

RM80

FCTNL TEST PT 3
CZRNFAO

AH-T115A-MC
FICHE 1 OF 2

JUL 1982
COPYRIGHT © 1982
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows of small, dense text. Each cell in the grid contains a small table or data entry, which is difficult to read due to the low resolution and high density of the characters. The text appears to be organized in a structured format, possibly representing test results or configuration data for a system.

RM80

FCTNL TEST PT 3
CZRNFA0

AH-T115A-MC
FICHE 2 OF 2

JUL 1982
COPYRIGHT © 1982
MADE IN USA



The main body of the document is a large, dark grid containing numerous small, illegible entries. The grid is organized into approximately 10 columns and 20 rows. Each cell in the grid appears to contain a small table or data entry, but the text is too faint to be read. The overall appearance is that of a microfiche or a high-density data storage format.

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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-T114A-MC
PRODUCT NAME: CZRNFA0 RM80 FCTNL PT 3
PRODUCT DATE: APRIL 1, 1982
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

1. INTRODUCTION
 1. ABSTRACT
 2. UNIT UNDER TEST
2. OPERATING REQUIREMENTS
 1. HARDWARE REQUIREMENTS
 2. MEDIA REQUIREMENTS
 3. PREREQUISITE DIAGNOSTIC PROGRAMS
3. OPERATING PROCEDURE
 1. LOADING
 2. SWITCH OPTIONS
 3. STARTING
 4. HALTING
 5. RESTARTING
4. OPERATOR INTERFACE
 1. PROGRAM I.D.
 2. CONSOLE DIALOGUE
 3. PROGRESS REPORTS
 4. PERFORMANCE REPORTS
 5. PROGRAM HALTS
 6. ERROR REPORTS
 7. EXECUTION TIME
5. ENVIRONMENTAL SUPPORT
 1. PROCESSOR COMPATIBILITY
 2. DUAL PORT CONFIGURATIONS
 3. MEMORY PARITY HARDWARE
 4. MEMORY MANAGEMENT HARDWARE
 5. ACT, APT COMPATIBILITY
 6. XXDP COMPATIBILITY
 7. OPERATING SYSTEM COMPATIBILITY
6. TEST DESCRIPTION

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

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM80 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RM80 DISK SUBSYSTEM. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

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

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

TO VERIFY THE FUNCTIONALITY OF THE RM80 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

THE TEST IS COMPRISED OF 4 PARTS, WHICH WOULD NORMALLY BE RUN IN SEQUENCE, STARTING WITH PART 1. BRIEFLY, PART 1 TESTS HOUSEKEEPING AND MECHANICAL POSITIONING OPERATIONS; PART 2 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING HEADER AND DATA; PART 3 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING DATA; PART 4 TESTS MECHANICAL POSITIONING OPERATIONS AND VARIOUS TIMING PARAMETERS OF THE RM80 DISK DRIVE.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM80 DISK SUBSYSTEM WHICH CONSISTS OF THE RH70 MASSBUS CONTROLLER, THE RM80 MASSBUS ADAPTER, AND THE STORAGE MODULE DISK DRIVE. NOTE THAT A DISK IS REQUIRED FOR TESTING AND IS CONSIDERED AN INTEGRAL OF THE STORAGE MODULE DISK DRIVE.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RM80 SUBSYSTEM FUNCTIONAL TEST:

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

2.2 MEDIA REQUIREMENTS

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

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

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

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RM80 DISKLESS TEST, PART 1 & 2

RM80 FUNCTIONAL TEST, PART 1 & 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 MAY BE STARTED AT LOCATION 200 OR 204. STARTING AT 200 WILL BE THE NORMAL STARTING ADDRESS. STARTING AT 204 WILL ENABLE THE RH/RM BASE ADDRESS TO BE CHANGED.

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. THIS OF COURSE, DEPENDS ON WHICH TEST WAS BEING PERFORMED AT THE TIME THE PROGRAM WAS HALTED. ALSO, THIS ONLY APPLIES TO A NORMAL (ERROR FREE) OPERATION OF THE PROGRAM.

3.5 RESTARTING

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

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

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

EXAMPLE 1

```
RMCS1=176700 <CR>      :NO CHANGE IN ADDRESS
RMVEC=000254 <CR>      :NO CHANGE IN ADDRESS
```

EXAMPLE 2

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


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

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

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

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

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

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

'DRIVE(S) TO BE TESTED, 0'

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

'DRIVE(S) TO BE TESTED, NONE'

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

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

'DRIVE 0'

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

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

4.3 PROGRESS REPORTS

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

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

'END OF PASS 1'

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

'TOTAL ERRORS SINCE LAST REPORT 0'

4.4 PERFORMANCE REPORT

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

4.5 PROGRAM HALTS

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

4.6 ERROR REPORTS

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

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

```

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

```

4.7 EXECUTION TIME

PASS 1 OF THE PROGRAM TAKES ABOUT 20 SECONDS. PASS 2 AND SUBSEQUENT PASSES TAKE 50 SECONDS.

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

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM80 SUBSYSTEM FUNCTIONAL TEST IS EXECUTABLE ON A PDP-11/70 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET, AND PROVIDING THAT DATA THROUGHPUT ON THE SYSTEM IS SUFFICIENT TO SUSTAIN DATA TRANSFER OPERATIONS.

5.2 DUAL PORT CONFIGURATIONS

THE RM80 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RM80 ADAPTER BUT IS EXECUTABLE ON RM80 SUBSYSTEMS HAVING THE DUAL PORT OPTION PROVIDING THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE IS NOT USED DURING THE EXECUTION OF THE RM80 SUBSYSTEM FUNCTIONAL TEST.

5.4 MEMORY MANAGEMENT HARDWARE

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

5.5 ACT11, APT11 COMPATIBILITY

THE RM80 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES. FURTHER, THE PROGRAM WILL EXECUTE A QUICK PASS DURING THE FIRST PASS IN SUPPORT OF QUICK VERIFY MODE.

5.6 XXDP COMPATIBILITY

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

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

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

6.0 TEST DESCRIPTION

TEST 1 CONTROLLER ACCESS TEST

PURPOSE:

TO VERIFY THAT THE UNIBUS ADDRESS OF THE RM80 SUBSYSTEM IS CORRECT, AS DEFINED AT LOCATION \$BASE.

PROCEDURE:

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

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

TEST 2 - 22 WRITE/READ DATA TESTS

PURPOSE:

TO TEST WRITE DATA AND READ DATA FUNCTIONALITY OF THE RM80 SUBSYSTEM USING A SET OF VARIABLES WHICH INCLUDE WORD COUNT, HEAD MOTION, HEAD SWITCHING AND ERROR CONDITIONS.

PROCEDURE:

ALTHOUGH EACH TEST EXERCISES A DIFFERENT VARIABLE, THE GENERAL PROCEDURE OF EACH TEST IS THE SAME. THE DRIVE IS INITIALIZED AND RECALIBRATED IF "PIP" OR "SKI" ARE ACTIVE SO THAT THERE ARE NO ERRORS WHEN A TEST BEGINS. 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

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

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: THE SECTOR USED DURING A TEST MAY DIFFER FROM THE PROGRAM LISTING BECAUSE THE PROGRAM SUBSTITUTES A GOOD SECTOR, IF THE ONE SELECTED IS LISTED IN THE BAD BLOCK TABLE.

TEST 2 WRITE, READ ZEROS TEST

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

TEST 3 WRITE, WRITE CHECK ZEROS TEST

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

TEST 4 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.

TEST 5 WRITE, READ ONES TEST

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

TEST 6 WRITE, WRITE CHECK ONES TEST

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

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

TEST 7 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.

TEST 10 WRITE, WRITE CHECK MULTIPLE SECTORS TEST

THE TEST SEEKS TO CYLINDER 559. TO INSURE THERE IS NO HEAD 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. THE TEST IS REPEATED FOR 16 BIT FORMAT WITH SSEI BIT SET.

TEST 11 WRITE, READ W/ IMPLIED SEEK TEST

THIS TEST SEEKS TO THE FIRST CYLINDER PRIOR TO WRITING DATA ON CYLINDER 559., TRACK 2 AND 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.

TEST 12 READ, WRITE CHECK W/ HEAD SWITCHING TEST

THE TEST SEEKS TO CYLINDER 559. TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE READ DATA COMMAND STARTS WITH CYLINDER 559., TRACK 2, SECTOR 30. THE WORD COUNT IS EQUAL TO MULTIPLE SECTORS WHICH CAUSES THE SUBSYSTEM TO SWITCH FROM TRACK 2 TO TRACK 3 AFTER THE FIRST SECTOR IS WRITTEN. THE SAME SECTORS ARE VERIFIED WITH A WRITE CHECK DATA COMMAND, USING THE SAME WORD COUNT AND STARTING SECTOR. IF THE 'SSE' BIT IS SET AFTER EXECUTING A DATA COMMAND, THE 16 BIT MODE WILL BE TERMINATED AND THE TEST IS REPEATED FOR 16 BIT FORMAT WITH THE SSEI BIT SET, WHERE SECTOR 31. IS THE LAST SECTOR. IF NO 'SSE' WAS SET, THEN BOTH FORMAT MODES WILL BE EXECUTED.

TEST 13 READ, WRITE CHECK W/ MID-TRANSFER SEEK 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

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

TEST 14 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.

TEST 15 WRITE, READ W/ HCI & SSEI TEST

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

TEST 16 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.

TEST 17 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.

TEST 20 WRITE, READ EARLY PEAK SHIFT TEST

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

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

TEST 21 WRITE, READ EACH TRACK W/ MIXED PEAK SHIFT TEST

THE TEST WRITES AND READS WITH EACH TRACK USING A MIX OF EARLY, NORMAL AND LATE PEAK SHIFTS.

TEST 22 READ 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.

TEST 23 FORMAT FE CYLINDERS

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

1
487
488

:*LAST REVISION 08-OCT-81

.TITLE CZRNFAO RM80 FCTNL PT3
:*COPYRIGHT (C) 1982
:*DIGITAL EQUIPMENT CORPORATION
:*COLORADO SPGS., CO. 80919

:*PROGRAM BY MIKE LEAVITT

:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

489

.SBTTL OPERATIONAL SWITCH SETTINGS

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

490

491
492

.SBTTL BASIC DEFINITIONS

001100	STACK = 1100	*** INITIAL ADDRESS OF THE STACK POINTER ***
104000	ERROR = EMT	:::BASIC DEFINITION OF ERROR CALL
000004	SCOPE = IOT	:::BASIC DEFINITION OF SCOPE CALL

:*MISCELLANEOUS DEFINITIONS

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

:*GENERAL PURPOSE REGISTER DEFINITIONS

000000	R0 = %0	:::GENERAL REGISTER
000001	R1 = %1	:::GENERAL REGISTER
000002	R2 = %2	:::GENERAL REGISTER
000003	R3 = %3	:::GENERAL REGISTER

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

.*PRIORITY LEVEL DEFINITIONS

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

.*"SWITCH REGISTER" SWITCH DEFINITIONS

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

493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520

.SBTTL RM80 REGISTER BIT DEFINITIONS

;*RMCS1 CONTROL STATUS REGISTER

```

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

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

```

000000 NOP = 000000      ;;NOP COMMAND
000002 ILF02 = 000002    ;;ILLEGAL COMMAND
000004 SEEK = 000004     ;;SEEK COMMAND
000006 RECAL = 000006    ;;RECALIBRATE COMMAND
000010 DRVCLR = 000010   ;;DRIVE CLEAR COMMAND
000012 RELEASE = 000012  ;;RELEASE COMMAND
000014 OFFSET = 000014  ;;OFFSET COMMAND
000016 RTC = 000016     ;;RETURN TO CENTERLINE COMMAND
000020 RIP = 000020     ;;READ IN PRESET COMMAND
000022 PAKACK = 000022   ;;PACK ACKNOWLEDGE COMMAND
000022 PACACK = PAKACK
000024 ILF24 = 000024    ;;ILLEGAL COMMAND
000026 ILF26 = 000026    ;;ILLEGAL COMMAND
    
```

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

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

630	020000	EBL	= BIT13	:END OF BLOCK
631	010000	REX	= BIT12	:EXCEPTION
632	004000	ESRC	= BIT11	:ENABLE SEARCH
633	002000	PLFS	= BIT10	:LOOKING FOR SYNC
634	001000	ECRC	= BIT09	:ENABLE CRC OUT
635	000400	PDA	= BIT08	:DATA AREA
636	000200	PHA	= BIT07	:HEADER AREA
637	000100	CONT	= BIT06	:CONTINUE
638	000040	WC	= BIT05	:WORD CLOCK
639	000020	EECC	= BIT04	:ENABLE ECC OUT
640	000010	MWD	= BIT03	:WRITE DATA BIT
641	000004	LS	= BIT02	:LAST SECTOR
642	000002	LST	= BIT01	:LAST SECTOR AND TRACK
643	000001	DMD	= BIT00	:DIAGNOSTIC MODE
644				
645		:*RMDT DRIVE TYPE REGISTER		
646				
647	100000	NSA	= BIT15	:NOT SECTOR ADDRESSED = 0
648	040000	TAP	= BIT14	:TAPE DRIVE = 0
649	020000	MOH	= BIT13	:MOVING HEAD = 1
650	004000	DRQ	= BIT11	:DRIVE REQUEST REQUIRED
651				
652	020026	SNGPRT	= 020026	:SINGLE PORT DRIVE TYPE
653	024026	DULPRT	= 024026	:DUAL PORT DRIVE TYPE
654				
655		:*RMOF OFFSET REGISTER		
656				
657	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
658	004000	ECI	= BIT11	:ECC INHIBIT
659	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
660	001000	SSEI	= BIT09	:SKIP SECTOR ERROR INHIBIT
661	000200	OFD	= BIT07	:OFFSET FORWARD
662				
663		:*RMDC DESIRED CYLINDER ADDRESS REGISTER		
664				
665	001777	CYLSK	= 001777	:MASK FOR CYLINDER ADDRESS
666				
667		:*RMR2 MAINTENANCE REGISTER #2		
668				
669		:READ ONLY BITS		
670	100000	RQA	= BIT15	:PORT A REQUEST
671	040000	RQB	= BIT14	:PORT B REQUEST
672	020000	TAG	= BIT13	:TAG CONTROL
673	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
674	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
675	002000	CH	= BIT10	:CONTROL OR HEAD TAG
676	001000	BB09	= BIT09	:TAG BUS
677	000400	BB08	= BIT08	:TAG BUS
678	000200	BB07	= BIT07	:TAG BUS
	000100	BB06	= BIT06	:TAG BUS
	000040	BB05	= BIT05	:TAG BUS
	000020	BB04	= BIT04	:TAG BUS
	000010	BB03	= BIT03	:TAG BUS
	000004	BB02	= BIT02	:TAG BUS
	000002	BB01	= BIT01	:TAG BUS
	000001	BB00	= BIT00	:TAG BUS


