	12-M99#	12-N01	12-N01#	12-N01#	12-N14	12-N14#	12-N14#	12-N30	12-N30#	12-N30#	12-N33	12-N33#	12-N35	
12-N35#	12-N38	12-N38#	12-N38#	12-N42	12-N42#	12-N42#	12-N44	12-N44#	12-N46	12-N46#	12-N46#	12-N46#	12-N48	
12-N48#	12-N48#	12-N61	12-N61#	12-N61#	12-N73	12-N73#	12-N73#	12-N91	12-N91#	12-N91#	12-N94	12-N94#	12-N96	
12-N96#	12-N99	12-N99#	12-N99#	12-003	12-003#	12-003#	12-016	12-016#	12-016#	12-033	12-033#	12-033#	12-036	
12-036#	12-038	12-038#	12-041	12-041#	12-041#	12-045	12-045#	12-045#	12-047	12-047#	12-047#	12-049	12-049#	
12-049#	12-062	12-062#	12-062#	12-078	12-078#	12-078#	12-081	12-081#	12-083	12-083#	12-086	12-086#	12-086#	
12-090	12-090#	12-090#	12-092	12-092#	12-092#	12-094	12-094#	12-094#	12-096	12-096#	12-096#	12-P11	12-P11#	
12-P11#	12-P22	12-P22#	12-P22#	12-P38	12-P38#	12-P38#	12-P41	12-P41#	12-P43	12-P43#	12-P46	12-P46#	12-P46#	
12-P50	12-P50#	12-P50#	12-P63	12-P63#	12-P63#	12-P75	12-P75#	12-P75#	12-P81	12-P81#	12-P81#	12-P84	12-P84#	
12-P91	12-P91#	12-P91#	12-P94	12-P94#	12-P96	12-P96#	12-P99	12-P99#	12-P99#	12-Q03	12-Q03#	12-Q03#	12-Q05	
12-Q05#	12-Q05#	12-Q07	12-Q07#	12-Q07#	12-Q19	12-Q19#	12-Q19#	12-Q32	12-Q32#	12-Q32#	12-Q41	12-Q41#	12-Q41#	
12-Q44	12-Q44#	12-Q52	12-Q52#	12-Q52#	12-Q55	12-Q55#	12-Q57	12-Q57#	12-Q60	12-Q60#	12-Q60#	12-Q64	12-Q64#	
12-Q64#	12-Q66	12-Q66#	12-Q66#	12-Q68	12-Q68#	12-Q68#								
FMT16	4-657#	12-107	12-254	12-396	12-571	12-719	12-864	12-:43	12-:95	12-=73	12-?56	12-A14	12-B02	12-B95
	12-C60	12-D00	12-D59	12-E85	12-F18	12-G49	12-H72	12-J27	12-K79	12-M24	12-N70	12-P19	16-60	17-32
	17-34	18-27	29-58	35-183										
FNC DTB	25-130	40-55#												
FNC MSK	4-504#	34-17	35-341											
GEN BUF	12-114	12-261	12-403	12-578	12-726	12-871	12-:50	12-:88	12-<02	12-=84	12->25	12-?67	12-a07	12-A21
GET	12-C63	12-D07	12-E88	12-F25	12-H79	12-J34	12-J83	12-K86	12-M31	12-N77	12-P26	17-20#	12-643	12-688
	12-67	12-134	12-176	12-221	12-281	12-323	12-367	12-422	12-464	12-510	12-529	12-600	12-643	12-688
	12-748	12-791	12-835	12-893	12-935	12-981	12-:00	12-:72	12-:21	12-:65	12-<24	12-<54	12-<82	12-=16
	12-=43	12->09	12->46	12->73	12-?06	12-?33	12-?91	12-a39	12-a83	12-A49	12-A91	12-B86	12-D37	12-D79
	12-E41	12-F57	12-G17	12-G79	12-G92	12-H34	12-H48	12-H99	12-I43	12-I92	12-J54	12-K01	12-K48	12-L06
	12-L48	12-L93	12-M51	12-M93	12-N38	12-N99	12-041	12-086	12-P46	12-P99	12-Q60	15-47	15-90	15-114
	15-144	15-168	15-199	16-76	16-97	16-129	16-163	16-192	20-31#					
GET BUF	6-0#	20-37												
GET INX	6-0#	12-67*	12-67*	12-529*	12-529*	12-:00*	12-:00*	19-13	20-38					
GET STS	12-129	12-171	12-216	12-276	12-318	12-362	12-417	12-459	12-505	12-595	12-638	12-683	12-743	12-786
	12-830	12-888	12-930	12-976	12-:67	12-:16	12-:60	12-<19	12-<49	12-=11	12->04	12->41	12->68	12-?01
	12-?28	12-?86	12-a34	12-a78	12-A44	12-A86	12-B81	12-D32	12-D74	12-E36	12-F52	12-G12	12-G76	12-H31
	12-H94	12-I38	12-I87	12-J49	12-J96	12-K43	12-L01	12-L43	12-L88	12-M46	12-M88	12-N33	12-N94	12-036