```

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

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

1
2
3
4
5
6
7
8
9

000000
000174 000174
000174 000000
000174 000000
000200 000137 005434
000204 000137 005424
000046 000210
000046 000046
000046 033756
000052 000052
000052 000000
000052 000210
001100
000024 001100
000024 000024
000044 000200
000044 000044
000044 001100
001100 001100
001100 000000
001102 001222
001104 000024
001106 000024
001110 000024
001112 000042
001112 001114

```
.SBTTL TRAP CATCHER
    =0
    ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
    ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
    ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
    =174
DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)
    JMP @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
    JMP @#START1     ;CHANGE RH/RM BUS ADDRESS

.SBTTL ACT11 HOOKS
    ;*****
    ;HOOKS REQUIRED BY ACT11
    $SVPC=.          ;SAVE PC
    =46              ;:1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
    $SENDAD          ;:2)SET LOC.52 TO ZERO
    =52              ;; RESTORE PC
    .WORD 0
    =$SVPC

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

    $APTHD:
    $HIBTS: .WORD 0    ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
    $MADR:  .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
    $STMT:  .WORD 20.  ;;RUN TIM OF LONGEST TEST
    $PASTM: .WORD 20.  ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
    $UNITM: .WORD 20.  ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
    TAGADR: .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
001114 000000
001116 000
001117 000
001120 000000
001122 000000
001124 000000
001126 000000
001130 000
001131 001
001132 000000
001134 000000
001136 000000
001140 000000
001142 000000
001144 000000
001146 000000
001150 000
001151 000
001152 000000
001154 177570
001156 177570
001160 177560
001162 177562
001164 177564
001166 177566
001170 000
001171 002
001172 012
001173 000
001174 000000
001176 000000
001200 000000
001202 000000
001204 000000
001206 000000
001210 000000
001212 207 377 377
001216 077
001217 015
001220 012 000

SCMTAG: .TAGADR
STSTNM: .WORD 0
SERFLG: .BYTE 0
SICNT: .WORD 0
SLPADR: .WORD 0
SLPERR: .WORD 0
SERTTL: .WORD 0
SITEMB: .BYTE 0
SERMAX: .BYTE 1
SERRPC: .WORD 0
SGDADR: .WORD 0
SBDADR: .WORD 0
SGDDAT: .WORD 0
SBDDAT: .WORD 0
SAUTOB: .BYTE 0
SINTAG: .BYTE 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$TMP0: .WORD 0
\$TMP1: .WORD 0
\$TMP2: .WORD 0
\$TMP3: .WORD 0
\$TMP4: .WORD 0
\$TIMES: 0
\$ESCAPE: 0
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

:::START OF COMMON TAGS
:::CONTAINS THE TEST NUMBER
:::CONTAINS ERROR FLAG
:::CONTAINS SUBTEST ITERATION COUNT
:::CONTAINS SCOPE LOOP ADDRESS
:::CONTAINS SCOPE RETURN FOR ERRORS
:::CONTAINS TOTAL ERRORS DETECTED
:::CONTAINS ITEM CONTROL BYTE
:::CONTAINS MAX. ERRORS PER TEST
:::CONTAINS PC OF LAST ERROR INSTRUCTION
:::CONTAINS ADDRESS OF 'GOOD' DATA
:::CONTAINS ADDRESS OF 'BAD' DATA
:::CONTAINS 'GOOD' DATA
:::CONTAINS 'BAD' DATA
:::RESERVED--NOT TO BE USED
:::AUTOMATIC MODE INDICATOR
:::INTERRUPT MODE INDICATOR
:::ADDRESS OF SWITCH REGISTER
:::ADDRESS OF DISPLAY REGISTER
:::TTY KBD STATUS
:::TTY KBD BUFFER
:::TTY PRINTER STATUS REG. ADDRESS
:::TTY PRINTER BUFFER REG. ADDRESS
:::CONTAINS NULL CHARACTER FOR FILLS
:::CONTAINS # OF FILLER CHARACTERS REQUIRED
:::INSERT FILL CHARS. AFTER A 'LINE FEED'
:::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
:::USER DEFINED
:::USER DEFINED
:::USER DEFINED
:::USER DEFINED
:::USER DEFINED
:::MAX. NUMBER OF ITERATIONS
:::ESCAPE ON ERROR ADDRESS
:::CODE FOR BELL
:::QUESTION MARK
:::CARRIAGE RETURN
:::LINE FEED

.SBTTL APT MAILBOX-ETABLE

001222
001222 000000
001224 000000
001226 000000

SMAIL: .EVEN
\$MSGTY: .WORD AMSGTY
\$FATAL: .WORD AFATAL
\$TESTN: .WORD ATESTN

:::APT MAILBOX
:::MESSAGE TYPE CODE
:::FATAL ERROR NUMBER
:::TEST NUMBER

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

.SBTTL USER DEFINED TAGS

001326 000000
 001330 000000
 001332 000000

CTLFG: .WORD 0 ;CONTAINS CONTROL-C FLAG
 CHGADR: .WORD 0 ;CHANGE RH/RM BUS ADDRESS = -1, NO CHANGE = 0
 XXDP: .WORD 0 ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
 ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
 ;'XXDP' DEVICE CODE FOR THE RM80.

001334 000
 001335 015

LSTRK: .BYTE 0 ;LO BYTE = 0
 .BYTE 13. ;HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT
 ;UNDER TEST. RM80 = 13.

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

001336

GETBUF:

001336 000000
 001340 000000
 001342 000000
 001344 000000
 001346 000000
 001350 000000
 001352 000000
 001354 000000
 001356 000000
 001360 000000
 001362 000000
 001364 000000
 001366 000000
 001370 000000
 001372 000000
 001374 000000
 001376 000000
 001400 000000
 001402 000000
 001404 000000
 001406 000000
 001410 000000

;REGISTER INPUT BUFFER
 RMCS1I: .WORD 0 ;CONTROL, STATUS REGISTER #1
 RMWCI: .WORD 0 ;WORD COUNT REGISTER
 RMBAI: .WORD 0 ;BUS ADDRESS REGISTER
 RMDAI: .WORD 0 ;DISK ADDRESS REGISTER
 RMCS2I: .WORD 0 ;CONTROL, STATUS REGISTER #2
 RMDSI: .WORD 0 ;DRIVE STATUS REGISTER
 RMER1I: .WORD 0 ;ERROR REGISTER #1
 RMASI: .WORD 0 ;ATTENTION SUMMARY REGISTER
 RMLAI: .WORD 0 ;LOOK AHEAD REGISTER
 RMDBI: .WORD 0 ;DATA BUFFER
 RMMR1I: .WORD 0 ;MAINTENANCE REGISTER #1
 RMDTI: .WORD 0 ;DRIVE TYPE REGISTER
 RMSNI: .WORD 0 ;SERIAL NUMBER REGISTER
 RMOFI: .WORD 0 ;OFFSET REGISTER
 RMDCI: .WORD 0 ;DESIRED CYLINDER REGISTER
 RMHRI: .WORD 0 ;HOLDING REGISTER
 RMMR2I: .WORD 0 ;MAINTENANCE REGISTER #2
 RMER2I: .WORD 0 ;ERROR REGISTER #2
 RMEC1I: .WORD 0 ;ECC POSITION REGISTER
 RMEC2I: .WORD 0 ;ECC PATTERN REGISTER
 RMBAEI: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER
 RMCS3I: .WORD 0 ;CONTROL, STATUS REGISTER #3

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

001412

PUTBUF:

001412 000000
 001414 000000
 001416 000000
 001420 000000
 001422 000000
 001424 000000
 001426 000000
 001430 000000
 001432 000000
 001434 000000
 001436 000000

;REGISTER OUTPUT BUFFER
 RMCS1O: .WORD 0 ;CONTROL, STATUS REGISTER #1
 RMWCO: .WORD 0 ;WORD COUNT REGISTER
 RMBAO: .WORD 0 ;BUS ADDRESS REGISTER
 RMDAO: .WORD 0 ;DISK ADDRESS REGISTER
 RMCS2O: .WORD 0 ;CONTROL, STATUS REGISTER #2
 RMDSO: .WORD 0 ;DRIVE STATUS REGISTER
 RMER1O: .WORD 0 ;ERROR REGISTER #1
 RMASO: .WORD 0 ;ATTENTION SUMMARY REGISTER
 RMLAO: .WORD 0 ;LOOK AHEAD REGISTER
 RMDBO: .WORD 0 ;DATA BUFFER
 RMMR1O: .WORD 0 ;MAINTENANCE REGISTER #1

001440 000000
 001442 000000
 001444 000000
 001446 000000
 001450 000000
 001452 000000
 001454 000000
 001456 000000
 001460 000000
 001462 000000
 001464 000000

RMDTO: .WORD 0 ;DRIVE TYPE REGISTER
 RMSNO: .WORD 0 ;SERIAL NUMBER REGISTER
 RMOFO: .WORD 0 ;OFFSET REGISTER
 RMDCO: .WORD 0 ;DESIRED CYLINDER REGISTER
 RMRHO: .WORD 0 ;HOLDING REGISTER
 RMR20: .WORD 0 ;MAINTENANCE REGISTER #2
 RMR20: .WORD 0 ;ERROR REGISTER #2
 RMEC10: .WORD 0 ;ECC POSITION REGISTER
 RMEC20: .WORD 0 ;ECC PATTERN REGISTER
 RMBAE0: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER
 RMCS30: .WORD 0 ;CONTROL, STATUS REGISTER #3

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

001466 000000
 001470
 001510 000000

TSTQUE: .WORD 0 ;CONTAINS DEVICE POINTER
 .BLKW 8. ;TEST QUE FOR DEVICES UNDER TEST
 .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

:SKIP SECTOR FAILURE ENABLE IS SET IF THE "SSF" BIT SHOULD BE
 :SET IN THE FIRST HEADER WORD WHEN THE FE CYLINDERS ARE BEING
 :REFORMATTED.
 SSFENB: .WORD 0 ;SSF ENABLE

001514 000000

:LOCATIONS "ASNDC" AND "ASNDC" CONTAIN THE CYLINDER, TRACK AND SECTOR
 :ADDRESS ASSIGNED BY THE BAD SECTOR MODULE.
 ASNDC: .WORD 0 ;ASSIGNED DESIRED CYLINDER
 ASNCA: .WORD 0 ;ASSIGNED TRACK, AND SECTOR
 CLKADR: .WORD 0 ;UNIBUS ADDRESS OF KW11 CLOCK
 CLKVCT: .WORD 0 ;VECTOR ADDRESS OF KW11 CLOCK

001516 000000
 001520 000000
 001522 000000
 001524 000000

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

001526

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

001555

:PUT TAGS HERE

.SBTTL ERROR POINTER TABLE

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

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

1	001604		\$ERRTB:		
2			:ERROR 1	WRONG UNIT SELECTED	
3	001604	070002		EMT1	
	001606	074140		EHT1	
	001610	074264		EDT1	
	001612	074354		EFT1	
4			:ERROR 2	DEVICE WENT UNAVAILABLE	
5				EMT2	
6	001614	070006		EHT1	
	001616	074140		EDT1	
	001620	074264		EFT1	
	001622	074354			
7			:ERROR 3	DEVICE WENT NONEXISTENT	
8				EMT3	
9	001624	070014		EHT1	
	001626	074140		EDT1	
	001630	074264		EFT1	
	001632	074354			
10			:ERROR 4	CONTROLLER NOT READY	
11				EMT4	
12	001634	070022		EHT1	
	001636	074140		EDT1	
	001640	074264		EFT1	
	001642	074354			
13			:ERROR 5	DRIVE NOT READY AND GO NOT RESET	
14				EMT5	
15	001644	070030		EHT1	
	001646	074140		EDT1	
	001650	074264		EFT1	
	001652	074354			
16			:ERROR 6	UNEXPECTED VALUE FOR "ATA" STATUS	
17					

18	001654	070036	EMT6	
	001656	074140	EHT1	
	001660	074264	EDT1	
	001662	074354	EFT1	
19				
20			:ERROR 7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER
21	001664	070044	EMT7	
	001666	000000	0	
	001670	000000	0	
	001672	000000	0	
22				
23			:ERROR 10	DRIVE NOT READY BUT GO IS RESET
24	001674	070052	EMT10	
	001676	074140	EHT1	
	001700	074264	EDT1	
	001702	074354	EFT1	
25				
26			:ERROR 11	GO NOT RESET BUT DRIVE IS READY
27	001704	070056	EMT11	
	001706	074140	EHT1	
	001710	074264	EDT1	
	001712	074354	EFT1	
28				
29			:ERROR 12	INCORRECT FUNCTION CODE
30	001714	070062	EMT12	
	001716	074140	EHT1	
	001720	074264	EDT1	
	001722	074354	EFT1	
31				
32			:ERROR 13	PARITY ERROR READING REMOTE REGISTERS
33	001724	070070	EMT13	
	001726	074140	EHT1	
	001730	074264	EDT1	
	001732	074354	EFT1	
34				
35			:ERROR 14	TRANSFER ERROR IS INCORRECT
36	001734	070102	EMT14	
	001736	074140	EHT1	
	001740	074264	EDT1	
	001742	074354	EFT1	
37				
38			:ERROR 15	INCORRECT WORD COUNT
39				

	001744	070110		EMT15
	001746	074140		EHT1
	001750	074264		EDT1
	001752	074354		EFT1
40				
41			:ERROR 16	INCORRECT BUS ADDRESS
42				
	001754	070116		EMT16
	001756	074140		EHT1
	001760	074264		EDT1
	001762	074354		EFT1
43				
44			:ERROR 17	INCORRECT LBT STATUS
45				
	001764	070126		EMT17
	001766	074140		EHT1
	001770	074264		EDT1
	001772	074354		EFT1
46				
47			:ERROR 20	INCORRECT AOE
48				
	001774	070136		EMT20
	001776	074140		EHT1
	002000	074264		EDT1
	002002	074354		EFT1
49				
50			:ERROR 21	INCORRECT DISK ADDRESS
51				
	002004	070146		EMT21
	002006	074140		EHT1
	002010	074264		EDT1
	002012	074354		EFT1
52				
53			:ERROR 22	INCORRECT CYLINDER ADDRESS
54				
	002014	070156		EMT22
	002016	074140		EHT1
	002020	074264		EDT1
	002022	074354		EFT1
55				
56			:ERROR 23	INCORRECT WLE STATUS
57				
	002024	070166		EMT23
	002026	074140		EHT1
	002030	074264		EDT1
	002032	074354		EFT1
58				
59			:ERROR 24	INCORRECT UPE STATUS
60				
	002034	070176		EMT24

	002036	074140	EHT1	
	002040	074264	EDT1	
	002042	074354	EFT1	
61				
62				:ERROR 25 INCORRECT WCF STATUS
63				
	002044	070206	EMT25	
	002046	074140	EHT1	
	002050	074264	EDT1	
	002052	074354	EFT1	
64				
65				:ERROR 26 INCORRECT WCE STATUS
66				
	002054	070216	EMT26	
	002056	074140	EHT1	
	002060	074264	EDT1	
	002062	074354	EFT1	
67				
68				:ERROR 27 INCORRECT MDPE STATUS
69				
	002064	070226	EMT27	
	002066	074140	EHT1	
	002070	074264	EDT1	
	002072	074354	EFT1	
70				
71				:ERROR 30 INCORRECT DCK STATUS
72				
	002074	070236	EMT30	
	002076	074140	EHT1	
	002100	074264	EDT1	
	002102	074354	EFT1	
73				
74				:ERROR 31 INCORRECT ECH STATUS
75				
	002104	070246	EMT31	
	002106	074140	EHT1	
	002110	074264	EDT1	
	002112	074354	EFT1	
76				
77				:ERROR 32 DLT SHOULD NOT BE SET
78				
	002114	070256	EMT32	
	002116	074140	EHT1	
	002120	074264	EDT1	
	002122	074354	EFT1	
79				
80				:ERROR 33 MXF SHOULD NOT BE SET
81				
	002124	070266	EMT33	
	002126	074140	EHT1	

	002130	074264	EDT1	
	002132	074354	EFT1	
82				
83				:ERROR 34 DTE SHOULD NOT BE SET
84				
	002134	070276	EMT34	
	002136	074140	EHT1	
	002140	074264	EDT1	
	002142	074354	EFT1	
85				
86				:ERROR 35 INCORRECT HCRC STATUS
87				
	002144	070306	EMT35	
	002146	074140	EHT1	
	002150	074264	EDT1	
	002152	074354	EFT1	
88				
89				:ERROR 36 INCORRECT HCE STATUS
90				
	002154	070316	EMT36	
	002156	074140	EHT1	
	002160	074264	EDT1	
	002162	074354	EFT1	
91				
92				:ERROR 37 INCORRECT FER STATUS
93				
	002164	070326	EMT37	
	002166	074140	EHT1	
	002170	074264	EDT1	
	002172	074354	EFT1	
94				
95				:ERROR 40 DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
96				
	002174	070336	EMT40	
	002176	074140	EHT1	
	002200	074264	EDT1	
	002202	074354	EFT1	
97				
98				:ERROR 41 LOST 'MOL' DURING PACK ACKNOWLEDGE
99				
	002204	070344	EMT41	
	002206	074140	EHT1	
	002210	074264	EDT1	
	002212	074354	EFT1	
100				
101				:ERROR 42 UNSAFE ERROR DURING PACK ACKNOWLEDGE
102				
	002214	070354	EMT42	
	002216	074140	EHT1	
	002220	074264	EDT1	

	002222	074354	EFT1	
103				
104			:ERROR 43	'OPI' ERROR DURING PACK ACKNOWLEDGE
105				
	002224	070366	EMT43	
	002226	074140	EHT1	
	002230	074264	EDT1	
	002232	074354	EFT1	
106				
107			:ERROR 44	'RMR' ERROR DURING PACK ACKNOWLEDGE
108				
	002234	070376	EMT44	
	002236	074140	EHT1	
	002240	074264	EDT1	
	002242	074354	EFT1	
109				
110			:ERROR 45	'ILR' ERROR DURING PACK ACKNOWLEDGE
111				
	002244	070406	EMT45	
	002246	074140	EHT1	
	002250	074264	EDT1	
	002252	074354	EFT1	
112				
113			:ERROR 46	'ILF' ERROR DURING PACK ACKNOWLEDGE
114				
	002254	070416	EMT46	
	002256	074140	EHT1	
	002260	074264	EDT1	
	002262	074354	EFT1	
115				
116			:ERROR 47	COMPOSITE ERROR STATUS IS INCORRECT
117				
	002264	070426	EMT47	
	002266	074140	EHT1	
	002270	074264	EDT1	
	002272	074354	EFT1	
118				
119			:ERROR 50	PARITY ERROR WRITING REMOTE REGISTERS
120				
	002274	070434	EMT50	
	002276	074140	EHT1	
	002300	074264	EDT1	
	002302	074354	EFT1	
121				
122			:ERROR 51	INCORRECT IAE STATUS DURING SEEK COMMAND
123				
	002304	070444	EMT51	
	002306	074140	EHT1	
	002310	074264	EDT1	
	002312	074354	EFT1	

124				
125			:ERROR 52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE
126				
	002314	070456	EMT52	
	002316	074140	EHT1	
	002320	074264	EDT1	
	002322	074354	EFT1	
127				
128			:ERROR 53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE, ASSUME
129			:	ON CYLINDER LATCH DIDN'T RESET
130				
	002324	070474	EMT53	
	002326	074140	EHT1	
	002330	074264	EDT1	
	002332	074354	EFT1	
131				
132			:ERROR 54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND
133				
	002334	070512	EMT54	
	002336	074140	EHT1	
	002340	074264	EDT1	
	002342	074354	EFT1	
134				
135			:ERROR 55	DEVICE CHECK DURING SEEK COMMAND
136				
	002344	070522	EMT55	
	002346	074140	EHT1	
	002350	074264	EDT1	
	002352	074354	EFT1	
137				
138			:ERROR 56	PIP IS STILL SET AFTER SEEK - SKI IS RESET
139				
	002354	070534	EMT56	
	002356	074140	EHT1	
	002360	074264	EDT1	
	002362	074354	EFT1	
140				
141			:ERROR 57	ATA DID NOT SET DURING SEEK COMMAND
142				
	002364	070552	EMT57	
	002366	074140	EHT1	
	002370	074264	EDT1	
	002372	074354	EFT1	
143				
144			:ERROR 60	IVC ERROR DURING SEEK COMMAND - LOST
145			:	VOLUME VALID
146				
	002374	070562	EMT60	
	002376	074140	EHT1	
	002400	074264	EDT1	

	002402	074354	EFT1	
147				
148			:ERROR 61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
149			:	VOLUME VALID IS STIL SET
150				
	002404	070600	EMT61	
	002406	074140	EHT1	
	002410	074264	EDT1	
	002412	074354	EFT1	
151				
152			:ERROR 62	MOL IS ZERO, BUT OPI WAS NOT
153			:	REPORTED DURING SEEK COMMAND
154				
	002414	070620	EMT62	
	002416	074140	EHT1	
	002420	074264	EDT1	
	002422	074354	EFT1	
155				
156			:ERROR 63	SSE SHOULD NOT BE SET
157				
	002424	070634	EMT63	
	002426	074140	EHT1	
	002430	074264	EDT1	
	002432	074354	EFT1	
158				
159			:ERROR 64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
160				
	002434	070644	EMT64	
	002436	074140	EHT1	
	002440	074264	EDT1	
	002442	074354	EFT1	
161				
162			:ERROR 65	DRIVE EXECUTED A SEEK WITH ERROR SET
163				
	002444	070664	EMT65	
	002446	074140	EHT1	
	002450	074264	EDT1	
	002452	074354	EFT1	
164				
165			:ERROR 66	UNEXPECTED ERROR SET IN RMER1
166				
	002454	070704	EMT66	
	002456	074140	EHT1	
	002460	074264	EDT1	
	002462	074354	EFT1	
167				
168			:ERROR 67	UNEXPECTED ERROR SET IN RMER2
169				
	002464	070716	EMT67	
	002466	074140	EHT1	

002470	074264	EDT1	
002472	074354	EFT1	
170			
171		:ERROR 70	ERRONEOUS "IAE" ERROR DURING RECALIBRATE
172			
002474	070730	EMT70	
002476	074140	EHT1	
002500	074264	EDT1	
002502	074354	EFT1	
173			
174		:ERROR 71	"ILF" ERROR DURING RECALIBRATE
175			
002504	070740	EMT71	
002506	074140	EHT1	
002510	074264	EDT1	
002512	074354	EFT1	
176			
177		:ERROR 72	"DPI" ERROR DURING RECALIBRATE DUE TO "MOL" = 0
178			
002514	070750	EMT72	
002516	074140	EHT1	
002520	074264	EDT1	
002522	074354	EFT1	
179			
180		:ERROR 73	"DPI" ERROR DURING RECALIBRATE BECAUSE ON
181		:	CYLINDER DIDNT DROP
182			
002524	070766	EMT73	
002526	074140	EHT1	
002530	074264	EDT1	
002532	074354	EFT1	
183			
184		:ERROR 74	"IVC" ERROR DURING RECALIBRATE - "VV" = 0
185			
002534	071004	EMT74	
002536	074140	EHT1	
002540	074264	EDT1	
002542	074354	EFT1	
186			
187		:ERROR 75	ERRONEOUS "IVC" ERROR DURING RECALIBRATE - "VV" = 1
188			
002544	071014	EMT75	
002546	074140	EHT1	
002550	074264	EDT1	
002552	074354	EFT1	
189			
190		:ERROR 76	"SKI" ERROR DURING RECALIBRATE
191			
002554	071034	EMT76	
002556	074140	EHT1	

	002560	074264	EDT1	
	002562	074354	EFT1	
192				
193				:ERROR 77 'DVC' OCCURRED DURING RECALIBRATE
194				
	002564	071044	EMT77	
	002566	074140	EHT1	
	002570	074264	EDT1	
	002572	074354	EFT1	
195				
196				:ERROR 100 LOST 'MOL' DURING RECALIBRATE - 'OPI' = 0
197				
	002574	071056	EMT100	
	002576	074140	EHT1	
	002600	074264	EDT1	
	002602	074354	EFT1	
198				
199				:ERROR 101 LOST 'VV' DURING RECALIBRATE - 'IVC' = 0
200				
	002604	071074	EMT101	
	002606	074140	EHT1	
	002610	074264	EDT1	
	002612	074354	EFT1	
201				
202				:ERROR 102 'ATA' DID NOT SET DURING RECALIBRATE
203				
	002614	071112	EMT102	
	002616	074140	EHT1	
	002620	074264	EDT1	
	002622	074354	EFT1	
204				
205				:ERROR 103 'OM' DID NOT RESET DURING RECALIBRATE
206				
	002624	071122	EMT103	
	002626	074140	EHT1	
	002630	074264	EDT1	
	002632	074354	EFT1	
207				
208				:ERROR 104 'PIP' IS STIL SET AFTER RECALIBRATE
209				
	002634	071134	EMT104	
	002636	074140	EHT1	
	002640	074264	EDT1	
	002642	074354	EFT1	
210				
211				:ERROR 105 UNEXPECTED 'ILR' ERROR DURING RECALIBRATE
212				
	002644	071152	EMT105	
	002646	074140	EHT1	
	002650	074264	EDT1	

	002652	074354	EFT1	
213				
214			:ERROR 106	UNEXPECTED 'RMR' ERROR DURING RECALIBRATE
215				
	002654	071162	EMT106	
	002656	074140	EHT1	
	002660	074264	EDT1	
	002662	074354	EFT1	
216				
217			:ERROR 107	'UNS' ERROR DURING RECALIBRATE - AC POWER IS LOW
218				
	002664	071172	EMT107	
	002666	074140	EHT1	
	002670	074264	EDT1	
	002672	074354	EFT1	
219				
220			:ERROR 110	CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
221				
	002674	071212	EMT110	
	002676	074162	EHT110	
	002700	074302	EDT110	
	002702	074372	EFT110	
222				
223			:ERROR 111	NONEXISTENT DEVICE
224				
	002704	071224	EMT111	
	002706	074166	EHT111	
	002710	074304	EDT111	
	002712	074374	EFT111	
225				
226			:ERROR 112	DEVICE NOT AVAILABLE
227				
	002714	071232	EMT112	
	002716	074166	EHT111	
	002720	074304	EDT111	
	002722	074374	EFT111	
228				
229			:ERROR 113	BUS TIMEOUT-NED STATUS FAILURE
230				
	002724	071240	EMT113	
	002726	000000	0	
	002730	000000	0	
	002732	000000	0	
231				
232			:ERROR 114	UNUSED
233				
	002734	000000	0	
	002736	000000	0	
	002740	000000	0	
	002742	000000	0	

234				
235			:ERROR 115	RMCS1 NOT INITIALIZED BY UNIBUS
236	002744	071256		
	002746	074140	EMT115	
	002750	074264	EHT1	
	002752	074354	EDT1	
			EFT1	
237				
238			:ERROR 116	RMBA NOT INITIALIZED BY UNIBUS
239	002754	071266		
	002756	074140	EMT116	
	002760	074264	EHT1	
	002762	074354	EDT1	
			EFT1	
240				
241			:ERROR 117	RMCS2 NOT INITIALIZED BY UNIBUS
242	002764	071276		
	002766	074140	EMT117	
	002770	074264	EHT1	
	002772	074354	EDT1	
			EFT1	
243				
244			:ERROR 120	RMER1 NOT INITIALIZED BY UNIBUS
245	002774	071306		
	002776	074140	EMT120	
	003000	074264	EHT1	
	003002	074354	EDT1	
			EFT1	
246				
247			:ERROR 121	RMAS NOT INITIALIZED BY UNIBUS
248	003004	071316		
	003006	074140	EMT121	
	003010	074264	EHT1	
	003012	074354	EDT1	
			EFT1	
249				
250			:ERROR 122	RMMR1 NOT INITIALIZED BY UNIBUS
251	003014	071326		
	003016	074140	EMT122	
	003020	074264	EHT1	
	003022	074354	EDT1	
			EFT1	
252				
253			:ERROR 123	RMDS NOT INITIALIZED BY UNIBUS
254	003024	071336		
	003026	074140	EMT123	
	003030	074264	EHT1	
	003032	074354	EDT1	
			EFT1	

255				
256			:ERROR	124 RMEC2 NOT INITIALIZED BY UNIBUS
257	003034	071346		EMT124
	003036	074140		EHT1
	003040	074264		EDT1
	003042	074354		EFT1
258				
259			:ERROR	125 RMMR2 NOT INITIALIZED BY UNIBUS
260	003044	071356		EMT125
	003046	074140		EHT1
	003050	074264		EDT1
	003052	074354		EFT1
261				
262			:ERROR	126 RMCS1 NOT CLEARED BY CONTROLLER CLEAR
263	003054	071366		EMT126
	003056	074140		EHT1
	003060	074264		EDT1
	003062	074354		EFT1
264				
265			:ERROR	127 RMBA NOT CLEARED BY CONTROLLER CLEAR
266	003064	071400		EMT127
	003066	074140		EHT1
	003070	074264		EDT1
	003072	074354		EFT1
267				
268			:ERROR	130 RMCS2 NOT CLEARED BY CONTROLLER CLEAR
269	003074	071412		EMT130
	003076	074140		EHT1
	003100	074264		EDT1
	003102	074354		EFT1
270				
271			:ERROR	131 RMER1 NOT CLEARED BY CONTROLLER CLEAR
272	003104	071424		EMT131
	003106	074140		EHT1
	003110	074264		EDT1
	003112	074354		EFT1
273				
274			:ERROR	132 RMAS NOT CLEARED BY CONTROLLER CLEAR
275	003114	071436		EMT132
	003116	074140		EHT1
	003120	074264		EDT1
	003122	074354		EFT1
276				

277		:ERROR 133	RMMR1 NOT CLEARED BY CONTROLLER CLEAR
278	003124 071450	EMT133	
	003126 074140	EHT1	
	003130 074264	EDT1	
	003132 074354	EFT1	