IPCK3 4-764#

H 11

SEQ 0343

C

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

T

OR

4-742#

J 11

SEQ 0345

NXXXE E EEEEEEE <CCCCCCCC> CCCCCCCCCC C

RGDTPT 42-3#

L 11

SEQ 0347

MEMO

RECEIVED

RMMR2

4-709#

C 12

SEQ 0351

RMMR21	6-0#	31-101	34-81	49-17										
RMMR20	6-0#													
RMOF	4-706#	12-122	12-162	12-206	12-269	12-309	12-352	12-410	12-450	12-495	12-586	12-629	12-673	12-734
	12-777	12-820	12-879	12-921	12-966	12-:58	12-:07	12-:50	12-<10	12-<70	12-:30	12-:96	12->59	12-?18
	12-?79	12-a25	12-a68	12-A37	12-A77	12-B71	12-C70	12-D23	12-D65	12-E26	12-E95	12-F43	12-G02	12-G67
	12-H21	12-H87	12-I29	12-I77	12-J42	12-J87	12-K33	12-K94	12-L34	12-L78	12-M39	12-M79	12-N23	12-N85
	12-027	12-071	12-P34	12-P70	12-Q26	16-67								
RMOFI	6-0#	18-21	18-27	25-377	25-487	35-344	35-420	49-14	49-16					
RMOFO	6-0#	12-107*	12-254*	12-396*	12-571*	12-719*	12-864*	12-:43*	12-:95*	12-:73*	12-?56*	12-A14*	12-C60*	12-D00*
	12-D59*	12-E85*	12-F18*	12-G49*	12-H72*	12-J27*	12-K79*	12-M24*	12-N70*	12-P19*	12-Q70	12-Q72*	16-60*	17-32
	29-58	35-183	49-11											
RMR	4-586#	32-51	32-83	32-87	33-232	33-250	33-253	35-128	35-159	35-162				
RMSN	4-705#													
RMSNI	6-0#	49-17												
RMSNO	6-0#													
RMWC	4-772#	12-11*	12-12	12-121	12-164	12-209	12-268	12-311	12-355	12-409	12-452	12-498	12-585	12-631
	12-676	12-733	12-779	12-823	12-878	12-923	12-969	12-:57	12-:09	12-:53	12-<09	12-<72	12-:33	12-:95
	12->61	12-?21	12-?78	12-a27	12-a71	12-A36	12-A79	12-B74	12-C73	12-D22	12-D67	12-E29	12-E98	12-F44
	12-G05	12-G69	12-H24	12-H86	12-I31	12-I80	12-J41	12-J89	12-K36	12-K93	12-L36	12-L81	12-M38	12-M81
	12-N26	12-N84	12-029	12-074	12-P33	12-P72	12-Q29	16-66						
RMWCI	6-0#	18-72	26-10	26-23	26-44	26-115	49-14							
RMWCO	6-0#	12-106*	12-157*	12-253*	12-304*	12-395*	12-444*	12-570*	12-624*	12-718*	12-772*	12-863*	12-915*	12-:42*
	12-:83*	12-:94*	12-<64*	12-:75*	12->20*	12-?58*	12-a03*	12-A13*	12-A72*	12-B67*	12-C59*	12-C99*	12-E84*	12-F17*
	12-G48*	12-H71*	12-I23*	12-I73*	12-J26*	12-J78*	12-K78*	12-L28*	12-M23*	12-M73*	12-N69*	12-O22*	12-P18*	12-P57*
	16-59*	16-302	17-22	18-73	26-24	26-45								
RQA	4-669#	31-102	34-82											
RQB	4-670#	31-102	34-82											
RTC	4-514#													
SA1	4-551#													
SA16	4-547#													
SA2	4-550#													
SA4	4-549#													
SAB	4-548#													
SADMSK	4-555#	26-68												
SAVREG	14-16	38-10#												
SC	4-723#	24-90	24-107	24-123	31-30									
SCO	4-604#													
SC1	4-603#													
SC2	4-602#													
SC3	4-601#													
SC4	4-600#													
SCOPE	4-491#	12-2	12-34	12-63	12-91	12-238	12-380	12-555	12-703	12-848	12-:27	12-:78	12-:63	12-?46
	12-a96	12-C82	12-F07	12-G39	12-H57	12-J12	12-K63	12-M08	12-N53	12-P04	13-9			
SCTMSG	12-110	12-257	12-399	12-574	12-722	12-867	12-:46	12-:98	12-:78	12-?61	12-A17	12-D03	12-F21	12-H75
	12-J30	12-K82	12-M27	12-N73	12-P22	39-3#								
SCTMSK	4-606#													
SEARCH	4-520#	25-104	26-5											
SECERR	12-184	12-229	12-331	12-375	12-472	12-547	12-651	12-696	12-799	12-843	12-943	12-:19	12-:29	12-:73
	12-<90	12-:51	12->81	12-?41	12-a47	12-a91	12-B49	12-C42	12-E04	12-E66	12-F74	12-G34	12-G96	12-H52
	12-I53	12-J02	12-K11	12-K58	12-L56	12-M01	12-N01	12-N46	12-O49	12-O94	12-Q07	12-Q68	25-24#	
SEEK	4-509#	12-<44	12-:06	12->36	12->96									
SEKSTS	12-<58	12-:20	12->50	12-?10	29-19#									
SHUT	13-16	13-20	37-13#											
SHUT2	37-22#	38-8	38-8											
SIZCLK	11-21	22-3#												
SKI	4-682#	15-176	16-148	29-48	29-51	29-70	29-100	29-102	29-106	29-117	29-179	31-115	33-92	33-116