279		:ERROR 134	RMDS NOT CLEARED BY CONTROLLER CLEAR
280			
281	003134 071462	EMT134	
	003136 074140	EHT1	
	003140 074264	EDT1	
	003142 074354	EFT1	
282		:ERROR 135	RMEC2 NOT CLEARED BY CONTROLLER CLEAR
283			
284	003144 071474	EMT135	
	003146 074140	EHT1	
	003150 074264	EDT1	
	003152 074354	EFT1	
285		:ERROR 136	RMMR2 NOT CLEARED BY CONTROLLER CLEAR
286			
287	003154 071506	EMT136	
	003156 074140	EHT1	
	003160 074264	EDT1	
	003162 074354	EFT1	
288		:ERROR 137	RMCS1 NOT CLEARED BY ERROR CLEAR
289			
290	003164 071520	EMT137	
	003166 074140	EHT1	
	003170 074264	EDT1	
	003172 074354	EFT1	
291		:ERROR 140	RMCS2 NOT CLEARED BY ERROR CLEAR
292			
293	003174 071530	EMT140	
	003176 074140	EHT1	
	003200 074264	EDT1	
	003202 074354	EFT1	
294		:ERROR 141	RMCS1 NOT CLEARED BY DRIVE CLEAR
295			
296	003204 071540	EMT141	
	003206 074140	EHT1	
	003210 074264	EDT1	
	003212 074354	EFT1	
297		:ERROR 142	RMDS NOT CLEARED BY DRIVE CLEAR
298			

299	003214 071550	EMT142	
	003216 074140	EHT1	
	003220 074264	EDT1	
	003222 074354	EFT1	
300			
301		:ERROR 143	RMER1 NOT CLEARED BY DRIVE CLEAR
302			
	003224 071560	EMT143	
	003226 074140	EHT1	
	003230 074264	EDT1	
	003232 074354	EFT1	
303			
304		:ERROR 144	RMAS NOT CLEARED BY DRIVE CLEAR
305			
	003234 071570	EMT144	
	003236 074140	EHT1	
	003240 074264	EDT1	
	003242 074354	EFT1	
306			
307		:ERROR 145	RMMR1 NOT CLEARED BY DRIVE CLEAR
308			
	003244 071600	EMT145	
	003246 074140	EHT1	
	003250 074264	EDT1	
	003252 074354	EFT1	
309			
310		:ERROR 146	RMMR2 NOT CLEARED BY DRIVE CLEAR
311			
	003254 071610	EMT146	
	003256 074140	EHT1	
	003260 074264	EDT1	
	003262 074354	EFT1	
312			
313		:ERROR 147	RMER2 NOT CLEARED BY DRIVE CLEAR
314			
	003264 071620	EMT147	
	003266 074140	EHT1	
	003270 074264	EDT1	
	003272 074354	EFT1	
315			
316		:ERROR 150	RMEC2 NOT CLEARED BY DRIVE CLEAR
317			
	003274 071630	EMT150	
	003276 074140	EHT1	
	003300 074264	EDT1	
	003302 074354	EFT1	
318			
319		:ERROR 151	MEDIUM NOT ON LINE
320			

	003304	071640		EMT151
	003306	074140		EHT1
	003310	074264		EDT1
	003312	074354		EFT1
321				
322			:ERROR 152	DRIVE FAULT
323				
	003314	071652		EMT152
	003316	074140		EHT1
	003320	074264		EDT1
	003322	074354		EFT1
324				
325			:ERROR 153	UNSAFE SHOULD BE SET BECAUSE DVC IS SET
326				
	003324	071664		EMT153
	003326	074140		EHT1
	003330	074264		EDT1
	003332	074354		EFT1
327				
328			:ERROR 154	UNSAFE SHOULD NOT BE SET, AC IS LOW
329				
	003334	071702		EMT154
	003336	074140		EHT1
	003340	074264		EDT1
	003342	074354		EFT1
330				
331			:ERROR 155	VOLUME VALID NOT SET BY PACK ACK
332				
	003344	071720		EMT155
	003346	074140		EHT1
	003350	074264		EDT1
	003352	074354		EFT1
333				
334			:ERROR 156	OFFSET MODE NOT SET BY OFFSET COMMAND
335				
	003354	071732		EMT156
	003356	074140		EHT1
	003360	074264		EDT1
	003362	074354		EFT1
336				
337			:ERROR 157	OFFSET MODE NOT RESET BY RTC COMMAND
338				
	003364	071744		EMT157
	003366	074140		EHT1
	003370	074264		EDT1
	003372	074354		EFT1
339				
340			:ERROR 160	RMOF NOT RESET BY RIP COMMAND
341				
	003374	071756		EMT160

	003376	074140	EHT1	
	003400	074264	EDT1	
	003402	074354	EFT1	
342				
343				:ERROR 161 RMDA NOT RESET BY RIP COMMAND
344				
	003404	071766	EMT161	
	003406	074140	EHT1	
	003410	074264	EDT1	
	003412	074354	EFT1	
345				
346				:ERROR 162 RMDC NOT RESET BY RIP COMMAND
347				
	003414	072000	EMT162	
	003416	074140	EHT1	
	003420	074264	EDT1	
	003422	074354	EFT1	
348				
349				:ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
350				WRITE BUFFER
351				
	003424	073734	EMT336	
	003426	074222	EHT336	
	003430	074320	EDT336	
	003432	074410	EFT336	
352				
353				:ERROR 164 OPI SHOULD NOT BE SET
354				
	003434	072022	EMT164	
	003436	074140	EHT1	
	003440	074264	EDT1	
	003442	074354	EFT1	
355				
356				:ERROR 165 IVC SHOULD NOT BE SET
357				
	003444	072030	EMT165	
	003446	074140	EHT1	
	003450	074264	EDT1	
	003452	074354	EFT1	
358				
359				:ERROR 166 IAE SHOULD NOT BE SET
360				
	003454	072036	EMT166	
	003456	074140	EHT1	
	003460	074264	EDT1	
	003462	074354	EFT1	
361				
362				:ERROR 167 NEM SHOULD NOT BE SET
363				
	003464	072044	EMT167	

ERROR POINTER TABLE

	003466	074140		EHT1
	003470	074264		EDT1
	003472	074354		EFT1
364				
365			:ERROR 170	INCORRECT 'MOL' STATUS DURING DIAGNOSTIC MODE
366				
	003474	072052		EMT170
	003476	074140		EHT1
	003500	074264		EDT1
	003502	074354		EFT1
367				
368			:ERROR 171	'ATA' NOT SET DURING RETURN TO CENTERLINE
369				
	003504	072064		EMT171
	003506	074140		EHT1
	003510	074264		EDT1
	003512	074354		EFT1
370				
371			:ERROR 172	'ATA' NOT SET BY OFFSET COMMAND
372				
	003514	072074		EMT172
	003516	074140		EHT1
	003520	074264		EDT1
	003522	074354		EFT1
373				
374			:ERROR 173	RMER2 NOT INITIALIZED BY UNIBUS INIT
375				
	003524	072104		EMT173
	003526	074140		EHT1
	003530	074264		EDT1
	003532	074354		EFT1
376				
377			:ERROR 174	RMER2 NOT INITIALIZED BY CONTROLLER CLEAR
378				
	003534	072114		EMT174
	003536	074140		EHT1
	003540	074264		EDT1
	003542	074354		EFT1
379				
380			:ERROR 175	SELECTED DEVICE IS IN WRITE PROTECT
381				
	003544	072126		EMT175
	003546	074140		EHT1
	003550	074264		EDT1
	003552	074354		EFT1
382				
383			:ERROR 176	CANNOT SET DIAGNOSTIC MODE
384				
	003554	072134		EMT176
	003556	074140		EHT1

	003550	074264	EDT1	
	003562	074354	EFT1	
385				
386				:ERROR 177 --RESERVED FOR POWER MONITOR BIT FAILURE--
387				
	003564	000000	0	
	003566	000000	0	
	003570	000000	0	
	003572	000000	0	
388				
389				:ERROR 200 INCORRECT 'PIP' STATUS DURING DIAGNOSTIC MODE
390				
	003574	072144	EMT200	
	003576	074140	EHT1	
	003600	074264	EDT1	
	003602	074354	EFT1	
391				
392				:ERROR 201 INCORRECT 'WRL' STATUS DURING DIAGNOSTIC MODE
393				
	003604	072156	EMT201	
	003606	074140	EHT1	
	003610	074264	EDT1	
	003612	074354	EFT1	
394				
395				:ERROR 202 INCORRECT 'SKI' STATUS DURING DIAGNOSTIC MODE
396				
	003614	072170	EMT202	
	003616	074140	EHT1	
	003620	074264	EDT1	
	003622	074354	EFT1	
397				
398				:ERROR 203 INCORRECT 'DVC' STATUS DURING DIAGNOSTIC MODE
399				
	003624	072202	EMT203	
	003626	074140	EHT1	
	003630	074264	EDT1	
	003632	074354	EFT1	
400				
401				:ERROR 204 'VV' WAS NOT RESET BY MAINTENANCE UNIT READY
402				
	003634	072214	EMT204	
	003636	074140	EHT1	
	003640	074264	EDT1	
	003642	074354	EFT1	
403				
404				:ERROR 205 SELECTED DEVICE HAS A PERSISTENT 'SKI' ERROR
405				
	003644	072232	EMT205	
	003646	074140	EHT1	
	003650	074264	EDT1	

	003652	074354	EFT1	
406				
407				
408				:ERROR 206 'LBC' DID NOT SET DURING DIAGNOSTIC MODE
	003654	072242	EMT206	
	003656	074140	EHT1	
	003660	074264	EDT1	
	003662	074354	EFT1	
409				
410				:ERROR 207 UNEXPECTED LOSS OF 'MOL' - MEDIUM IS OFF LINE
411				
	003664	072252	EMT207	
	003666	074140	EHT1	
	003670	074264	EDT1	
	003672	074354	EFT1	
412				
413				:ERROR 210 UNEXPECTED LOSS OF VOLUME VALID - 'VV' = 0
414				
	003674	072264	EMT210	
	003676	074140	EHT1	
	003700	074264	EDT1	
	003702	074354	EFT1	
415				
416				:ERROR 211 UNEXPECTED MECHANICAL MOTION - 'PIP' = 1
417				
	003704	072272	EMT211	
	003706	074140	EHT1	
	003710	074264	EDT1	
	003712	074354	EFT1	
418				
419				:ERROR 212 UNEXPECTED DEVICE FAULT - 'DVC' = 1
420				
	003714	072306	EMT212	
	003716	074140	EHT1	
	003720	074264	EDT1	
	003722	074354	EFT1	
421				
422				:ERROR 213 UNEXPECTED SEEK INCOMPLETE ERROR - 'SKI' = 1
423				
	003724	072322	EMT213	
	003726	074140	EHT1	
	003730	074264	EDT1	
	003732	074354	EFT1	
424				
425				:ERROR 214 DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
426				
	003734	072332	EMT214	
	003736	074152	EHT2	
	003740	074274	EDT2	
	003742	074364	EFT2	

427				
428			:ERROR 215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
429	003744	072352	EMT215	
	003746	074152	EHT2	
	003750	074274	EDT2	
	003752	074364	EFT2	
430				
431			:ERROR 216	INCORRECT "IVC" STATUS
432	003754	072364	EMT216	
	003756	074140	EHT1	
	003760	074264	EDT1	
	003762	074354	EFT1	
433				
434			:ERROR 217	INCORRECT "IAE" STATUS
435	003764	072374	EMT217	
	003766	074140	EHT1	
	003770	074264	EDT1	
	003772	074354	EFT1	
436				
437			:ERROR 220	INCORRECT "WLE" STATUS
438	003774	072404	EMT220	
	003776	074140	EHT1	
	004000	074264	EDT1	
	004002	074354	EFT1	
439				
440			:ERROR 221	INCORRECT "OPI" STATUS
441	004004	072414	EMT221	
	004006	074140	EHT1	
	004010	074264	EDT1	
	004012	074354	EFT1	
442				
443			:ERROR 222	RM80 DID NOT DETECT RMR ERROR
444	004014	072424	EMT222	
	004016	074140	EHT1	
	004020	074264	EDT1	
	004022	074354	EFT1	
445				
446			:ERROR 223	RM80 DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
447	004024	072434	EMT223	
	004026	074176	EHT223	
	004030	074310	EDT223	
	004032	074400	EFT223	

ERROR POINTER TABLE

448				
449			:ERROR 224	SSE NOT DETECTED DURING DATA TRANSFER
450	004034	072444		EMT224
	004036	074140		EHT1
	004040	074264		EDT1
	004042	074354		EFT1
451				
452			:ERROR 225	"SSEI" NOT RESET BY END OF TRACK DURING DATA TRANSFER
453	004044	072460		EMT225
	004046	074140		EHT1
	004050	074264		EDT1
	004052	074354		EFT1
454				
455			:ERROR 226	"SSEI" SHOULD BE SET AFTER DATA TRANSFER
456	004054	072500		EMT226
	004056	074140		EHT1
	004060	074264		EDT1
	004062	074354		EFT1
457				
458			:ERROR 227	SSE WAS DETECTED W/ SSEI SET
459	004064	072510		EMT227
	004066	074140		EHT1
	004070	074264		EDT1
	004072	074354		EFT1
460				
461			:ERROR 230	UNUSED
462	004074	072524		EMT230
	004076	000000		0
	004100	000000		0
	004102	000000		0
463				
464			:ERROR 231	UNUSED
465	004104	072526		EMT231
	004106	000000		0
	004110	000000		0
	004112	000000		0
466				
467			:ERROR 232	UNUSED
468	004114	072530		EMT232
	004116	000000		0
	004120	000000		0
	004122	000000		0

469

470			:ERROR	233	UNUSED
471	004124	072532		EMT233	
	004126	000000		0	
	004130	000000		0	
	004132	000000		0	
472			:ERROR	234	UNUSED
473					
474	004134	000000		0	
	004136	000000		0	
	004140	000000		0	
	004142	000000		0	
475			:ERROR	235	UNUSED
476					
477	004144	000000		0	
	004146	000000		0	
	004150	000000		0	
	004152	000000		0	
478			:ERROR	236	UNUSED
479					
480	004154	000000		0	
	004156	000000		0	
	004160	000000		0	
	004162	000000		0	
481			:ERROR	237	UNUSED
482					
483	004164	000000		0	
	004166	000000		0	
	004170	000000		0	
	004172	000000		0	
484			:ERROR	240	UNUSED
485					
486	004174	000000		0	
	004176	000000		0	
	004200	000000		0	
	004202	000000		0	
487			:ERROR	241	UNUSED
488					
489	004204	000000		0	
	004206	000000		0	
	004210	000000		0	
	004212	000000		0	
490			:ERROR	242	UNUSED
491					

492	004214	000000	0	
	004216	000000	0	
	004220	000000	0	
	004222	000000	0	
493				
494			:ERROR 243	UNUSED
495	004224	000000	0	
	004226	000000	0	
	004230	000000	0	
	004232	000000	0	
496				
497			:ERROR 244	UNUSED
498	004234	000000	0	
	004236	000000	0	
	004240	000000	0	
	004242	000000	0	
499				
500			:ERROR 245	UNUSED
501	004244	000000	0	
	004246	000000	0	
	004250	000000	0	
	004252	000000	0	
502				
503			:ERROR 246	'ATA' NOT RESET BY GO WHEN 'ERR' = 0
504	004254	072560	EMT246	
	004256	074140	EHT1	
	004260	074264	EDT1	
	004262	074354	EFT1	
505				
506			:ERROR 247	'ATA' NOT RESET BY WRITING RMAS
507	004264	072570	EMT247	
	004266	074140	EHT1	
	004270	074264	EDT1	
	004272	074354	EFT1	
508				
509			:ERROR 250	'ATA' WAS RESET BY GO WHEN 'ERR' = 1
510	004274	072602	EMT250	
	004276	074140	EHT1	
	004300	074264	EDT1	
	004302	074354	EFT1	
511				
512			:ERROR 251	PROGRAM INTERRUPT WAS NOT GENERATED
513				

ERROR POINTER TABLE

	004304	072616		EMT251	
	004306	074152		EHT2	
	004310	074274		EDT2	
	004312	074364		EFT2	
514					
515			:ERROR	252	PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
516					
	004314	072624		EMT252	
	004316	074152		EHT2	
	004320	074274		EDT2	
	004322	074364		EFT2	
517					
518			:ERROR	253	OFFSET MODE WAS NOT RESET BY WRITING RMDC
519					
	004324	072632		EMT253	
	004326	074140		EHT1	
	004330	074264		EDT1	
	004332	074354		EFT1	
520					
521			:ERROR	254	INCORRECT "ILF" STATUS
522					
	004334	072650		EMT254	
	004336	074140		EHT1	
	004340	074264		EDT1	
	004342	074354		EFT1	
523					
524			:ERROR	255	INCORRECT "ATA" STATUS
525					
	004344	072660		EMT255	
	004346	074140		EHT1	
	004350	074264		EDT1	
	004352	074354		EFT1	
526					
527			:ERROR	256	INCORRECT "ILR" STATUS
528					
	004354	072670		EMT256	
	004356	074210		EHT256	
	004360	074310		EDT223	
	004362	074400		EFT223	
529					
530			:ERROR	257	INVALID IAE STATUS DURING SEARCH COMMAND
531					
	004364	072700		EMT257	
	004366	074140		EHT1	
	004370	074264		EDT1	
	004372	074354		EFT1	
532					
533			:ERROR	260	"IVC" WAS NOT DETECTED DURING SEARCH COMMAND
534					
	004374	072712		EMT260	

	004376	074140		EHT1
	004400	074264		EDT1
	004402	074354		EFT1
535				
536			:ERROR 261	DRIVE EXECUTED SEARCH WITH ERROR SET
537				
	004404	072724		EMT261
	004406	074140		EHT1
	004410	074264		EDT1
	004412	074354		EFT1
538				
539			:ERROR 262	'LBC' ERROR NOT SET DURING DIAGNOSTIC MODE
540				
	004414	072744		EMT262
	004416	074140		EHT1
	004420	074264		EDT1
	004422	074354		EFT1
541				
542			:ERROR 263	'SKI' ERROR DURING SEARCH COMMAND
543				
	004424	072754		EMT263
	004426	074140		EHT1
	004430	074264		EDT1
	004432	074354		EFT1
544				
545			:ERROR 264	'IVC' ERROR DURING SEARCH - LOST VOLUME VALID
546				
	004434	072764		EMT264
	004436	074140		EHT1
	004440	074264		EDT1
	004442	074354		EFT1
547				
548			:ERROR 265	ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
549				
	004444	073004		EMT265
	004446	074140		EHT1
	004450	074264		EDT1
	004452	074354		EFT1
550				
551			:ERROR 266	DEVICE FAULT (DVC) DURING SEARCH
552				
	004454	073024		EMT266
	004456	074140		EHT1
	004460	074264		EDT1
	004462	074354		EFT1
553				
554			:ERROR 267	SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER
555			:	ADDRESS IS TOO LARGE
556				
	004464	073036		EMT267

ERROR POINTER TABLE

	004466	074140		EHT1
	004470	074264		EDT1
	004472	074354		EFT1
557				
558			:ERROR 270	OPI ERROR DURING SEARCH BECAUSE MOL = 0
559				
	004474	073054		EMT270
	004476	074140		EHT1
	004500	074264		EDT1
	004502	074354		EFT1
560				
561			:ERROR 271	OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
562			:	DIDN'T DROP
563				
	004504	073070		EMT271
	004506	074140		EHT1
	004510	074264		EDT1
	004512	074354		EFT1
564				
565			:ERROR 272	LOST MOL DURING SEARCH, OPI IS NOT SET
566				
	004514	073106		EMT272
	004516	074140		EHT1
	004520	074264		EDT1
	004522	074354		EFT1
567				
568			:ERROR 273	PIP STIL SET AFTER SEARCH
569				
	004524	073124		EMT273
	004526	074140		EHT1
	004530	074264		EDT1
	004532	074354		EFT1
570				
571			:ERROR 274	PARITY ERROR OCCURRED WHILE WRITING REMOTE
572			:	REGISTERS BUT MXF DID NOT SET
573				
	004534	073142		EMT274
	004536	074140		EHT1
	004540	074264		EDT1
	004542	074354		EFT1
574				
575			:ERROR 275	MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
576			:	COMMAND STARTED
577				
	004544	073160		EMT275
	004546	074140		EHT1
	004550	074264		EDT1
	004552	074354		EFT1
578				
579			:ERROR 276	'OPI' ERROR DURING DATA TRANSFER BECAUSE 'MOL' WAS

580		:	ZERO
581			
	004554	073172	EMT276
	004556	074140	EHT1
	004560	074264	EDT1
	004562	074354	EFT1
582			
583		:ERROR	277 'OPI' ERROR DURING DATA TRANSFER BECAUSE 1) ON
584		:	CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
585		:	3) RUN TIMED OUT
586			
	004564	073206	EMT277
	004566	074140	EHT1
	004570	074264	EDT1
	004572	074354	EFT1
587			
588		:ERROR	300 'IVC' ERROR DURING DATA TRANSFER BECAUSE VOLUME
589		:	WAS NOT VALID
590			
	004574	073224	EMT300
	004576	074140	EHT1
	004600	074264	EDT1
	004602	074354	EFT1
591			
592		:ERROR	301 ERRONEOUS 'IVC' ERROR DURING DATA TRANSFER - VOLUME
593		:	IS VALID
594			
	004604	073244	EMT301
	004606	074140	EHT1
	004610	074264	EDT1
	004612	074354	EFT1
595			
596		:ERROR	302 'ILR' ERROR DURING DATA TRANSFER
597			
	004614	073266	EMT302
	004616	074140	EHT1
	004620	074264	EDT1
	004622	074354	EFT1
598			
599		:ERROR	303 'ILF' ERROR DURING DATA TRANSFER
600			
	004624	073300	EMT303
	004626	074140	EHT1
	004630	074264	EDT1
	004632	074354	EFT1
601			
602		:ERROR	304 'RMR' ERROR DURING DATA TRANSFER
603			
	004634	073312	EMT304
	004636	074140	EHT1
	004640	074264	EDT1

ERROR POINTER TABLE

	004642	074354	EFT1	
604				
605				
606				:ERROR 305 INCORRECT "IAE" STATUS DURING DATA TRANSFER
	004644	073324	EMT305	
	004646	074140	EHT1	
	004650	074264	EDT1	
	004652	074354	EFT1	
607				
608				
609				:ERROR 306 "SKI" ERROR DURING DATA TRANSFER
	004654	073336	EMT306	
	004656	074140	EHT1	
	004660	074264	EDT1	
	004662	074354	EFT1	
610				
611				
612				:ERROR 307 DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
	004664	073346	EMT307	
	004666	074140	EHT1	
	004670	074264	EDT1	
	004672	074354	EFT1	
613				
614				
615				:ERROR 310 DEVICE FAULT DURING DATA TRANSFER
	004674	073366	EMT310	
	004676	074140	EHT1	
	004700	074264	EDT1	
	004702	074354	EFT1	
616				
617				
618				:ERROR 311 LOSS OF BIT CLOCK DURING DATA TRANSFER
	004704	073400	EMT311	
	004706	074140	EHT1	
	004710	074264	EDT1	
	004712	074354	EFT1	
619				
620				
621				:ERROR 312 LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
	004714	073412	EMT312	
	004716	074140	EHT1	
	004720	074264	EDT1	
	004722	074354	EFT1	
622				
623				
624				:ERROR 313 UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
	004724	073424	EMT313	
	004726	074140	EHT1	
	004730	074264	EDT1	
	004732	074354	EFT1	

625				
626			:ERROR 314	DRIVE TIMING ERROR DURING DATA TRANSFER
627	004734	073444		EMT314
	004736	074140		EHT1
	004740	074264		EDT1
	004742	074354		EFT1
628				
629			:ERROR 315	WRITE LOCK ERROR
630	004744	073456		EMT315
	004746	074140		EHT1
	004750	074264		EDT1
	004752	074354		EFT1
631				
632			:ERROR 316	ERRONEOUS WRITE LOCK ERROR
633	004754	073470		EMT316
	004756	074140		EHT1
	004760	074264		EDT1
	004762	074354		EFT1
634				
635			:ERROR 317	HEADER CRC ERROR DURING DATA TRANSFER
636	004764	073502		EMT317
	004766	074140		EHT1
	004770	074264		EDT1
	004772	074354		EFT1
637				
638			:ERROR 320	FORMAT ERROR DURING DATA TRANSFER
639	004774	073512		EMT320
	004776	074140		EHT1
	005000	074264		EDT1
	005002	074354		EFT1
640				
641			:ERROR 321	HEADER COMPARE ERROR DURING DATA TRANSFER
642	005004	073522		EMT321
	005006	074140		EHT1
	005010	074264		EDT1
	005012	074354		EFT1
643				
644			:ERROR 322	HEADER ERRORS SHOULD NOT BE SET
645	005014	073532		EMT322
	005016	074140		EHT1
	005020	074264		EDT1
	005022	074354		EFT1

646				
647			:ERROR 323	DATA CHECK ERROR DURING DATA TRANSFER
648				
	005024	073540		EMT323
	005026	074140		EHT1
	005030	074264		EDT1
	005032	074354		EFT1
649				
650			:ERROR 324	CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
651				
	005034	073550		EMT324
	005036	074140		EHT1
	005040	074264		EDT1
	005042	074354		EFT1
652				
653			:ERROR 325	UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
654				
	005044	073562		EMT325
	005046	074140		EHT1
	005050	074264		EDT1
	005052	074354		EFT1
655				
656			:ERROR 326	DATA PARITY ERROR DURING READ COMMAND
657				
	005054	073574		EMT326
	005056	074140		EHT1
	005060	074264		EDT1
	005062	074354		EFT1
658				
659			:ERROR 327	OFFSET MODE NOT RESET BY WRITE COMMAND
660				
	005064	073612		EMT327
	005066	074140		EHT1
	005070	074264		EDT1
	005072	074354		EFT1
661				
662			:ERROR 330	DATA PARITY ERROR DURING WRITE COMMAND
663				
	005074	073624		EMT330
	005076	074140		EHT1
	005100	074264		EDT1
	005102	074354		EFT1
664				
665			:ERROR 331	WRITE CLOCK FAILURE DURING WRITE COMMAND
666				
	005104	073634		EMT331
	005106	074140		EHT1
	005110	074264		EDT1
	005112	074354		EFT1
667				

ERROR POINTER TABLE

668		:ERROR 332	DATA LATE ERROR DURING DATA TRANSFER
669			
	005114 073646		EMT332
	005116 074140		EHT1
	005120 074264		EDT1
	005122 074354		EFT1
670		:ERROR 333	PIP STIL SET AFTER DATA TRANSFER - SKI = 0
671			
672			
	005124 073660		EMT333
	005126 074140		EHT1
	005130 074264		EDT1
	005132 074354		EFT1
673		:ERROR 334	LOST MOL DURING DATA TRANSFER - OPI = 0
674			
675			
	005134 073676		EMT334
	005136 074140		EHT1
	005140 074264		EDT1
	005142 074354		EFT1
676		:ERROR 335	LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
677			
678			
	005144 073714		EMT335
	005146 074140		EHT1
	005150 074264		EDT1
	005152 074354		EFT1
679		:ERROR 336	DATA READ DOES NOT COMPARE WITH DATA WRITTEN
680			
681			
	005154 073734		EMT336
	005156 074222		EHT336
	005160 074320		EDT336
	005162 074410		EFT336
682		:ERROR 337	WRITE CHECK ERROR NOT DETECTED
683			
684			
	005164 073744		EMT337
	005166 074234		EHT337
	005170 074330		EDT337
	005172 074420		EFT337
685		:ERROR 340	WRITE CHECK ERROR AT UNEXPECTED ADDRESS
686			
687			
	005174 073754		EMT340
	005176 074222		EHT336
	005200 074320		EDT336
	005202 074410		EFT336
688		:ERROR 341	INCORRECT DATA DURING WRITE CHECK ERROR
689			

ERROR POINTER TABLE

690	005204	073766	EMT341	
	005206	074222	EHT336	
	005210	074320	EDT336	
	005212	074410	EFT336	
691				
692			:ERROR	342 "IVC" ERROR NOT DETECTED DURING DATA TRANSFER
693	005214	073774	EMT342	
	005216	074140	EHT1	
	005220	074264	EDT1	
	005222	074354	EFT1	
694				
695			:ERROR	343 "FER" NOT DETECTED DURING DATA TRANSFER
696	005224	074006	EMT343	
	005226	074140	EHT1	
	005230	074264	EDT1	
	005232	074354	EFT1	
697				
698			:ERROR	344 "HCE" NOT DETECTED DURING DATA TRANSFER
699	005234	074020	EMT344	
	005236	074246	EHT344	
	005240	074340	EDT344	
	005242	074430	EFT344	
700				
701			:ERROR	345 "BSE" NOT DETECTED DURING DATA TRANSFER
702	005244	074032	EMT345	
	005246	074140	EHT1	
	005250	074264	EDT1	
	005252	074354	EFT1	
703				
704			:ERROR	346 HEADER ERROR WAS DETECTED W/ HCI SET
705	005254	074042	EMT346	
	005256	074140	EHT1	
	005260	074264	EDT1	
	005262	074354	EFT1	
706				
707			:ERROR	347 DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
708	005264	074056	EMT347	
	005266	074140	EHT1	
	005270	074264	EDT1	
	005272	074354	EFT1	
709				
710			:ERROR	350 LOST VOLUME VALID DURING SEARCH - "IVC" = 0
711				

ERROR POINTER TABLE

005274	074070	EMT350
005276	074140	EHT1
005300	074264	EDT1
005302	074354	EFT1

712
713
714