33-119 33-218 35-173 35-176 35-193 35-204 35-206^{E 12} 35-207 35-209 35-221 35-513 36-74 36-89 36-108
SEQ 0353

TAP

4-648#

G 12

SEQ 0355

TYPOS	9-42	9-53	9-99	10-70	10-91	14-23	14-27	14-32	14-34	38-10#					
U0	4-749#														
U1	4-749#														
U2	4-749#														
UBUSQS	10-42	39-14#													
UNS	4-574#	12-G58	12-H13	31-70	32-51	32-55	32-59	33-232	33-264	33-269	35-279	35-282	35-287		
UNTMSK	4-753#	24-48	24-50												
UPE	4-736#	25-270													
USE	4-693#	12-804	12-897	17-29											
USRF IL	16-254	16-257	16-267*	16-275*	16-326	51-12#									
VV	4-568#	15-122	16-116	25-196	29-91	29-156	29-157	29-206	29-211	32-23	32-26	33-106	33-145	33-146	
	33-169	33-175	34-29	35-90	35-119	35-505	35-506	35-544	35-549	36-35	36-36	36-56	36-61		
WC	4-638#	31-80	34-70												
WCD	4-524#	12-349	12-492	12-817	12-963	12-:47	12-?15	12-@65	12-K30	12-Q47					
WCE	4-735#	12-516	12-522	12-987	12-992	25-169	40-77	40-78							
WCEHI	4-760#														
WCELO	4-761#														
WCF	4-583#	4-590	25-240	35-479	35-482										
WCH	4-525#														
WD	4-528#	12-158	12-305	12-446	12-625	12-773	12-917	12-:85	12-:03	12-<66	12->24	12->55	12-@02	12-@21	
	12-A73	12-D60	12-F19	12-F37	12-G50	12-G63	12-125	12-J80	12-L30	12-M75	12-023				
WH	4-529#	12-108	12-255	12-397	12-572	12-720	12-865	12-:44	12-:96	12-=76	12-?59	12-A15	12-C62	12-C67	
	12-D01	12-E87	12-E92	12-H73	12-J28	12-K80	12-M25	12-N71	12-P20	35-342					
WLE	4-577#	4-590	25-221	25-225	25-238	25-253	25-268	35-279	35-311	35-315	35-323	40-81	40-82		
WRL	4-563#	25-223	35-318												
XSIZ	9-67#	10-123	10-153												
XXDP	6-0#	9-30*	9-33*	9-34	9-36*	9-41	9-52	9-77	9-79						
ZEROS	12-112	12-259	12-401	12-:48	12->22	12-?65	12-A19	12-H77	12-P24	42-38#					