```

;ERROR 351 "ATA" DID NOT SET DURING SEARCH

```

005304	074106	EMT351
005306	074140	EHT1
005310	074264	EDT1
005312	074354	EFT1

715
716
717

```

;ERROR 352 PROGRAM TIMEOUT WHILE TESTING RMLA

```

005314	074116	EMT352
005316	000000	0
005320	000000	0
005322	000000	0

718
719
720

```

;ERROR 353 LOOK AHEAD TEST FAILS

```

005324	074122	EMT353
005326	074260	EHT353
005330	074352	EDT353
005332	074440	EFT353

721
722
723

```

;ERROR 354 BSE SHOULD NOT BE SET

```

005334	074132	EMT354
005336	074140	EHT1
005340	074264	EDT1
005342	074354	EFT1

724
725

```

;PUT ERROR TABLE HERE

```


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

.SBTTL ERROR TABLE USAGE

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

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

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

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

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

```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3      005344 011600      BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
4      005346 005740      TST      -(R0)      ;ADJUST PC -2
5      005350 022626      CMP      (SP)+,(SP)+  ;RESTORE STACK POINTER
6      005352 104401 005360  TYPE      65$      ;:TYPE ASCIZ STRING
        005356 000417      BR       64$      ;:GET OVER THE ASCIZ
        ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
        64$:
7      005416 010046      MOV      R0,-(SP)    ;SETUP FOR TYPING OUT PC
8      005420 104402      TYPOC
9      005422 000240      NOP
        ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
        ;TO STOP ON UNEXPECTED TIMEOUT.
10
11
12      .SBTTL  START OF PROGRAM
13
14      005424 012737 177777 001330  START1: MOV      #-1,CHGADR  ;CHANGE RH/RM BUS ADDRESS
15      005432 000402      BR       START2
16
17      005434 005037 001330  START:  CLR      CHGADR  ;NO CHANGE IN ADDRESS
18      005440 000240  START2:  NOP
19      005442 005227 000000  INC      #0          ;TTY LOOP, WAIT FOR INCREMENT
20      005446 001375      BNE     #-4          ;OF WORD
21      005450 000005      RESET
        ;RESET THE WORLD
22
23      .SBTTL  INITIALIZE THE COMMON TAGS
        ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
        MOV      #SCMTAG,R6  ;:FIRST LOCATION TO BE CLEARED
        CLR      (R6)+      ;:CLEAR MEMORY LOCATION
        CMP      #SWR,R6    ;:DONE?
        BNE     #-6        ;:LOOP BACK IF NO
        MOV      #STACK,SP  ;:SETUP THE STACK POINTER
        ;:INITIALIZE A FEW VECTORS
        MOV      #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
        MOV      #340,@IOTVEC+2 ;:LEVEL 7
        MOV      #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
        MOV      #340,@EMTVEC+2 ;:LEVEL 7
        MOV      #STRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
        MOV      #340,@TRAPVEC+2 ;:LEVEL 7
        MOV      #SPURDN,@PWRVEC ;:POWER FAILURE VECTOR
        MOV      #340,@PWRVEC+2 ;:LEVEL 7
        MOV      $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
        CLR      $TIMES      ;:INITIALIZE NUMBER OF ITERATIONS
        CLR      $ESCAPE     ;:CLEAR THE ESCAPE ON ERROR ADDRESS
        MOV     #1,$SERMAX   ;: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
    
```

```

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

005700 005037 001230      CLR    $PASS           ;;CLEAR PASS COUNT
005704 132737 000200 001243  BITB  #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
005712 001403      BEQ    67$            ;;YES,USE NON-APT SWITCH
005714 012737 001244 001154  MOV    #SSWREG,SWR    ;;NO,USE APT SWITCH REGISTER
005722
24 005722 012737 005344 000004 67$:  ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 005730 012737 000300 000006  MOV    #BADTMO,ERRVEC ;SETUP FOR UNEXPECTED TIMEOUT
26  MOV    #PR6,ERRVEC+2 ;LEVEL 6
27
28  ;SBTTL TYPE PROGRAM NAME
    ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005736 005227 177777      INC    #-1            ;;FIRST TIME?
005742 001033      BNE    68$           ;;BRANCH IF NO
005744 022737 033756 000042  CMP    #SENDAD,@#42  ;;ACT-11?
005752 001427      BEQ    68$           ;;BRANCH IF YES
005754 104401 005762      TYPE  ,69$          ;;TYPE ASCIZ STRING
005760 000424      BR     68$          ;;GET OVER THE ASCIZ
    ;;69$: .ASCIZ <CRLF>@CZRNF AO - RM80 FUNCTIONAL TEST, PT 3@<CRLF>
    68$:
    ;SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
006032 005737 000042      TST    @#42          ;;ARE WE RUNNING UNDER XXDP/ACT?
006036 001012      BNE    70$          ;;BRANCH IF YES
006040 123727 001242 000001  CMPB  $ENV,#1        ;;ARE WE RUNNING UNDER APT?
006046 001406      BEQ    70$          ;;BRANCH IF YES
006050 023727 001154 000176  CMP    SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
006056 001005      BNE    71$          ;;BRANCH IF NO
006060 104407      GTSWR              ;;GET SOFT-SWR SETTINGS
006062 000403      BR     71$
006064 112737 000001 001150 70$:  MOVB  #1,$AUTOB     ;;SET AUTO-MODE INDICATOR
006072 71$:

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

```



```

47 006201      000
48 006202 104401 001217      .BYTE 0      ;SUPRESS LEADING ZEROS
49 006206 000460      TYPE 3$      ;CR-LF
50      2$:      BR 3$      ;GET NUMBER OF DRIVES
51 006210 005227 177777      INC #-1      ;FIRST TIME THRU HERE ?
52 006214 001055      BNE 3$      ;NO
53 006216 104401 006224      TYPE 75$     ;TYPE ASCIZ STRING
      006222 000410      BR 74$     ;GET OVER THE ASCIZ
      ;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
      74$:
54 006244 005046      CLR -(SP)    ;CLEAR WORD ON STACK
55 006246 113716 001332      MOVB XXDP,(SP) ;GET DRIVE ADDRESS
56 006252 104403      TYPOS      ;TYPE DRIVE ADDRESS
57 006254      001      .BYTE 1      ;ONLY 1 CHARACTER
58 006255      000      .BYTE 0      ;SUPRESS LEADING ZEROS
59 006256 104401 006264      TYPE 76$     ;TYPE ASCIZ STRING
      006262 000432      BR 3$      ;GET OVER THE ASCIZ
      ;:76$: .ASCIZ /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
      3$:
63
64      ;CHECK FOR AUTO MODE OR STANDALONE MODE
65 006350 005737 000042      TST 2#42    ;RUNNING IN AUTO MODE ?
66 006354 001537      BEQ STANDALONE ;BR IF NO
67 006356 012737 000377 001300      MOV #377,$DEV ;SET DEVICE MAP FOR ALL DRIVES
68
69      ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
70      XSIZ:
71 006364 132737 000200 001243      BITB #BIT7,$ENVM ;SIZING ALLOWED ?
72 006372 001124      BNE 12$     ;NO
73
74 006374 005001      CLR R1      ;START FROM DRIVE 0
75 006376 013700 001276      MOV $BASE,R0 ;LOAD THE BASE ADDRESS
76 006402 104401 067044      TYPE ,SYSTAT ;TYPE 'UNIT STATUS:'
77
78 006406 136137 067344 001300 1$: BITB ATNTBL(R1),$DEV ;IS DEVICE PRESENT IN MAP ?
79 006414 001507      BEQ 11$     ;BR IF NO
80 006416 104401 001217      TYPE 3$      ;CR-LF
81 006422 010146      MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
      006424 104403      TYPOS      ;GO TYPE--OCTAL ASCII
      006426      002      .BYTE 2      ;TYPE 2 DIGIT(S)
      006427      000      .BYTE 0      ;SUPPRESS LEADING ZEROS
82 006430 104401 067236      TYPE ,BLNKS4 ;TYPE 4 BLANKS
83
84 006434 012760 000040 000010      MOV #CLR, RMCS2(R0) ;CLEAR MASS BUS
85 006442 010160 000010      MOV R1, RMCS2(R0) ;LOAD THE DRIVE ADDRESS
86 006446 005760 000012      TST RMD5(R0) ;ACCESS DRIVE REGISTER
87 006452 032760 010000 000010      BIT #NED, RMCS2(R0) ;IS DRIVE PRESENT ?
88 006460 001027      BNE 3$      ;BR IF NO
89 006462 032760 004000 000000      BIT #DVA, RMCS1(R0) ;IS DRIVE AVAILABLE ?
90 006470 001426      BEQ 4$      ;BR IF NO
91 006472 012737 067062 006632      MOV #SRM80, 10$ ;ASSUME RM80 DEVICE
92 006500 022760 020026 000026      CMP #20026, RMDT(R0) ;SINGLE PORT RM80 ?
93 006506 001407      BEQ 2$      ;BR IF YES
94 006510 022760 024026 000026      CMP #24026, RMDT(R0) ;DUAL PORT RM80 ?
95 006516 001403      BEQ 2$      ;BR IF YES
96 006520 104401 067067      TYPE ,NOTRM ;DRIVE NOT AN RM80
97 006524 000412      BR 5$      ;CHECK NEXT DRIVE
    