SSCMRE	4-791#													
SSCMTM	4-791#	5-0	5-0	5-0	5-0	5-0								
SSESCA	4-491#													
SSNEWT	4-491#	12-2	12-34	12-63	12-91	12-238	12-380	12-555	12-703	12-848	12-:27	12-;78	12-=63	12-?46
SSSET	12-a96	12-c82	12-f07	12-g39	12-h57	12-j12	12-k63	12-m08	12-n53	12-p04				
SSSETM	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10#
SSSKIP	9-19	9-19#												
.SACT1	4-491#													
.SAPT8	4-483#	4-786												
.SAPTB	4-483#	5-0	5-0#											
.SAPTH	4-483#	4-789												
.SAPTY	4-483#	38-12												
.SCATC	4-479#	4-784												
.SCMTA	4-480#	4-791												
.SEOP	4-480#	13-20												
.SERRO	4-480#	38-7												
.SERRT	4-480#													
.SPOWE	4-482#	38-11												
.SRDDE	4-481#													
.SRDOC	4-481#	38-9												
.SREAD	4-481#	38-8												
.SSAVE	4-482#	38-1												
.SSCOP	4-480#	38-6												
.SSIZE	4-482#													
.STRAP	4-482#	38-10												
.STYPB	4-481#	38-2												
.STYPD	4-481#	38-3												
.STYPE	4-480#	38-5												
.STYPO	4-481#	38-4												
.EQUAT	4-479#	4-491												
.HEADE	4-479#	4-487												
.SETUP	4-479#	4-782												
.SWRHI	4-479#	4-488												
.SWRLO	4-479#	4-488#	4-489											
CALCLR	4-194#	12-36	12-65											
CALSUB	4-208#	12-99	12-110	12-126	12-129	12-131	12-134	12-138	12-151	12-168	12-171	12-173	12-176	12-180
	12-182	12-184	12-197	12-213	12-216	12-218	12-221	12-225	12-227	12-229	12-233	12-246	12-257	12-273
	12-276	12-278	12-281	12-285	12-298	12-315	12-318	12-320	12-323	12-327	12-329	12-331	12-344	12-359
	12-362	12-364	12-367	12-371	12-373	12-375	12-388	12-399	12-414	12-417	12-419	12-422	12-426	12-439
	12-456	12-459	12-461	12-464	12-468	12-470	12-472	12-487	12-502	12-505	12-507	12-510	12-514	12-519
	12-547	12-563	12-574	12-592	12-595	12-597	12-600	12-604	12-618	12-635	12-638	12-640	12-643	12-647
	12-649	12-651	12-664	12-680	12-683	12-685	12-688	12-692	12-694	12-696	12-698	12-711	12-722	12-740
	12-743	12-745	12-748	12-752	12-766	12-783	12-786	12-788	12-791	12-795	12-797	12-799	12-812	12-827
	12-830	12-832	12-835	12-839	12-841	12-843	12-856	12-867	12-885	12-888	12-890	12-893	12-897	12-910
	12-927	12-930	12-932	12-935	12-939	12-941	12-943	12-958	12-973	12-976	12-978	12-981	12-985	12-989
	12-:19	12-:35	12-:46	12-:64	12-:67	12-:69	12-:72	12-:76	12-:98	12-:13	12-:16	12-:18	12-:21	12-:25
	12-:27	12-:29	12-:42	12-:57	12-:60	12-:62	12-:65	12-:69	12-:71	12-:73	12-:86	12-:98	12-<16	12-<19
	12-<21	12-<24	12-<28	12-<41	12-<46	12-<49	12-<51	12-<54	12-<58	12-<76	12-<79	12-<82	12-<86	12-<88
	12-<90	12-=03	12-=08	12-=11	12-=13	12-=16	12-=20	12-=37	12-=40	12-=43	12-=47	12-=49	12-=51	12-=53
	12-=78	12-=87	12->01	12->04	12->06	12->09	12->13	12->34	12->38	12->41	12->43	12->46	12->50	12->65
	12->68	12->70	12->73	12->77	12->79	12->81	12->94	12->98	12-?01	12-?03	12-?06	12-?10	12-?25	12-?28
	12-?30	12-?33	12-?37	12-?39	12-?41	12-?61	12-?70	12-?83	12-?86	12-?88	12-?91	12-?95	12-a16	12-a31
	12-a34	12-a36	12-a39	12-a43	12-a45	12-a47	12-a60	12-a75	12-a78	12-a80	12-a83	12-a87	12-a89	12-a91
	12-A06	12-A17	12-A41	12-A44	12-A46	12-A49	12-A53	12-A66	12-A83	12-A86	12-A88	12-A91	12-A95	12-B12
	12-B28	12-B40	12-B49	12-B61	12-B78	12-B81	12-B83	12-B86	12-B90	12-C05	12-C21	12-C33	12-C42	12-C65
	12-C76	12-C92	12-D03	12-D29	12-D32	12-D34	12-D37	12-D41	12-D54	12-D71	12-D74	12-D76	12-D79	12-D83