```

98	006526	032760	010000	000012	2\$:	BIT	#MOL,RMDS(R0)	:IS MEDIUM ON LINE ?
99	006534	001412				BEQ	6\$:BR IF NO
100	006536	000417				BR	7\$	
101								
102	006540	104401	067121		3\$:	TYPE	.NOTPRS	:DRIVE NOT PRESENT
103	006544	000402				BR	5\$:CHECK NEXT DRIVE
104								
105	006546	104401	067136		4\$:	TYPE	.NOTAVL	:DRIVE NOT AVAILABLE
106	006552	146137	067344	001300	5\$:	BICB	ATNTBL(R1),SDEVM	:CLEAR DEVICE FROM BIT MAP
107	006560	000425				BR	11\$:CHECK NEXT DRIVE
108								
109	006562	104401	067155		6\$:	TYPE	.UNTOFF	:DRIVE OFFLINE
110	006566	146137	067344	001300		BICB	ATNTBL(R1),SDEVM	:CLEAR DEVICE FROM BIT MAP
111	006574	000413				BR	9\$:PRINT DRIVE TYPE
112								
113	006576	005737	001332		7\$:	TST	XXDP	:LOADED FROM RM80 ?
114	006602	001406				BEQ	8\$:NO
115	006604	123701	001332			CMPB	XXDP,R1	:IS THIS THE DRIVE ?
116	006610	001003				BNE	8\$:BR IF NO
117	006612	104401	067104			TYPE	.LODEV	:DRIVE IS LOAD DEVICE
118	006616	000755				BR	5\$	
119								
120	006620	104401	067166		8\$:	TYPE	.UNTON	:DRIVE ONLINE
121	006624	104401	067240		9\$:	TYPE	.BLNKS2	:TYPE 2 BLANKS
122	006630	104401				TYPE		:PRINT DRIVE TYPE
123	006632	000000			10\$:	.WORD	0	:MESSAGE ADDRESS HERE
124								
125	006634	005201			11\$:	INC	R1	:INCREMENT THE DRIVE ADDRESS
126	006636	020127	000007			CMP	R1,#7	:ALL DRIVES ARE CHECKED ?
127	006642	003661				BLE	1\$:BRANCH IF NOT
128								
129	006644	104401	001217		12\$:	TYPE	.SCRLF	:CR-LF
130	006650	000137	007332			JMP	CMNSTART	:JUMP TO COMMON START

```

1      .SBTTL  STANDALONE INPUT ROUTINES
2
3      006654
4      006654  004737  063670
5
6      006660  005227  177777
7      006664  001023
8
9      ;SEE IF OPERATOR WANTS HELP TEXT
10
11     006666  104401  065110          TYPE      ,MSHELP          ;WANT HELP ?
12     006672  104411                      RDCHR          ;GET RESPONSE
13     006674  012637  001176          MOV      (SP)+,$TMP1      ;SAVE AND ECHO RESPONSE
14     006700  123727  001176  000131  CMPB     $TMP1,#'Y        ;WAS IT A YES RESPONSE ?
15     006706  001005                      BNE      1$              ;NO
16     006710  104401  001176          TYPE     , $TMP1         ;TYPE 'Y'
17     006714  104401  104254          TYPE     ,HELP          ;YES - TYPE HELP TEXT
18     006720  000414                      BR       3$              ;
19     006722  104401  067232  1$:      TYPE     ,N              ;TYPE 'N'
20     006726  104401  001217          TYPE     , $CRLF        ;CR-LF
21     006732  000407                      BR       3$              ;
22
23     ;SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
24     006734
25     006734  005737  001330  2$:      TST      CHGADR          ;CHANGE RH/RM BUS ADDRESS ?
26     006740  001457                      BEQ      7$              ;BR IF NO
27     006742  005037  001330          CLR     CHGADR          ;NO CHANGE NEXT TIME
28     006746  104401  001217          TYPE     , $CRLF        ;CR-LF
29
30     ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
31     006752  3$:
32     006752  104401  066521          TYPE     ,CNSL01        ;TYPE CURRENT BUS ADDRESS
33     006756  013746  001276          MOV     $BASE,-(SP)     ;SAVE $BASE FOR TYPEOUT
34     006762  104402                      TYPOC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
35     006764  104401  067240          TYPE     ,BLNKS2        ;TYPE 2 BLANKS
36     006770  104413                      RDOCT          ;GET NEW BUS ADDRESS
37     006772  012637  001176          MOV     (SP)+,$TMP1     ;CARRIAGE RETURN ?
38     006776  001412                      BEQ      5$              ;YES-SKIP TO NEXT ENTRY
39     007000  022737  160000  001176  CMP     #160000,$TMP1    ;BASE ADDRESS IN I/O PAGE ?
40     007006  101403                      BLOS     4$              ;YES
41     007010  104401  066531          TYPE     ,CNSL02        ;TYPE WARNING MESSAGE
42     007014  000756                      BR       3$              ;TRY AGAIN
43     007016  013737  001176  001276  4$:      MOV     $TMP1,$BASE     ;STORE NEW BUS ADDRESS
44     007024  104401  066573          5$:      TYPE     ,CNSL03
45     007030  005046                      CLR     -(SP)
46     007032  113716  001272          MOVB   $VECT1,(SP)     ;GET CURRENT VECTOR ADDRESS
47     007036  104402                      TYPOC
48     007040  104401  067240          TYPE     ,BLNKS2        ;TYPE 2 BLANKS
49     007044  104413                      RDOCT          ;GET NEW VECTOR ADDRESS
50     007046  012637  001176          MOV     (SP)+,$TMP1     ;CARRIAGE RETURN?
51     007052  001412                      BEQ      7$              ;YES-SKIP TO NEXT ENTRY
52     007054  022737  001000  001176  CMP     #1000,$TMP1     ;VECTOR ADDRESS < 1000 ?
53     007062  101003                      BHI     6$              ;YES!!
54     007064  104401  066602          TYPE     ,CNSL04        ;TYPE WARNING MESSAGE
55     007070  000755                      BR       5$              ;RETRY
56     007072  113737  001176  001272  6$:      MOVB   $TMP1,$VECT1    ;STORE NEW VECTOR ADDRESS
    
```



```

57
58          ;DIALOGUE TO INPUT DEVICE NUMBERS
59 007100 005227 177777      7$: INC # -1          ;FIRST TIME THRU ?
60 007104 001002              BNE 8$          ;BR IF NO
61 007106 104401 066636      TYPE ,CNSLO7      ;TYPE INPUT INSTRUCTIONS
62 007112 104401 001217      8$: TYPE ,SCRLF      ;CR-LF
63 007116 005037 001300      9$: CLR $DEV      ;CLEAR DEVICE MAP
64 007122 104401 067022      TYPE ,MSDRVS      ;TYPE 'DRIVE(S): '
65 007126 104411              RDCHR
66 007130 012637 001176      MOV (SP)+,$TMP1      ;GET RESPONSE
67 007134 023727 001176 000101  CMP $TMP1,#'A      ;IS INPUT 'A' ?
68 007142 001007              BNE 10$         ;NO
69 007144 104401 066074      TYPE ,ALL          ;YES, TYPE 'ALL' AND GO
70 007150 012737 000377 001300  MOV #377,$DEV      ;SET DEVICE MAP FOR ALL DRIVES
71 007156 000137 006364      JMP XSIZ          ;AUTO SIZE.
72
73 007162 023727 001176 000015 10$: CMP $TMP1,#CR      ;CARRIAGE RETURN ?
74 007170 001436              BEQ 12$         ;YES
75 007172 104401 001176      TYPE $TMP1        ;ECHO RESPONSE
76 007176 023727 001176 000060  CMP $TMP1,#'0      ;NUMBER < 0 ?
77 007204 002430              BLT 12$         ;YES
78 007206 023727 001176 000067  CMP $TMP1,#'7      ;NUMBER > 7 ?
79 007214 003427              BLE 13$         ;NO
80 007216 000423              BR 12$          ;ILLEGAL INPUT
81
82 007220 104411              11$: RDCHR
83 007222 012637 001176      MOV (SP)+,$TMP1      ;GET RESPONSE
84 007226 023727 001176 000015  CMP $TMP1,#CR      ;CARRIAGE RETURN ?
85 007234 001432              BEQ 14$         ;YES
86 007236 104401 066105      TYPE ,COMMA       ;TYPE ' '
87 007242 104401 001176      TYPE $TMP1        ;ECHO RESPONSE
88 007246 023727 001176 000060  CMP $TMP1,#'0      ;NUMBER < 0 ?
89 007254 002404              BLT 12$         ;YES
90 007256 023727 001176 000067  CMP $TMP1,#'7      ;NUMBER > 7 ?
91 007264 003403              BLE 13$         ;NO
92 007266 104401 067000      12$: TYPE ,CNSLO8      ;TYPE '' ?ILLEGAL INPUT''
93 007272 000711              BR 9$           ;RETRY
94
95 007274 013701 001176      13$: MOV $TMP1,R1      ;R1 = DRIVE NUMBER
96 007300 042701 177770      BIC #'C7,R1
97 007304 156137 067344 001300  BISB ATNTBL(R1),$DEV ;SET DEVICE IN MAP
98 007312 122737 000377 001300  CMPB #377,$DEV      ;DONE ?
99 007320 101337              BHI 11$         ;NO
100 007322 104401 001217      14$: TYPE ,SCRLF      ;CR-LF
101 007326 000137 006364      JMP XSIZ          ;GO SIZE DEVICES
    
```

```

1
2 007332
3 007332 104401 067176
4 007336 013700 001300
5 007342 001004
6 007344 104401 066105
7 007350 104401 067225
8 007354 012701 001470
9 007360 010137 001466
10 007364 012702 000001
11 007370 005003
12 007372 030200
13 007374 001413
14 007376 104401 066105
15 007402 010311
16 007404 010346
    007406 104403
    007410 001
    007411 000
17 007412 116361 067344 000001
18 007420 062701 000002
19 007424 006302
20 007426 105702
21 007430 001402
22 007432 005203
23 007434 000756
24 007436 005011
25 007440 104401 001217
26
27
28 007444 004737 040610
29 007450 000425
30 007452 104401 007460
    007456 000413
    007506
31 007506 005737 000042
32 007512 001002
33 007514 000137 005434
34 007520 000137 033746
35 007524
36
37 007524 000240
38 007526 105737 001300
39 007532 001007
40 007534 005737 000042
41 007540 001002
42 007542 000137 005434
43 007546 000137 033746
44
45 007552 105037 001116
46 007556 005037 001206
47 007562 005037 001326
48 007566 005037 001512
49 007572 004737 063670
50 007576 012746 000240
    007602 012746 007610

```

```

:ASSEMBLE TEST QUE FROM DEVICE MAP
CMNSTART:
    TYPE      DRIVES      :TYPE 'DRIVE(S) TO BE TESTED'
    MOV      $DEVN,R0      :R0 = DEVICE MAP
    BNE      1$            :BR IF DRIVES TO TEST
    TYPE      ,COMMA       :TYPE ' '
    TYPE      ,NONE        :TYPE 'NONE'
1$:  MOV      #TSTQUE+2,R1  :R1 = ADDRESS OF FIRST ENTRY IN QUE
    MOV      R1,TSTQUE     :INITIALIZE ENTRY POINTER
    MOV      #1,R2         :R2 = DEVICE POINTER
    CLR      R3            :R3 = DEVICE NUMBER
2$:  BIT      R2,R0         :IS THIS DEVICE IN MAP ?
    BEQ      3$            :NO !!
    TYPE      ,COMMA       :TYPE ' '
    MOV      R3,(R1)       :YES - ENTER DEVICE NUMBER IN QUE
    MOV      R3,-(SP)      :SAVE R3 FOR TYPEOUT
    TYPOS    1            :GO TYPE--OCTAL ASCII
    .BYTE    0            :TYPE 1 DIGIT(S)
    .BYTE    0            :SUPPRESS LEADING ZEROS
    MOVB     ATNTBL(R3),1(R1) :ENTER ATTENTION BIT IN QUE
3$:  ADD      #2,R1         :ADVANCE ENTRY POINTER
    ASL      R2            :ADVANCE DEVICE POINTER
    TSTB     R2            :DONE ALL DEVICES ?
    BEQ      4$            :YES
    INC      R3            :ADVANCE DEVICE NUMBER
4$:  BR       2$           :ENTER NEXT DEVICE
    CLR      (R1)          :TERMINATE TEST QUE
    TYPE     ,SCRLF        :CR-LF

:SIZE FOR CLOCK
    JSR     PC,SIZCLK      :SEE IF CLOCK PRESENT
    BR      6$            :YES - CLOCK IS PRESENT
    TYPE    ,65$          :TYPE ASCII STRING
    BR      64$          :GET OVER THE ASCII
64$: .ASCIZ <CRLF>/NO 'L' OR 'P' CLOCK/

65$: TST      @#42         :ANY MONITOR PRESENT ?
    BNE     5$           :BR IF YES
    JMP     START        :JUMP TO START
5$:  JMP     $GET42      :RETURN CONTROL TO MONITOR
6$:

READY: NOP
    TSTB   $DEVN         :READY TO START TEST
    BNE    2$           :ANY DRIVES IN MAP ?
    TST    @#42         :ANY MONITOR PRESENT ?
    BNE    1$           :BR IF YES
    JMP    START        :JUMP TO START
1$:  JMP    $GET42      :RETURN CONTROL TO MONITOR
2$:  CLRB   $TSTNM       :RESET TEST NUMBER
    CLR    $TIMES       :INITIALIZE NUMBER OF ITERATIONS
    CLR    CTLFG        :CLEAR CONTROL-C FLAG
    CLR    MEDENB       :CLEAR MEDIA ENABLE
    JSR    PC,$TKINT    :INITIALIZE TTY
    MOV    #PR5,-(SP)   :PUT NEW PS ON STACK
    MOV    #64$,-(SP)  :PUT NEW PC ON STACK

```

```

007606 000002      RTI      ;:POP NEW PC AND PS
007610
51 007610 117737 171652 001234 64$:  MOVB  @TSTQUE,$UNIT ;LOAD DRIVE NUMBER
52
53 ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND SET LAST TRACK ADDRESS
54 007616 013700 001276      MOV  $BASE,R0 ;R0 = UNIBUS ADDRESS
55 007622 012760 000040 000010  MOV  #CLR, RMCS2(R0) ;CLEAR MASSBUS
56 007630 113760 001234 000010  MOVB $UNIT, RMCS2(R0) ;SELECT DEVICE UNDER TEST
57 007636 012737 006400 001334  MOV  #TAB!TA4!TA1, LSTRK ;SET LAST TRACK = 13.
58
59 ;TYPE DRIVE NUMBER TO BE TESTED($UNIT)
60 007644 104401 001217      TYPE  ,SCRLF ;CR-LF
61 007650 104401 067036      TYPE  ,MSGDRV ;TYPE 'DRIVE'
62 007654 013746 001234      MOV   $UNIT,-(SP) ;:SAVE $UNIT FOR TYPEOUT
                                ;:TYPE DRIVE NUMBER
                                ;:GO TYPE--OCTAL ASCII
                                ;:TYPE 2 DIGIT(S)
                                ;:SUPPRESS LEADING ZEROS
                                ;:THESE TWO LOOPS ARE ADDED TO
                                ;:WAIT FOR TTY
                                007660 104403      TYPOS
                                007662      .BYTE 2 ;TYPE 2 DIGIT(S)
                                007663      .BYTE 0 ;SUPPRESS LEADING ZEROS
63 007664 005004      CLR  R4 ;THESE TWO LOOPS ARE ADDED TO
64 007666 005304      DEC  R4 ;WAIT FOR TTY
65 007670 001376      BNE  .-2
66 007672 005304      DEC  R4
67 007674 001376      BNE  .-2
    
```


1
2

 :*TEST 1 CONTROLLER ACCESS TEST

 TST1:

007676
 007676 000004
 007700 000240
 007702 012706 001100
 007706 013700 001276
 007712 013701 001466
 007716 012737 000001 001226
 3
 4 007724 005001
 5 007726 013746 000004
 007732 013746 000006
 6 007736 012737 010040 000004
 7 007744 012737 000300 000006
 8
 9 007752 110160 000001
 10 007756 010160 000002
 11 007762 016002 000002
 12 007766 010160 000004
 13 007772 016002 000004
 14 007776 016046 000010
 15 010002 010160 000010
 16 010006 016002 000010
 17 010012 012660 000010
 18 010016 010160 000022
 19 010022 016002 000022
 20 010026 012637 000006
 010032 012637 000004
 21 010036 000417
 22
 23 010040 022626
 24 010042 012637 000006
 010046 012637 000004
 25 010052 104110
 26 010054 005737 000042
 27 010060 001002
 28 010062 000137 005424
 29 010066 005037 001300
 30 010072 000137 033746
 31 010076
 32
 33

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 #1,\$TESTN :SET TEST NUMBER IN APT MAIL BOX
 CLR R1
 MOV ERRVEC,-(SP) :PUSH ERRVEC ON STACK
 MOV ERRVEC+2,-(SP) :PUSH ERRVEC+2 ON STACK
 MOV #1\$,ERRVEC
 MOV #PR6,ERRVEC+2
 MOV R1,RMCS1+1(R0) :MOVE HI BYTE TO RMCS1
 MOV R1,RMWC(R0) :MOVE WORD COUNT REGISTER
 MOV RMWC(R0),R2
 MOV R1,RMBA(R0) :MOVE BUS ADDRESS REGISTER
 MOV RMBA(R0),R2
 MOV RMCS2(R0),-(SP) :PUSH RMCS2(R0) ON STACK
 MOV R1,RMCS2(R0) :MOVE CONTROL STATUS REGISTER
 MOV RMCS2(R0),R2
 MOV (SP)+,RMCS2(R0) :POP STACK INTO RMCS2(R0)
 MOV R1,RMDB(R0) :MOVE DATA BUFFER
 MOV RMDB(R0),R2
 MOV (SP)+,ERRVEC+2 :POP STACK INTO ERRVEC+2
 MOV (SP)+,ERRVEC :POP STACK INTO ERRVEC
 BR 3\$:NO BUS TIMEOUT OCCURRED
 1\$: CMP (SP)+,(SP)+ :ADJUST STACK
 MOV (SP)+,ERRVEC+2 :POP STACK INTO ERRVEC+2
 MOV (SP)+,ERRVEC :POP STACK INTO ERRVEC
 EMT 110
 TST 2#42 :STAND ALONE MODE ?
 BNE 2\$:NO!!
 JMP START1 :YES-GO GET \$BASE
 2\$: CLR \$DEVN :FUDGE NO DRIVES IN MAP
 JMP \$GET42 :RETURN CONTROL TO MONITOR
 3\$:

 :*TEST 2 WRITE, READ ZEROS

 TST2:

010076
 010076 000004
 010100 000240
 010102 012706 001100
 010106 013700 001276
 010112 013701 001466
 010116 012737 000002 001226
 34
 35
 36

SCOPE :SCOPE CALL
 NOP :START OF TEST
 MOV #STACK,SP :INITIALIZE STACK POINTER
 MOV \$BASE,R0 :R0 = UNIBUS ADDRESS
 MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
 MOV #2,\$TESTN :SET TEST NUMBER IN APT MAIL BOX

 :LOOP #1 FORMAT,WRITE,READ

```

37
38 010124          1$:
39
40
41 010124 004737 034032      :PREPARE DEVICE FOR TEST
    010130 154130          JSR   PC,TSTPRP      :PREPARE DEVICE FOR TEST
                                .WORD 154130      :TASK DESCRIPTOR AS FOLLOWS:
                                :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                :CLEAR CONTROLLER & SELECT DEVICE
                                :VERIFY CONTROLLER CLEAR OPERATION
                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                :VERIFY PACK ACKNOWLEDGE
                                :RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                :VERIFY RECALIBRATION
                                :GO TO 2$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 25$ IF ERROR

    010132 000404          BR    2$
    010134 000240          NOP
    010136 104000          EMT
    010140 000137 011104    JMP   25$

42
43      :LOAD PARAMETERS AND GENERATE DATA BUFFER
44 2$:
45 010144          MOV   #559,RMDCO      :CYLINDER = 559.
46 010152          MOV   #TA2,RMDAO      :TRACK = 2, SECTOR = 0
47 010160          MOV   #BUFONE,RMBAO    :BUS ADDRESS
48 010166          MOV   #-258,RMWCO     :2 + 256. WORDS (2'S COMP)
49 010174          MOV   #FMT16,RMFOF    :16 BIT FORMAT
50 010202          MOV   #WH!GO,RMCS10    :WRITE HEADER AND DATA COMMAND
51
52      :VERIFY THAT SECTOR IS NOT BAD
    010210 004737 034762    JSR   PC,BADSCT      :CALL BAD SECTOR MODULE
    010214 000405          BR    3$      :GO TO 3$ IF NO ERROR
    010216 104401 066006    TYPE   ,SCTMSG      :TYPE BAD SECTOR MESSAGE
    010222 104000          EMT
    010224 000137 011104    JMP   25$      :ERROR # DEFINED BY BADSCT SUBROUTINE
                                :GO TO 25$ IF ERROR
53 3$:
54 010230          MOV   #ZEROS,$TMP0     :STARTING ADDRESS OF PATTERN
55 010236          MOV   #1,$TMP1        :RANGE OF PATTERN
56 010244          JSR   PC,GENBUF       :GO GENERATE BUFFER FOR FORMAT
57
58      :SETUP PARAMETERS AND EXECUTE COMMAND
59 010250          MOV   #PUTINX,R2      :R2 = BYTE ENTRY POSITION
60 010254          MOVB  #RMDC,(R2)+
61 010260          MOVB  #RMBA,(R2)+
62 010264          MOVB  #RMWC,(R2)+
63 010270          MOVB  #RMFOF,(R2)+
64 010274          MOVB  #RMCS1,(R2)+
65 010300          MOVB  #200,(R2)
66 010304          MOVB  #200,(R2)      :TERMINATE TABLE
67 4$:
68 010310          JSR   PC,PUT          :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    010314 000404          BR    5$      :GO TO 5$ IF NO ERROR
    010316 000240          NOP
    010320 104000          EMT
    010322 000137 011104    JMP   25$      :ERROR # DEFINED BY PUT SUBROUTINE
                                :GO TO 25$ IF ERROR
69 5$:
70
71      :SETUP GET INDEX TABLE TO READ ALL REGISTERS
    010326 004737 040024    JSR   PC,GETSTS     :GO TO GETSTS SUBROUTINE
    
```



```

72
73      010332 004737 040732      :WAIT FOR COMMAND TO COMPLETE
      JSR      PC,TIMOUT      :GO TO TIMEOUT SUBROUTINE
74
75      010336 004737 040110      :GO GET REGISTER STATUS
      JSR      PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
76      010342 000404      BR      6$      :GO TO 6$ IF NO ERROR
      010344 000240      NOP      :RETURN HERE IF ERROR
      010346 104000      EMT      :ERROR # DEFINED BY GET SUBROUTINE
      010350 000137 011104      JMP      25$      :GO TO 25$ IF ERROR
77      010354
78
79      010354 004737 054166      :VERIFY RESULTS OF WRITE COMMAND
      JSR      PC,DTASTS      :GO VERIFY RESULTS OF DATA TRANSFER
80      010360 000405      BR      7$      :GO TO 7$ IF NO ERROR
      010362 000240      NOP      :RETURN HERE IF ERROR
      010364 104000      EMT      :ERROR # DEFINED BY DTASTS SUBROUTINE
      010366 004736      JSR      PC,@(SP)+      :GO BACK FOR MORE ERROR CHECKS
      010370 000137 011104      JMP      25$      :GO TO 25$ IF ERROR
81
82
83      010374 012737 010404 001124 :MOVE LOOP ADDRESSES TO NEXT OPERATION
84      MOV      #8$,SLPERR
85      010402 000410      BR      9$      :SKIP TO WRITE OPERATION
86
87      ::*****
88      :LOOP #2      WRITE,READ
89
90      010404
91
92
93      010404 004737 034032      :PREPARE DEVICE FOR TEST
      010410 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 9$ IF NO ERROR
      :RETURN HERE IF ERROR
      :ERROR # DEFINED BY TSTPRP SUBROUTINE
      :GO TO 25$ IF ERROR
      BR      9$
      NOP
      EMT
      JMP      25$
94
95
96
97      010424
98      010424 012737 104260 001416 :SETUP PARAMETERS AND EXECUTE COMMAND
99      010432 012737 177400 001414 :MOVE MEMORY ADDRESS
100     010440 012737 000061 001412 :CHANGE WORD COUNT
101     010446 012702 001555 :WRITE DATA COMMAND
102     010452 112722 000006 :LOAD PUT REGISTER INDEX TABLE
103     010456 112722 000034      MOV      #BUFONE+4,RMBAD
104     010462 112722 000032      MOV      #-256.,RMWCO
105     010466 112722 000004      MOV      #WD!GO,RMCS10
106     010472 112722 000002      MOV      #PUTINX,R2
      MOV      #RMDA,(R2)+
      MOV      #RMDC,(R2)+
      MOV      #RMCF,(R2)+
      MOV      #RMBA,(R2)+
      MOV      #RMWC,(R2)+

```



```

107 010476 112722 000000      MOVB  #RMCS1,(R2)+
108 010502 112722 000200      MOVB  #200,(R2)+                ;TERMINATE TABLE
109
110 010506 004737 040360      JSR   PC,PUT                    ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    010512 000404                BR    11$                       ;GO TO 11$ IF NO ERROR
    010514 000240                NOP                               ;RETURN HERE IF ERROR
    010516 104000                EMT                               ;ERROR # DEFINED BY PUT SUBROUTINE
    010520 000137 011104        JMP   25$                       ;GO TO 25$ IF ERROR
111 010524                    11$:
112
113 010524 004737 040024      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
    JSR   PC,GETSTS                ;GO TO GETSTS SUBROUTINE
114
115 010530 004737 040732      ;WAIT FOR COMMAND TO COMPLETE
    JSR   PC,TIMOUT                ;GO TO TIMOUT SUBROUTINE
116
117 010534 004737 040110      ;GO GET REGISTER STATUS
    JSR   PC,GET                    ;GO READ REGISTER(S) WITH GET SUBROUTINE
    010540 000404                BR    12$                       ;GO TO 12$ IF NO ERROR
    010542 000240                NOP                               ;RETURN HERE IF ERROR
    010544 104000                EMT                               ;ERROR # DEFINED BY GET SUBROUTINE
    010546 000137 011104        JMP   25$                       ;GO TO 25$ IF ERROR
119 010552                    12$:
120
121 010552 004737 041126      ;VERIFY RESULTS OF WRITE COMMAND
    JSR   PC,PRIERR                ;GO CHECK FOR PRIMARY ERRORS
    010556 000405                BR    13$                       ;GO TO 13$ IF NO ERROR
    010560 000240                NOP                               ;RETURN HERE IF ERROR
    010562 104000                EMT                               ;ERROR # DEFINED BY PRIERR SUBROUTINE
    010564 004736                JSR   PC,@(SP)+                ;GO BACK FOR MORE ERROR CHECKS
    010566 000137 011104        JMP   25$                       ;GO TO 25$ IF ERROR
123 010572                    13$:
124 010572 004737 054166      JSR   PC,DTASTS                ;GO VERIFY RESULTS OF DATA TRANSFER
    010576 000405                BR    14$                       ;GO TO 14$ IF NO ERROR
    010600 000240                NOP                               ;RETURN HERE IF ERROR
    010602 104000                EMT                               ;ERROR # DEFINED BY DTASTS SUBROUTINE
    010604 004736                JSR   PC,@(SP)+                ;GO BACK FOR MORE ERROR CHECKS
    010606 000137 011104        JMP   25$                       ;GO TO 25$ IF ERROR
125 010612                    14$:
126 010612 004737 041760      JSR   PC,SECERR                ;GO CHECK FOR SECONDARY ERRORS
    010616 000405                BR    15$                       ;GO TO 15$ IF NO ERROR
    010620 000240                NOP                               ;RETURN HERE IF ERROR
    010622 104000                EMT                               ;ERROR # DEFINED BY SECERR SUBROUTINE
    010624 004736                JSR   PC,@(SP)+                ;GO BACK FOR MORE ERROR CHECKS
    010626 000137 011104        JMP   25$                       ;GO TO 25$ IF ERROR
127 010632                    15$:
128
129 010632 012737 010642 001124 ;CHANGE LOOP ADDRESSES
    MOV   #16$,SLPERR
131 010640 000410                BR    17$                ;SKIP TO NEXT OPERATION
132
133 *****
134 ;LOOP #3      READ
135
136 010642                    16$:
137
138 ;PREPARE DEVICE FOR TEST
    
```

139	010642	004737	034032		JSR	PC,TSTPRP		:PREPARE DEVICE FOR TEST
	010646	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
	010650	000404			BR	17\$:GO TO 17\$ IF NO ERROR
	010652	000240			NOP			:RETURN HERE IF ERROR
	010654	104000			EMT			:ERROR # DEFINED BY TSTPRP SUBROUTINE
	010656	000137	011104		JMP	25\$:GO TO 25\$ IF ERROR
140	010662			17\$:				
141								
142								
143	010662							:SETUP PARAMETERS AND EXECUTE COMMAND
144	010662	012737	105264	001416	18\$:	MOV	#BUFTWO+4,RMBA0	:CHANGE MEMORY ADDRESS
145	010670	012737	000071	001412		MOV	#RD!GO,RMCS10	:READ DATA COMMAND
146	010676	012702	001555			MOV	#PUTINX,R2	:LOAD PUT REGISTER INDEX TABLE
147	010702	112722	000006			MOVB	#RMDA,(R2)+	
148	010706	112722	000032			MOVB	#RMOF,(R2)+	
149	010712	112722	000034			MOVB	#RMDC,(R2)+	
150	010716	112722	000004			MOVB	#RMBA,(R2)+	
151	010722	112722	000002			MOVB	#RMWC,(R2)+	
152	010726	112722	000000			MOVB	#RMCS1,(R2)+	
153	010732	112712	000200			MOVB	#200,(R2)	
154								
155	010736	004737	040360		JSR	PC,PUT		:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	010742	000404			BR	19\$:GO TO 19\$ IF NO ERROR
	010744	000240			NOP			:RETURN HERE IF ERROR
	010746	104000			EMT			:ERROR # DEFINED BY PUT SUBROUTINE
	010750	000137	011104		JMP	25\$:GO TO 25\$ IF ERROR
156	010754			19\$:				
157								
158								
159	010754	004737	040024		:SETUP GET INDEX TABLE TO READ ALL REGISTERS			
160					JSR	PC,GETSTS		:GO TO GETSTS SUBROUTINE
161								
162					:WAIT FOR COMMAND TO COMPLETE			
163	010760	004737	040732		JSR	PC,TIMOUT		:GO TO TIMOUT SUBROUTINE
164								
165								
166					:GO GET REGISTER STATUS			
167	010764	004737	040110		JSR	PC,GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
	010770	000404			BR	20\$:GO TO 20\$ IF NO ERROR
	010772	000240			NOP			:RETURN HERE IF ERROR
	010774	104000			EMT			:ERROR # DEFINED BY GET SUBROUTINE
	010776	000137	011104		JMP	25\$:GO TO 25\$ IF ERROR
164	011002			20\$:				
165								
166								
167	011002	004737	041126		:VERIFY RESULTS OF READ COMMAND			
	011006	000405			JSR	PC,PRIERR		:GO CHECK FOR PRIMARY ERRORS
	011010	000240			BR	21\$:GO TO 21\$ IF NO ERROR
	011012	104000			NOP			:RETURN HERE IF ERROR
	011014	004736			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	011016	000137	011104		JSR	PC,@(SP)+		:GO BACK FOR MORE ERROR CHECKS
168	011022			21\$:	JMP	25\$:GO TO 25\$ IF ERROR


```

169 011022 004737 054166      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      011026 000405          BR      22$           ;GO TO 22$ IF NO ERROR
      011030 000240          NOP           ;RETURN HERE IF ERROR
      011032 104000          EMT           ;ERROR # DEFINED BY DTASTS SUBROUTINE
      011034 004736          JSR    PC,a(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      011036 000137 011104    JMP     25$           ;GO TO 25$ IF ERROR
170 011042          22$:
171 011042 004737 041760      JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      011046 000405          BR      23$           ;GO TO 23$ IF NO ERROR
      011050 000240          NOP           ;RETURN HERE IF ERROR
      011052 104000          EMT           ;ERROR # DEFINED BY SECERR SUBROUTINE
      011054 004736          JSR    PC,a(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      011056 000137 011104    JMP     25$           ;GO TO 25$ IF ERROR
172 011062          23$:
173
174          ;GO VERIFY DATA
175 011062 004737 037462      JSR    PC,CMPBUF     ;GO COMPARE WRITE, READ DATA BUFFERS
      011066 104260          .WORD  BUFOFF+4     ;STARTING ADDRESS OF WRITE BUFFER
      011070 105264          .WORD  BUFTWO+4    ;STARTING ADDRESS OF READ BUFFER
      011072 000404          BR      24$           ;GO TO 24$ IF NO ERROR
      011074 000240          NOP           ;RETURN HERE IF ERROR
      011076 104000          EMT           ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      011100 000137 011104    JMP     25$           ;GO TO 25$ IF ERROR
176 011104          24$:
177
178 011104          25$:
179
180          ;*****
          ;*TEST 3      WRITE, WRITE CHECK ZEROS
          ;*****
          TST3:
          SCOPE          ;SCOPE CALL
          NOP           ;START OF TEST
          MOV    #STACK,SP ;INITIALIZE STACK POINTER
          MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
          MOV    TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
          MOV    #3,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
181
182          ;*****
183          ;LOOP #1      FORMAT,WRITE,WRITE CHECK DATA
184
185 011132          1$:
186
187          ;PREPARE DEVICE FOR TEST
188 011132 004737 034032      JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      011136 154130          .WORD  154130       ;TASK DESCRIPTOR AS FOLLOWS:
          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
          ;CLEAR CONTROLLER & SELECT DEVICE
          ;VERIFY CONTROLLER CLEAR OPERATION
          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
          ;VERIFY PACK ACKNOWLEDGE
          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
          ;VERIFY RECALIBRATION
          BR      2$           ;GO TO 2$ IF NO ERROR
          NOP           ;RETURN HERE IF ERROR
          EMT           ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          JMP     24$         ;GO TO 24$ IF ERROR
011140 000404          BR      2$
011142 000240          NOP
011144 104000          EMT
011146 000137 012062      JMP     24$

```



```

189
190      ;LOAD PARAMETERS AND GENERATE DATA BUFFER
191 011152      2$:
192 011152      012737 001057 001446      MOV      #559.,RMDCO      ;CYLINDER = 559.
193 011160      012737 001000 001420      MOV      #TA2,RMDAO      ;TRACK = 2, SECTOR = 0
194 011166      012737 104254 001416      MOV      #BUFONE,RMBAO    ;BUS ADDRESS
195 011174      012737 177376 001414      MOV      #-258.,RMWCO     ;2 + 256. WORDS (2'S COMP)
196 011202      012737 010000 001444      MOV      #FMT16,RMOFO    ;16 BIT FORMAT
197 011210      012737 000063 001412      MOV      #WH!GO,RMCS10   ;WRITE HEADER AND DATA COMMAND
198
199      ;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT    ;CALL BAD SECTOR MODULE
      BR      3$           ;GO TO 3$ IF NO ERROR
      TYPE    ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
      EMT
      JMP     24$         ;ERROR # DEFINED BY BADSCT SUBROUTINE
                          ;GO TO 24$ IF ERROR
200 011236      3$:
201 011236      012737 067456 001174      MOV      #ZEROS,$TMP0    ;STARTING ADDRESS OF PATTERN
202 011244      012737 000001 001176      MOV      #1,$TMP1       ;RANGE OF PATTERN
203 011252      004737 037214      JSR      PC,GENBUF      ;GO GENERATE BUFFER FOR FORMAT
204
205      ;SETUP PARAMETERS AND EXECUTE COMMAND
      MOV     #PUTINX,R2    ;R2 = BYTE ENTRY POSITION
      MOVB   #RMDC,(R2)+
      MOVB   #RMDA,(R2)+
      MOVB   #RMBA,(R2)+
      MOVB   #RMWC,(R2)+
      MOVB   #RMOF,(R2)+
      MOVB   #RMCS1,(R2)+
      MOVB   #200,(R2)
206 011256      012702 001555
207 011262      112722 000034
208 011266      112722 000006
209 011272      112722 000004
210 011276      112722 000002
211 011302      112722 000032
212 011306      112722 000000
213 011312      112712 000200
214 011316
215 011316      004737 040360      4$:
      JSR     PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR     5$          ;GO TO 5$ IF NO ERROR
      NOP
      EMT
      JMP     24$         ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY PUT SUBROUTINE
                          ;GO TO 24$ IF ERROR
216 011334      5$:
217
218      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR     PC,GETSTS   ;GO TO GETSTS SUBROUTINE
219
220      ;WAIT FOR COMMAND TO COMPLETE
      JSR     PC,TIMOUT   ;GO TO TIMEOUT SUBROUTINE
221
222      ;GO GET REGISTER STATUS
223 011344      004737 040110      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      JSR     PC,GET
      BR     6$          ;GO TO 6$ IF NO ERROR
      NOP
      EMT
      JMP     24$         ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY GET SUBROUTINE
                          ;GO TO 24$ IF ERROR
224 011356      000137 012062
225 011362
226
227 011362      004737 054166      6$:
      ;VERIFY RESULTS OF WRITE COMMAND
      JSR     PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      BR     7$          ;GO TO 7$ IF NO ERROR
      NOP
      EMT
                          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY DTASTS SUBROUTINE
    
```

```

011374 004736 JSR PC,a(SP)+ ;GO BACK FOR MORE ERROR CHECKS
011376 000137 012062 JMP 24$ ;GO TO 24$ IF ERROR
228 011402 7$:
229
230 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
231 011402 012737 011412 001124 MOV #8$, $LPERR
232 011410 000410 BR 9$ ;SKIP TO WRITE OPERATION
233
234 ;*****
235 ;LOOP #2 WRITE,WRITE CHECK DATA
236
237 011412 8$:
238
239 ;PREPARE DEVICE FOR TEST
240 011412 004737 034032 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
011416 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 9$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 24$ IF ERROR
011420 000404 BR 9$
011422 000240 NOP
011424 104000 EMT
011426 000137 012062 JMP 24$
241 011432 9$:
242
243 ;SETUP PARAMETERS AND EXECUTE COMMAND
244 011432 10$:
245 011432 012737 104260 001416 MOV #BUFONE+4,RMBAO ;CHANGE MEMORY ADDRESS
246 011440 012737 177400 001414 MOV #-256.,RMWCO ;CHANGE WORD COUNT
247 011446 012737 000061 001412 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
248 011454 012702 001555 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
249 011460 112722 000006 MOVB #RMDA,(R2)+
250 011464 112722 000034 MOVB #RMDC,(R2)+
251 011470 112722 000032 MOVB #RMOF,(R2)+
252 011474 112722 000004 MOVB #RMBA,(R2)+
253 011500 112722 000002 MOVB #RMWC,(R2)+
254 011504 112722 000000 MOVB #RMCS1,(R2)+
255 011510 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
256
257 011514 004737 040360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011520 000404 BR 11$ ;GO TO 11$ IF NO ERROR
011522 000240 NOP ;RETURN HERE IF ERROR
011524 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
011526 000137 012062 JMP 24$ ;GO TO 24$ IF ERROR
258 011532 11$:
259
260 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
011532 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
261
262 ;WAIT FOR COMMAND TO COMPLETE
011536 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
263
264 ;GO GET REGISTER STATUS
    
```



```

265 011542 004737 040110      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
    011546 000404              BR     12$            ;GO TO 12$ IF NO ERROR
    011550 000240              NOP                    ;RETURN HERE IF ERROR
    011552 104000              EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
    011554 000137 012062      JMP     24$            ;GO TO 24$ IF ERROR
266 011560                      12$:
267
268
269 011560 004737 041126      ;VERIFY RESULTS OF WRITE COMMAND
    011564 000405              JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
    011566 000240              BR     13$            ;GO TO 13$ IF NO ERROR
    011570 104000              NOP                    ;RETURN HERE IF ERROR
    011572 004736              EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
    011574 000137 012062      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    011574 000137 012062      JMP     24$            ;GO TO 24$ IF ERROR
270 011600                      13$:
271 011600 004737 054166      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
    011604 000405              BR     14$            ;GO TO 14$ IF NO ERROR
    011606 000240              NOP                    ;RETURN HERE IF ERROR
    011610 104000              EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
    011612 004736              JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    011614 000137 012062      JMP     24$            ;GO TO 24$ IF ERROR
272 011620                      14$:
273 011620 004737 041760      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
    011624 000405              BR     15$            ;GO TO 15$ IF NO ERROR
    011626 000240              NOP                    ;RETURN HERE IF ERROR
    011630 104000              EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
    011632 004736              JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    011634 000137 012062      JMP     24$            ;GO TO 24$ IF ERROR
274 011640                      15$:
275
276
277 011640 012737 011650 001124 ;CHANGE LOOP ADDRESSES
    011646 000410              MOV    #16$,$LPERR
    011646 000410              BR     17$
                                ;SKIP TO NEXT OPERATION
279
280
281
282
283 011650                      ;*****
    011650                      ;LOOP #3      WRITE CHECK DATA
284
285
286 011650 004737 034032      ;PREPARE DEVICE FOR TEST
    011654 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 17$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 24$ IF ERROR
    011656 000404              BR     17$
    011660 000240              NOP
    011662 104000              EMT
    011664 000137 012062      JMP     24$
287 011670                      17$:
288
289
290 011670                      ;SETUP PARAMETERS AND EXECUTE COMMAND
    011670                      18$:
    