Table with 14 columns of alphanumeric identifiers (e.g., 12-E02, 12-F28, 12-G26, etc.) arranged in a grid.

COMME
NDCOM
ERROR

Table with 14 columns of alphanumeric identifiers, including entries like 4-491#, 4-12#, 11-23, 12-26, etc., continuing the cross-reference.

ESCAPE
GETPRI
GETREG
GETSWR
MSG
MULT
NEWTST
NWTST
POP

Table with 14 columns of alphanumeric identifiers, including entries like 4-491#, 4-171#, 12-67, 12-529, etc., continuing the cross-reference.

18-95

19-25

19-26

19-27

20-70

21-59

22-20^{M 12}

22-21

23-35

24-145

24-149

25-131

28-51

30-25
SEQ 0361

PUSH	31-58	34-58	34-59	38-1	38-3	38-9	38-11	38-11	38-12	38-12	18-17	19-10	19-11	19-12
	4-491#	12-5	12-6	12-38	12-39	16-47	16-263	16-282	16-298	17-20	34-51	34-52	38-1	38-3
	20-35	21-29	22-4	22-5	23-9	24-141	25-128	28-20	30-12	31-52				
	38-9	38-11	38-11	38-12	38-12	38-12								
PUTREG	4-449#	12-F32	12-F91	12-G58	12-H13									
REPORT	4-491#													
RGBFMC	4-77#	6-0	6-0											
SETPRI	4-491#	9-23	11-41	37-6	37-10	38-8								
SETTRA	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10#
SETUP	4-491#	9-19												
SKIP	4-491#													
SLASH	4-491#													
STARS	4-491#	4-786	4-789	4-789	4-789	5-0	5-0	5-0	12-2	12-2	12-34	12-34	12-63	12-63
	12-91	12-91	12-93	12-145	12-191	12-238	12-238	12-240	12-292	12-338	12-380	12-380	12-382	12-433
	12-478	12-555	12-555	12-557	12-611	12-658	12-703	12-703	12-705	12-759	12-806	12-848	12-848	12-850
	12-904	12-949	12-:27	12-:27	12-:29	12-:91	12-:36	12-:78	12-:78	12-:80	12-<35	12-<97	12-=63	12-=63
	12-=65	12->28	12->88	12-?46	12-?46	12-?48	12-@10	12-@54	12-@96	12-@96	12-@98	12-A60	12-B55	12-C55
	12-C57	12-C82	12-C82	12-C84	12-D48	12-E11	12-E80	12-E82	12-F07	12-F07	12-F09	12-F81	12-G39	12-G39
	12-G41	12-H03	12-H57	12-H57	12-H60	12-I10	12-I60	12-J12	12-J12	12-J15	12-J65	12-K18	12-K63	12-K63
	12-K65	12-L17	12-L63	12-M08	12-M08	12-M10	12-M62	12-N08	12-N53	12-N53	12-N55	12-O10	12-O56	12-P04