```



```

291 011670 012737 000051 001412      MOV      #WCD!GO,RMCS10      ;WRITE CHECK DATA DATA COMMAND
292 011676 012702 001555              MOV      #PUTINX,R2         ;LOAD PUT REGISTER INDEX TABLE
293 011702 112722 000006      MOVB    #PMDA,(R2)+
294 011706 112722 000032      MOVB    #RMOF,(R2)+
295 011712 112722 000034      MOVB    #RMDC,(R2)+
296 011716 112722 000004      MOVB    #RMBA,(R2)+
297 011722 112722 000002      MOVB    #RMWC,(R2)+
298 011726 112722 000000      MOVB    #RMCS1,(R2)+
299 011732 112712 000200      MOVB    #200,(R2)
300
301 011736 004737 040360      JSR     PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011742 000404              BR      19$                 ;GO TO 19$ IF NO ERROR
      011744 000240              NOP
      011746 104000              EMT
      011750 000137 012062      JMP     24$                 ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 24$ IF ERROR
302 011754              19$:
303
304              ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      011754 004737 040024      JSR     PC,GETSTS           ;GO TO GETSTS SUBROUTINE
305
306              ;WAIT FOR COMMAND TO COMPLETE
      011760 004737 040732      JSR     PC,TIMOUT          ;GO TO TIMOUT SUBROUTINE
307
308              ;GO GET REGISTER STATUS
309 011764 004737 040110      JSR     PC,GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011770 000404              BR      20$                 ;GO TO 20$ IF NO ERROR
      011772 000240              NOP                          ;RETURN HERE IF ERROR
      011774 104000              EMT                          ;ERROR # DEFINED BY GET SUBROUTINE
      011776 000137 012062      JMP     24$                 ;GO TO 24$ IF ERROR
310 012002              20$:
311
312              ;VERIFY RESULTS OF WRITE CHECK COMMAND
313 012002 004737 041126      JSR     PC,PRIERR           ;GO CHECK FOR PRIMARY ERRORS
      012006 000405              BR      21$                 ;GO TO 21$ IF NO ERROR
      012010 000240              NOP                          ;RETURN HERE IF ERROR
      012012 104000              EMT                          ;ERROR # DEFINED BY PRIERR SUBROUTINE
      012014 004736              JSR     PC,@(SP)+           ;GO BACK FOR MORE ERROR CHECKS
      012016 000137 012062      JMP     24$                 ;GO TO 24$ IF ERROR
314 012022              21$:
315 012022 004737 054166      JSR     PC,DTASTS           ;GO VERIFY RESULTS OF DATA TRANSFER
      012026 000405              BR      22$                 ;GO TO 22$ IF NO ERROR
      012030 000240              NOP                          ;RETURN HERE IF ERROR
      012032 104000              EMT                          ;ERROR # DEFINED BY DTASTS SUBROUTINE
      012034 004736              JSR     PC,@(SP)+           ;GO BACK FOR MORE ERROR CHECKS
      012036 000137 012062      JMP     24$                 ;GO TO 24$ IF ERROR
316 012042              22$:
317 012042 004737 041760      JSR     PC,SECERR           ;GO CHECK FOR SECONDARY ERRORS
      012046 000405              BR      23$                 ;GO TO 23$ IF NO ERROR
      012050 000240              NOP                          ;RETURN HERE IF ERROR
      012052 104000              EMT                          ;ERROR # DEFINED BY SECERR SUBROUTINE
      012054 004736              JSR     PC,@(SP)+           ;GO BACK FOR MORE ERROR CHECKS
      012056 000137 012062      JMP     24$                 ;GO TO 24$ IF ERROR
318 012062              23$:
319
320 012062              24$:
321
322              ;;*****
    
```

```

        : *TEST 4      WRITE, WRITE CHECK ZEROS W/ WCE ERROR
        : *****
        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,WRITE CHECK DATA
        1$:
        :PREPARE DEVICE FOR TEST
        JSR PC,TSTPRP   :PREPARE DEVICE FOR TEST
        .WORD 154130    :TASK DESCRIPTOR AS FOLLOWS:
                       :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                       :CLEAR CONTROLLER & SELECT DEVICE
                       :VERIFY CONTROLLER CLEAR OPERATION
                       :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                       :VERIFY PACK ACKNOWLEDGE
                       :RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                       :VERIFY RECALIBRATION
                       :GO TO 2$ IF NO ERROR
                       :RETURN HERE IF ERROR
                       :ERROR # DEFINED BY TSTPRP SUBROUTINE
                       :GO TO 28$ IF ERROR

        BR 2$
        NOP
        EMT
        JMP 28$

        :LOAD PARAMETERS AND GENERATE DATA BUFFER
        2$:
        MOV #559.,RMDCO :CYLINDER = 559.
        MOV #TA2,RMDAO  :TRACK = 2, 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,RMCS1O :WRITE HEADER AND DATA COMMAND

        :VERIFY THAT SECTOR IS NOT BAD
        JSR PC,BADSCT   :CALL BAD SECTOR MODULE
        BR 3$          :GO TO 3$ IF NO ERROR
        TYPE .SCTMSG   :TYPE BAD SECTOR MESSAGE
        EMT
        JMP 28$       :ERROR # DEFINED BY BADSCT SUBROUTINE
                       :GO TO 28$ IF ERROR

        3$:
        MOV #ZEROS,$TMP0 :STARTING ADDRESS OF PATTERN
        MOV #1,$TMP1     :RANGE OF PATTERN
        JSR PC,GENBUF    :GO GENERATE BUFFER FOR FORMAT

        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)+

012062 000004
012062 000240
012064 000240
012066 012706 001100
012072 013700 001276
012076 013701 001466
012102 012737 000004 001226

323
324
325
326
327 012110
328
329
330 012110 004737 034032
    012114 154130

    012116 000404
    012120 000240
    012122 104000
    012124 000137 013256

331
332
333 012130
334 012130 012737 001057 001446
335 012136 012737 001000 001420
336 012144 012737 104254 001416
337 012152 012737 177376 001414
338 012160 012737 010000 001444
339 012166 012737 000063 001412
340
341
    012174 004737 034762
    012200 000405
    012202 104401 066006
    012206 104000
    012210 000137 013256

342 012214
343 012214 012737 067456 001174
344 012222 012737 000001 001176
345 012230 004737 037214
346
347 012234 012702 001555
348 012240 112722 000034
349 012244 112722 000006
350 012250 112722 000004
351 012254 112722 000002
352 012260 112722 000032
353 012264 112722 000000
    
```



```

354 012270 112712 000200          MOVB   #200,(R2)          ;TERMINATE TABLE
355 012274                                     4$:
356 012274 004737 040360          JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    012300 000404                                     BR     5$              ;GO TO 5$ IF NO ERROR
    012302 000240                                     NOP                                ;RETURN HERE IF ERROR
    012304 104000                                     EMT                                ;ERROR # DEFINED BY PUT SUBROUTINE
    012306 000137 013256          JMP    28$            ;GO TO 28$ IF ERROR
357 012312                                     5$:
358
359                                     ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
    012312 004737 040024          JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
360
361                                     ;WAIT FOR COMMAND TO COMPLETE
    012316 004737 040732          JSR    PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
362
363                                     ;GO EGT REGISTER STATUS
364 012322 004737 040110          JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
    012326 000404                                     BR     6$              ;GO TO 6$ IF NO ERROR
    012330 000240                                     NOP                                ;RETURN HERE IF ERROR
    012332 104000                                     EMT                                ;ERROR # DEFINED BY GET SUBROUTINE
    012334 000137 013256          JMP    28$            ;GO TO 28$ IF ERROR
365 012340                                     6$:
366
367                                     ;VERIFY RESULTS OF WRITE COMMAND
368 012340 004737 054166          JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
    012344 000405                                     BR     7$              ;GO TO 7$ IF NO ERROR
    012346 000240                                     NOP                                ;RETURN HERE IF ERROR
    012350 104000                                     EMT                                ;ERROR # DEFINED BY DTASTS SUBROUTINE
    012352 004736                                     JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    012354 000137 013256          JMP    28$            ;GO TO 28$ IF ERROR
369 012360                                     7$:
370
371                                     ;MOVE LOOP ADDRESSES TO NEXT OPERATION
372 012360 012737 012370 001124    MOV    #8$,SLPERR
373 012366 000410                                     BR     9$              ;SKIP TO WRITE OPERATION
374
375                                     ;*****
376                                     ;LOOP #2      WRITE,WRITE CHECK DATA
377
378 012370                                     8$:
379
380                                     ;PREPARE DEVICE FOR TEST
381 012370 004737 034032          JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
    012374 154130          .WORD 154130         ;TASK DESCRIPTOR AS FOLLOWS:
    012376 000404                                     BR     9$              ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
    012400 000240                                     NOP                                ;CLEAR CONTROLLER & SELECT DEVICE
    012402 104000                                     EMT                                ;VERIFY CONTROLLER CLEAR OPERATION
    012404 000137 013256          JMP    28$            ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    012376 000404                                     BR     9$              ;VERIFY PACK ACKNOWLEDGE
    012400 000240                                     NOP                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
    012402 104000                                     EMT                                ;VERIFY RECALIBRATION
    012404 000137 013256          JMP    28$            ;GO TO 9$ IF NO ERROR
    012410 000137 013256          JMP    28$            ;RETURN HERE IF ERROR
    012410 000137 013256          JMP    28$            ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    012410 000137 013256          JMP    28$            ;GO TO 28$ IF ERROR
382
383                                     9$:

```



```

384                                     :SETUP PARAMETERS AND EXECUTE COMMAND
385 012410                               10$:
386 012410 012737 177400 001414      MOV    #-256.,RMWCO      ;CHANGE WORD COUNT
387 012416 012737 104260 001416      MOV    #BUFONE+4,RMBAO ;CHANGE MEMORY ADDRESS
388 012424 012737 000061 001412      MOV    #WD!GO,RMCS10  ;WRITE DATA COMMAND
389 012432 012702 001555                MOV    #PUTINX,R2     ;LOAD PUT REGISTER INDEX TABLE
390 012436 112722 000006                MOVVB  #RMDA,(R2)+
391 012442 112722 000034                MOVVB  #RMDC,(R2)+
392 012446 112722 000032                MOVVB  #RMOF,(R2)+
393 012452 112722 000004                MOVVB  #RMBA,(R2)+
394 012456 112722 000002                MOVVB  #RMWC,(R2)+
395 012462 112722 000000                MOVVB  #RMCS1,(R2)+
396 012466 112722 000200                MOVVB  #200,(R2)+      ;TERMINATE TABLE
397
398 012472 004737 040360                JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    012476 000404                BR     11$           ;GO TO 11$ IF NO ERROR
    012500 000240                NOP                    ;RETURN HERE IF ERROR
    012502 104000                EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
    012504 000137 013256                JMP    28$           ;GO TO 28$ IF ERROR
399 012510                               11$:
400
401                                     :SETUP GET INDEX TABLE TO READ ALL REGISTERS
    012510 004737 040024                JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
402
403                                     :WAIT FOR COMMAND TO COMPLETE
    012514 004737 040732                JSR    PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
404
405                                     :GO READ STATUS FOR WRITE COMMAND
406 012520 004737 040110                JSR    PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
    012524 000404                BR     12$           ;GO TO 12$ IF NO ERROR
    012526 000240                NOP                    ;RETURN HERE IF ERROR
    012530 104000                EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
    012532 000137 013256                JMP    28$           ;GO TO 28$ IF ERROR
407 012536                               12$:
408
409                                     :CHECK FOR ERRORS DURING WRITE OPERATION
410 012536 004737 041126                JSR    PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
    012542 000405                BR     13$           ;GO TO 13$ IF NO ERROR
    012544 000240                NOP                    ;RETURN HERE IF ERROR
    012546 104000                EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
    012550 004736                JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
    012552 000137 013256                JMP    28$           ;GO TO 28$ IF ERROR
411 012556                               13$:
412 012556 004737 054166                JSR    PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
    012562 000405                BR     14$           ;GO TO 14$ IF NO ERROR
    012564 000240                NOP                    ;RETURN HERE IF ERROR
    012566 104000                EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
    012570 004736                JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
    012572 000137 013256                JMP    28$           ;GO TO 28$ IF ERROR
413 012576                               14$:
414 012576 004737 041760                JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
    012602 000405                BR     15$           ;GO TO 15$ IF NO ERROR
    012604 000240                NOP                    ;RETURN HERE IF ERROR
    012606 104000                EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
    012610 004736                JSR    PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
    012612 000137 013256                JMP    28$           ;GO TO 28$ IF ERROR
415 012616                               15$:
    
```

```

416
417
418 012616 012737 012630 001124 ;CHANGE LOOP ADDRESSES
419 MOV #16$, $LPERR
420 ;*****
421 ;LOOP #3 WRITE CHECK DATA
422
423 012624 012703 000001 16$: MOV #1, R3 ;R3+WCE BIT POSITION
424 012630 BIS R3, BUFTWO-2 ;CHANGE LAST WORD OF BUFFER
425 012630 050337 105256
426
427 ;PREPARE DEVICE FOR TEST
428 012634 004737 034032 JSR PC, TSTPRP ;PREPARE DEVICE FOR TEST
012640 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 17$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 28$ IF ERROR

012642 000404 BR 17$
012644 000240 NOP
012646 104000 EMT
012650 000137 013256 JMP 28$
429 012654 17$:
430
431 ;READ DATA FROM DEVICE
432 18$:
433 012654 012737 000051 001412 MOV #WCD!GO, RMCS10 ;WRITE CHECK DATA DATA COMMAND
434 012662 012702 001555 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
435 012666 112722 000006 MOVB #RMDA, (R2)+
436 012672 112722 000032 MOVB #RMOF, (R2)+
437 012676 112722 000034 MOVB #RMDC, (R2)+
438 012702 112722 000004 MOVB #RMBA, (R2)+
439 012706 112722 000002 MOVB #RMWC, (R2)+
440 012712 112722 000000 MOVB #RMCS1, (R2)+
441 012716 112712 000200 MOVB #200, (R2)
442
443 012722 004737 040360 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
012726 000404 BR 19$ ;GO TO 19$ IF NO ERROR
012730 000240 NOP ;RETURN HERE IF ERROR
012732 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
012734 000137 013256 JMP 28$ ;GO TO 28$ IF ERROR
444 012740 19$:
445
446 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
012740 004737 040024 JSR PC, GETSTS ;GO TO GETSTS SUBROUTINE
447
448 ;WAIT FOR COMMAND TO COMPLETE
012744 004737 040732 JSR PC, TIMEOUT ;GO TO TIMEOUT SUBROUTINE
449
450 ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
451 012750 004737 040110 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
012754 000404 BR 20$ ;GO TO 20$ IF NO ERROR
012756 000240 NOP ;RETURN HERE IF ERROR
012760 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE

```



```

012762 000137 013256          JMP      28$          ;GO TO 28$ IF ERROR
452 012766          20$:
453
454          ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
455 012766 004737 041126          JSR      PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
                                BR          21$          ;GO TO 21$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
                                JMP      28$          ;GO TO 28$ IF ERROR
456 013006          21$:
457 013006 032737 040000 001346          BIT      #WCE,RMCS2I  ;WAS 'WCE' DETECTED??
458 013014 001030          BNE          23$          ;YES!!
459
460 013016 004737 054166          JSR      PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR          22$          ;GO TO 22$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
                                JMP      28$          ;GO TO 28$ IF ERROR
461 013036          22$:
462 013036 013737 001346 001140          MOV      RMCS2I,$GDDAT ;EXPECTED STATUS
463 013044 052737 040000 001140          BIS      #WCE,$GDDAT
464 013052 013737 001346 001142          MOV      RMCS2I,$BDDAT ;RECEIVED STATUS
465 013060 010337 001174          MOV      R3,$STMP0    ;FAILING BIT POSITION
466 013064 012737 105256 001176          MOV      #BUFTWO-2,$STMP1 ;FAILING ADDRESS
467 013072 104337          EMT      337
468 013074 000470          BR          28$
469 013076          23$:
470 013076 112737 000022 001526          MOVB    #RMDB,GETINX  ;SETUP GET INDEX TABLE
                                MOVB    #200,GETINX+1 ;SETUP TERMINATOR BYTE
                                JSR      PC,GET      ;GO READ RMDB VIA GET SUBROUTINE
                                BR          24$          ;GO TO 24$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR DEFINED BY GET SUBROUTINE
                                JMP      28$          ;GO TO 28$ IF ERROR
471 013130          24$:
472 013130 013737 001360 001142          MOV      RMDBI,$BDDAT ;RECEIVED DATA
473 013136 013737 105256 001140          MOV      BUFTWO-2,$GDDAT ;EXPECTED DATA
474 013144 040337 001140          BIC      R3,$GDDAT
475 013150 012737 105256 001134          MOV      #BUFTWO-2,$GDADR ;EXPECTED ADDRESS
476 013156 013737 001342 001136          MOV      RMBAI,$BDADR ;RECEIVED ADDRESS
477 013164 162737 000002 001136          SUB      #2,$BDADR    ;ADJUST BUS ADDRESS
478 013172 023737 001134 001136          CMP      $GDADR,$BDADR ;ADDRESSES OK??
479 013200 001402          BEQ      25$          ;YES!!
480 013202 104340          EMT      340
481 013204 000424          BR          28$
482 013206 023737 001140 001142 25$:          CMP      $GDDAT,$BDDAT ;DATA OK??
483 013214 001402          BEQ      26$          ;YES!!
484 013216 104341          EMT      341
485 013220 000416          BR          28$
486 013222          26$:
487
488 013222 004737 041760          JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
                                BR          27$          ;GO TO 27$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
    
```



```

013234 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
013236 000137 013256 JMP 28$ ;GO TO 28$ IF ERROR
489 013242 27$: BIC R3,BUFTWO-2 ;RESTORE DATA PATTERN
490 013242 040337 105256 ASL R3 ;SHIFT TO NEXT BIT
491 013246 006303 BEQ 28$ ;EXIT IF DONE
492 013250 001402 JMP 16$ ;REPEAT TEST FOR NEXT DATA BIT
493 013252 000137 012630
494 013256
495
496
*****
*TEST 5 WRITE, READ ONES
*****
TST5:
013256 000004 SCOPE ;SCOPE CALL
013260 000240 NOP ;START OF TEST
013262 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
013266 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
013272 013701 001466 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
013276 012737 000005 001226 MOV #5,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
497
498
499
500
501 013304
502
503
504 013304 004737 034032 ;PREPARE THE DEVICE FOR TEST
013310 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 2$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 25$ IF ERROR
013312 000404 BR 2$
013314 000240 NOP
013316 104000 EMT
013320 000137 014264 JMP 25$
505
506 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
507 013324 2$: MOV #559.,RMDCO ;CYLINDER = 559.
013324 012737 001057 001446 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
013332 012737 001000 001420 MOV #BUFOE,RMBAO ;BUS ADDRESS
013340 012737 104254 001416 MOV #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
013346 012737 177376 001414 MOV #FMT16,RMOFO ;16 BIT FORMAT
013354 012737 010000 001444 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
013362 012737 000063 001412
514
515 ;VERIFY THAT SECTOR IS NOT BAD
013370 004737 034762 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
013374 000405 BR 3$ ;GO TO 3$ IF NO ERROR
013376 104401 066006 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
013402 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
013404 000137 014264 JMP 25$ ;GO TO 25$ IF ERROR
516 013410
517 013410 012737 067414 001174 MOV #ONES,$TMP0 ;STARTING ADDRESS OF PATTERN
    
```

```

518 013416 012737 000001 001176      MOV    #1,$STMP1      ;RANGE OF PATTERN
519 013424 004737 037214              JSR    PC,GENBUF     ;GO GENERATE BUFFER FOR FORMAT
520
521      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
522 013430 012702 001555      MOV    #PUTINX,R2    ;R2 = BYTE ENTRY POSITION
523 013434 112722 000034      MOVB  #RMDC,(R2)+
524 013440 112722 000006      MOVB  #RMDA,(R2)+
525 013444 112722 000004      MOVB  #RMBA,(R2)+
526 013450 112722 000002      MOVB  #RMWC,(R2)+
527 013454 112722 000032      MOVB  #RMOF,(R2)+
528 013460 112722 000000      MOVB  #RMCS1,(R2)+
529 013464 112712 000200      MOVB  #200,(R2)     ;TERMINATE TABLE
530 013470
531
532      ;FORMAT THE DRIVE
533 013470 004737 040360      JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    013474 000404      BR     5$           ;GO TO 5$ IF NO ERROR
    013476 000240      NOP                    ;RETURN HERE IF ERROR
    013500 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
    013502 000137 014264      JMP    25$          ;GO TO 25$ IF ERROR
534 013506
535
536      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
    013506 004737 040024      JSR    PC,GETSTS    ;GO TO GETSTS SUBROUTINE
537
538      ;WAIT FOR COMMAND TO COMPLETE
    013512 004737 040732      JSR    PC,TIMOUT    ;GO TO TIMOUT SUBROUTINE
539
540      ;GO READ STATUS FOR FORMAT OPERATION
541 013516 004737 040110      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
    013522 000404      BR     6$           ;GO TO 6$ IF NO ERROR
    013524 000240      NOP                    ;RETURN HERE IF ERROR
    013526 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
    013530 000137 014264      JMP    25$          ;GO TO 25$ IF ERROR
542 013534
543
544      ;VERIFY NO ERRORS DURING FORMAT
545 013534 004737 054166      JSR    PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
    013540 000405      BR     7$           ;GO TO 7$ IF NO ERROR
    013542 000240      NOP                    ;RETURN HERE IF ERROR
    013544 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
    013546 004736      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
    013550 000137 014264      JMP    25$          ;GO TO 25$ IF ERROR
546 013554
547
548      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
549 013554 012737 013564 001124      MOV    #8$,$LPERR
550 013562 000410      BR     9$           ;SKIP TO WRITE OPERATION
551
552      ;*****
553      ;LOOP #2      WRITE,READ
554
555 013564
556
557
558      ;PREPARE DEVICE FOR TEST
559 013564 004737 034032      JSR    PC,TSTPRP   ;PREPARE DEVICE FOR TEST
    