	12-P04	12-P06	12-P59	12-Q14	13-20	14-2	15-57	15-70	15-84	15-107	15-139	15-161	15-194	16-211
	16-291	16-295	25-26	25-34	25-119	26-1	26-3	38-1	38-2	38-3	38-4	38-5	38-6	38-7
	38-8	38-8	38-8	38-8	38-8	38-9	38-10	38-11	38-11	38-12				
SWRSU	4-491#	9-19	9-19#											
TAGS	4-111#	5-0												
TRMTRP	38-10#													
TYPBIN	4-491#													
TYPDEC	4-491#	13-20	13-20											
TYPNAM	4-479#	4-491#	9-25											
TYPNUM	4-491#													
TYPOCS	4-491#	9-99	14-23	14-27	14-32	14-34								
TYPOCT	4-491#	10-56	38-8											
TYPTXT	4-491#	9-6	9-39	9-50	9-56	9-57	11-24	13-20	13-20	37-18				
XPER	4-20#	7-4	7-7	7-10	7-13	7-16	7-19	7-22	7-25	7-28	7-31	7-34	7-37	7-40
	7-43	7-46	7-49	7-52	7-55	7-58	7-61	7-64	7-67	7-70	7-73	7-76	7-79	7-82
	7-85	7-88	7-91	7-94	7-97	7-100	7-103	7-106	7-109	7-112	7-115	7-118	7-121	7-124
	7-127	7-131	7-134	7-137	7-140	7-143	7-147	7-151	7-155	7-158	7-161	7-164	7-167	7-170
	7-173	7-176	7-179	7-183	7-186	7-189	7-192	7-195	7-198	7-201	7-204	7-207	7-210	7-213
	7-216	7-219	7-222	7-225	7-228	7-231	7-234	7-237	7-240	7-243	7-246	7-249	7-252	7-255
	7-258	7-261	7-264	7-267	7-270	7-273	7-276	7-279	7-282	7-285	7-288	7-291	7-294	7-297
	7-300	7-303	7-306	7-309	7-312	7-315	7-318	7-321	7-324	7-327	7-330	7-333	7-336	7-339
	7-342	7-345	7-348	7-352	7-355	7-358	7-361	7-364	7-367	7-370	7-373	7-376	7-379	7-382
	7-385	7-388	7-391	7-394	7-397	7-400	7-403	7-406	7-409	7-412	7-415	7-418	7-421	7-424
	7-427	7-430	7-433	7-436	7-439	7-442	7-445	7-448	7-451	7-454	7-457	7-460	7-463	7-466
	7-469	7-472	7-475	7-478	7-481	7-484	7-487	7-490	7-493	7-496	7-499	7-502	7-505	7-508
	7-511	7-514	7-517	7-520	7-523	7-526	7-529	7-532	7-535	7-538	7-541	7-544	7-547	7-550
	7-553	7-557	7-560	7-564	7-567	7-570	7-574	7-578	7-582	7-587	7-591	7-595	7-598	7-601
	7-604	7-607	7-610	7-613	7-616	7-619	7-622	7-625	7-628	7-631	7-634	7-637	7-640	7-643
	7-646	7-649	7-652	7-655	7-658	7-661	7-664	7-667	7-670	7-673	7-676	7-679	7-682	7-685
	7-688	7-691	7-694	7-697	7-700	7-703	7-706	7-709	7-712	7-715	7-718	7-721	7-724	