```


013570	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 9\$ IF NO ERROR
									:RETURN HERE IF ERROR
									:ERROR # DEFINED BY TSTPRP SUBROUTINE
									:GO TO 25\$ IF ERROR
560	013604					9\$:			
561									
562									
563	013604								
564	013604	012737	104260	001416					
565	013612	012737	177400	001414					
566	013620	012737	000061	001412					
567	013626	012702	001555						
568	013632	112722	000006						
569	013636	112722	000034						
570	013642	112722	000032						
571	013646	112722	000004						
572	013652	112722	000002						
573	013656	112722	000000						
574	013662	112722	000200						
575									
576	013666	004737	040360						
	013672	000404							
	013674	000240							
	013676	104000							
	013700	000137	014264						
577	013704								
578									
579									
	013704	004737	040024						
580									
581									
	013710	004737	040732						
582									
583									
	013714	004737	040110						
	013720	000404							
	013722	000240							
	013724	104000							
	013726	000137	014264						
585	013732								
586									
587									
	013732	004737	041126						
	013736	000405							
	013740	000240							
	013742	104000							
	013744	004736							
	013746	000137	014264						
589	013752								

```

:WRITE DATA TO THE DRIVE
10$:
MOV #BUFONE+4,RMBAD ;CHANGE MEMORY ADDRESS
MOV #-256,RMWC0 ;CHANGE WORD COUNT
MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMDF,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+ ;TERMINATE TABLE

JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 11$ ;GO TO 11$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 25$ ;GO TO 25$ IF ERROR

11$:
;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 TIMEOUT SUBROUTINE

;GO READ STATUS FOR WRITE COMMAND
JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
BR 12$ ;GO TO 12$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY GET SUBROUTINE
JMP 25$ ;GO TO 25$ IF ERROR

12$:
;CHECK FOR ERRORS DURING WRITE OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 13$ ;GO TO 13$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 25$ ;GO TO 25$ IF ERROR

13$:

```



```

622 014130 000137 014264          JMP      25$          :GO TO 25$ IF ERROR
623 014134          19$:
624          :SETUP GET INDEX TABLE TO READ ALL REGISTERS
        JSR      PC,GETSTS      :GO TO GETSTS SUBROUTINE
625          :WAIT FOR COMMAND TO COMPLETE
        JSR      PC,TIMOUT      :GO TO TIMOUT SUBROUTINE
626 014140 004737 040732
627          :GO READ STATUS FOR READ OPERATION
628          JSR      PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
629 014144 004737 040110          BR      20$          :GO TO 20$ IF NO ERROR
        014150 000404          NOP          :RETURN HERE IF ERROR
        014152 000240          EMT          :ERROR # DEFINED BY GET SUBROUTINE
        014154 104000          JMP      25$          :GO TO 25$ IF ERROR
        014156 000137 014264
630 014162          20$:
631          :CHECK FOR ERRORS DURING READ OPERATION
632          JSR      PC,PRIERR      :GO CHECK FOR PRIMARY ERRORS
633 014162 004737 041126          BR      21$          :GO TO 21$ IF NO ERROR
        014166 000405          NOP          :RETURN HERE IF ERROR
        014170 000240          EMT          :ERROR # DEFINED BY PRIERR SUBROUTINE
        014172 104000          JSR      PC,@(SP)+      :GO BACK FOR MORE ERROR CHECKS
        014174 004736          JMP      25$          :GO TO 25$ IF ERROR
        014176 000137 014264
634 014202          21$:
635 014202 004737 054166          JSR      PC,DTASTS      :GO VERIFY RESULTS OF DATA TRANSFER
        014206 000405          BR      22$          :GO TO 22$ IF NO ERROR
        014210 000240          NOP          :RETURN HERE IF ERROR
        014212 104000          EMT          :ERROR # DEFINED BY DTASTS SUBROUTINE
        014214 004736          JSR      PC,@(SP)+      :GO BACK FOR MORE ERROR CHECKS
        014216 000137 014264          JMP      25$          :GO TO 25$ IF ERROR
636 014222          22$:
637 014222 004737 041760          JSR      PC,SECERR      :GO CHECK FOR SECONDARY ERRORS
        014226 000405          BR      23$          :GO TO 23$ IF NO ERROR
        014230 000240          NOP          :RETURN HERE IF ERROR
        014232 104000          EMT          :ERROR # DEFINED BY SECERR SUBROUTINE
        014234 004736          JSR      PC,@(SP)+      :GO BACK FOR MORE ERROR CHECKS
        014236 000137 014264          JMP      25$          :GO TO 25$ IF ERROR
638 014242          23$:
639 014242 004737 037462          JSR      PC,CMPBUF      :GO COMPARE WRITE, READ DATA BUFFERS
        014246 104260          .WORD      BUFOFF+4      :STARTING ADDRESS OF WRITE BUFFER
        014250 105264          .WORD      BUFTWO+4      :STARTING ADDRESS OF READ BUFFER
        014252 000404          BR      24$          :GO TO 24$ IF NO ERROR
        014254 000240          NOP          :RETURN HERE IF ERROR
        014256 104000          EMT          :ERROR # DEFINED BY CMPBUF SUBROUTINE
        014260 000137 014264          JMP      25$          :GO TO 25$ IF ERROR
640 014264          24$:
641          25$:
642 014264
643
644
        :*****
        :*TEST 6      WRITE, WRITE CHECK ONES
        :*****
TST6:
        SCOPE          :SCOPE CALL
        NOP           :START OF TEST
        MOV      #STACK,SP      :INITIALIZE STACK POINTER
    
```



```

014274 013700 001276      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
014300 013701 001466      MOV      TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
014304 012737 000006 001226  MOV      #6,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

645
646      ;:*****
647      ;:LOOP #1          FORMAT,WRITE,WRITE CHECK DATA
648
649 014312      1$:
650
651      ;:PREPARE THE DEVICE FOR TEST
652 014312 004737 034032      JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
014316 154130      .WORD   154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                           ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                           ;CLEAR CONTROLLER & SELECT DEVICE
                                           ;VERIFY CONTROLLER CLEAR OPERATION
                                           ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                           ;VERIFY PACK ACKNOWLEDGE
                                           ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                           ;VERIFY RECALIBRATION
                                           ;GO TO 2$ IF NO ERROR
                                           ;RETURN HERE IF ERROR
                                           ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                           ;GO TO 24$ IF ERROR

014320 000404      BR       2$
014322 000240      NOP
014324 104000      EMT
014326 000137 015242      JMP      24$

653
654      ;:LOAD PARAMETERS AND GENERATE DATA BUFFER
655 014332      2$:
656 014332 012737 001057 001446      MOV      #559.,RMDCO   ;CYLINDER = 559.
657 014340 012737 001000 001420      MOV      #TA2,RMDAO   ;TRACK = 2, SECTOR = 0
658 014346 012737 104254 001416      MOV      #BUFONE,RMBAO ;BUS ADDRESS
659 014354 012737 177376 001414      MOV      #-258.,RMWCO  ;2 + 256. WORDS (2'S COMP)
660 014362 012737 010000 001444      MOV      #FMT16,RMFO  ;16 BIT FORMAT
661 014370 012737 000063 001412      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
662
663      ;:VERIFY THAT SECTOR IS NOT BAD
014376 004737 034762      JSR      PC,BADSCT    ;CALL BAD SECTOR MODULE
014402 000405      BR       3$          ;GO TO 3$ IF NO ERROR
014404 104401 066006      TYPE     .SCTMSG     ;TYPE BAD SECTOR MESSAGE
014410 104000      EMT
014412 000137 015242      JMP      24$          ;ERROR # DEFINED BY BADSCT SUBROUTINE
                                           ;GO TO 24$ IF ERROR

664 014416      3$:
665 014416 012737 067414 001174      MOV      #ONES,$TMP0   ;STARTING ADDRESS OF PATTERN
666 014424 012737 000001 001176      MOV      #1,$TMP1     ;RANGE OF PATTERN
667 014432 004737 037214      JSR      PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
668
669      ;:LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
670 014436 012702 001555      MOV      #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
671 014442 112722 000034      MOVB    #RMDC,(R2)+
672 014446 112722 000006      MOVB    #RMDA,(R2)+
673 014452 112722 000004      MOVB    #RMBA,(R2)+
674 014456 112722 000002      MOVB    #RMWC,(R2)+
675 014462 112722 000032      MOVB    #RMOF,(R2)+
676 014466 112722 000000      MOVB    #RMCS1,(R2)+
677 014472 112712 000200      MOVB    #200,(R2)    ;TERMINATE TABLE
678 014476
679
680      ;:FORMAT THE DRIVE
681 014476 004737 040360      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    
```



```

014502 000404 BR 5$ ;GO TO 5$ IF NO ERROR
014504 000240 NOP ;RETURN HERE IF ERROR
014506 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
014510 000137 015242 JMP 24$ ;GO TO 24$ IF ERROR
682 014514 5$:
683
684 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
014514 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
685
686 ;WAIT FOR COMMAND TO COMPLETE
014520 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
687
688 ;GO READ STATUS FOR FORMAT OPERATION
689 014524 004737 040110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
014530 000404 BR 6$ ;GO TO 6$ IF NO ERROR
014532 000240 NOP ;RETURN HERE IF ERROR
014534 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
014536 000137 015242 JMP 24$ ;GO TO 24$ IF ERROR
690 014542 6$:
691
692 ;VERIFY NO ERRORS DURING FORMAT
693 014542 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
014546 000405 BR 7$ ;GO TO 7$ IF NO ERROR
014550 000240 NOP ;RETURN HERE IF ERROR
014552 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
014554 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
014556 000137 015242 JMP 24$ ;GO TO 24$ IF ERROR
694 014562 7$:
695
696 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
697 014562 012737 014572 001124 MOV #8$,SLPERR
698 014570 000410 BR 9$ ;SKIP TO WRITE OPERATION
699
700 ;*****
701 ;LOOP #2 WRITE,WRITE CHECK DATA
702
703 014572 8$:
704
705 ;PREPARE DEVICE FOR WRITE OPERATION
706
707 014572 004737 034032 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
014576 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
014600 000404 BR 9$ ;GO TO 9$ IF NO ERROR
014602 000240 NOP ;RETURN HERE IF ERROR
014604 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
014606 000137 015242 JMP 24$ ;GO TO 24$ IF ERROR
708 014612 9$:
709
710 ;WRITE DATA TO THE DRIVE
711 014612 10$:
    
```

```

712 014612 012737 104260 001416      MOV    #BUFONE+4,RMBA0      ;CHANGE MEMORY ADDRESS
713 014620 012737 177400 001414      MOV    #-256.,RMWCO        ;CHANGE WORD COUNT
714 014626 012737 000061 001412      MOV    #WD!GO,RMCS10      ;WRITE DATA COMMAND
715 014634 012702 001555          MOV    #PUTINX,R2         ;LOAD PUT REGISTER INDEX TABLE
716 014640 112722 000006      MOVSB #RMDA,(R2)+
717 014644 112722 000034      MOVSB #RMDC,(R2)+
718 014650 112722 000032      MOVSB #RMOF,(R2)+
719 014654 112722 000004      MOVSB #RMBA,(R2)+
720 014660 112722 000002      MOVSB #RMWC,(R2)+
721 014664 112722 000000      MOVSB #RMCS1,(R2)+
722 014670 112722 000200      MOVSB #200,(R2)+          ;TERMINATE TABLE
723
724 014674 004737 040360          JSR    PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    014700 000404          BR     11$                ;GO TO 11$ IF NO ERROR
    014702 000240          NOP
    014704 104000          EMT
    014706 000137 015242      JMP    24$                ;ERROR # DEFINED BY PUT SUBROUTINE
    ;GO TO 24$ IF ERROR
725 014712          11$:
726
727          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
    014712 004737 040024      JSR    PC,GETSTS          ;GO TO GETSTS SUBROUTINE
728
729          ;WAIT FOR COMMAND TO COMPLETE
    014716 004737 040732      JSR    PC,TIMOUT         ;GO TO TIMOUT SUBROUTINE
730
731          ;GO READ STATUS FOR WRITE COMMAND
732 014722 004737 040110      JSR    PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014726 000404          BR     12$                ;GO TO 12$ IF NO ERROR
    014730 000240          NOP
    014732 104000          EMT
    014734 000137 015242      JMP    24$                ;ERROR # DEFINED BY GET SUBROUTINE
    ;GO TO 24$ IF ERROR
733 014740          12$:
734
735          ;CHECK FOR ERRORS DURING WRITE OPERATION
736 014740 004737 041126      JSR    PC,PRIERR         ;GO CHECK FOR PRIMARY ERRORS
    014744 000405          BR     13$                ;GO TO 13$ IF NO ERROR
    014746 000240          NOP
    014750 104000          EMT
    014752 004736          JSR    PC,@(SP)+         ;GO BACK FOR MORE ERROR CHECKS
    014754 000137 015242      JMP    24$                ;GO TO 24$ IF ERROR
737 014760          13$:
738 014760 004737 054166      JSR    PC,DTASTS         ;GO VERIFY RESULTS OF DATA TRANSFER
    014764 000405          BR     14$                ;GO TO 14$ IF NO ERROR
    014766 000240          NOP
    014770 104000          EMT
    014772 004736          JSR    PC,@(SP)+         ;GO BACK FOR MORE ERROR CHECKS
    014774 000137 015242      JMP    24$                ;GO TO 24$ IF ERROR
739 015000          14$:
740 015000 004737 041760      JSR    PC,SECERR         ;GO CHECK FOR SECONDARY ERRORS
    015004 000405          BR     15$                ;GO TO 15$ IF NO ERROR
    015006 000240          NOP
    015010 104000          EMT
    015012 004736          JSR    PC,@(SP)+         ;GO BACK FOR MORE ERROR CHECKS
    015014 000137 015242      JMP    24$                ;GO TO 24$ IF ERROR
741 015020          15$:
742
743          ;CHANGE LOOP ADDRESSES
    
```



```

744 015020 012737 015030 001124      MOV      #16$,SLPERR
745 015026 000410                      BR        17$                :SKIP TO NEXT OPERATION
746
747
748      :*****
749      :LOOP #3      WRITE CHECK DATA
750 015030      16$:
751
752      :PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
753 015030 004737 034032      JSR      PC,TSTPRP      :PREPARE DEVICE FOR TEST
015034 154130      .WORD 154130      :TASK DESCRIPTOR AS FOLLOWS:
                                :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                :CLEAR CONTROLLER & SELECT DEVICE
                                :VERIFY CONTROLLER CLEAR OPERATION
                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                :VERIFY PACK ACKNOWLEDGE
                                :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                :VERIFY RECALIBRATION
                                :GO TO 17$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 24$ IF ERROR
                                :
015036 000404      BR        17$
015040 000240      NOP
015042 104000      EMT
015044 000137 015242      JMP      24$
754 015050      17$:
755
756      :WRITE CHECK DATA DATA FROM DEVICE
757 015050      18$:
758 015050 012737 000051 001412      MOV      #WCD!GO, RMCS10      :WRITE CHECK DATA DATA COMMAND
759 015056 012702 001555      MOV      #PUTINX,R2          :LOAD PUT REGISTER INDEX TABLE
760 015062 112722 000006      MOV      #RMDA,(R2)+
761 015066 112722 000032      MOV      #RMOF,(R2)+
762 015072 112722 000034      MOV      #RMDC,(R2)+
763 015076 112722 000004      MOV      #RMBA,(R2)+
764 015102 112722 000002      MOV      #RMWC,(R2)+
765 015106 112722 000000      MOV      #RMCS1,(R2)+
766 015112 112712 000200      MOV      #200,(R2)
767
768 015116 004737 040360      JSR      PC,PUT              :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
015122 000404      BR        19$                :GO TO 19$ IF NO ERROR
015124 000240      NOP                          :RETURN HERE IF ERROR
015126 104000      EMT                          :ERROR # DEFINED BY PUT SUBROUTINE
015130 000137 015242      JMP      24$                :GO TO 24$ IF ERROR
769 015134      19$:
770
771      :SETUP GET INDEX TABLE TO READ ALL REGISTERS
015134 004737 040024      JSR      PC,GETSTS          :GO TO GETSTS SUBROUTINE
772
773      :WAIT FOR COMMAND TO COMPLETE
015140 004737 040732      JSR      PC,TIMOUT         :GO TO TIMOUT SUBROUTINE
774
775      :GO READ STATUS FOR WRITE CHECK DATA OPERATION
776 015144 004737 040110      JSR      PC,GET            :GO READ REGISTER(S) WITH GET SUBROUTINE
015150 000404      BR        20$                :GO TO 20$ IF NO ERROR
015152 000240      NOP                          :RETURN HERE IF ERROR
015154 104000      EMT                          :ERROR # DEFINED BY GET SUBROUTINE
015156 000137 015242      JMP      24$                :GO TO 24$ IF ERROR
777 015162      20$:
778
    
```



```

779                                     ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
780 015162 004737 041126                JSR   PC,PRIERR                :GO CHECK FOR PRIMARY ERRORS
    015166 000405                        BR    21$                      :GO TO 21$ IF NO ERROR
    015170 000240                        NOP                               :RETURN HERE IF ERROR
    015172 104000                        EMT                               :ERROR # DEFINED BY PRIERR SUBROUTINE
    015174 004736                        JSR   PC,@(SP)+                :GO BACK FOR MORE ERROR CHECKS
    015176 000137 015242                JMP   24$                      :GO TO 24$ IF ERROR

781 015202                                21$:
782 015202 004737 054166                JSR   PC,DTASTS                :GO VERIFY RESULTS OF DATA TRANSFER
    015206 000405                        BR    22$                      :GO TO 22$ IF NO ERROR
    015210 000240                        NOP                               :RETURN HERE IF ERROR
    015212 104000                        EMT                               :ERROR # DEFINED BY DTASTS SUBROUTINE
    015214 004736                        JSR   PC,@(SP)+                :GO BACK FOR MORE ERROR CHECKS
    015216 000137 015242                JMP   24$                      :GO TO 24$ IF ERROR

783 015222                                22$:
784 015222 004737 041760                JSR   PC,SECERR                :GO CHECK FOR SECONDARY ERRORS
    015226 000405                        BR    23$                      :GO TO 23$ IF NO ERROR
    015230 000240                        NOP                               :RETURN HERE IF ERROR
    015232 104000                        EMT                               :ERROR # DEFINED BY SECERR SUBROUTINE
    015234 004736                        JSR   PC,@(SP)+                :GO BACK FOR MORE ERROR CHECKS
    015236 000137 015242                JMP   24$                      :GO TO 24$ IF ERROR

785 015242                                23$:
786
787 015242                                24$:
788
789
:*****
:*TEST 7          WRITE, WRITE CHECK ONES W/ WCE ERROR
:*****
TST7:
    015242                                SCOPE                          :SCOPE CALL
    015242 000004                        NOP                          :START OF TEST
    015244 000240                        MOV   #STACK,SP              :INITIALIZE STACK POINTER
    015246 012706 001100                MOV   $BASE,R0               :R0 = UNIBUS ADDRESS
    015252 013700 001276                MOV   TSTQUE,R1              :(R1) = DEVICE BEING TESTED
    015256 013701 001466                MOV   #7,$TESTN              ;;SET TEST NUMBER IN APT MAIL BOX
    015262 012737 000007 001226

790
791
792
793
794 015270                                :*****
795
796
797 015270 004737 034032                :LOOP #1          FORMAT,WRITE,WRITE CHECK DATA
    015274 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
    BR    2$                          :GO TO 2$ IF NO ERROR
    NOP                               :RETURN HERE IF ERROR
    EMT                               :ERROR # DEFINED BY TSTPRP SUBROUTINE
    JMP   28$                          :GO TO 28$ IF ERROR

    015276 000404                        BR    2$
    015300 000240                        NOP
    015302 104000                        EMT
    015304 000137 016440                JMP   28$

798 015310                                2$:
799
    
```

```
800 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
801 015310 012737 001057 001446 MOV #559.,RMDCO ;CYLINDER = 559.
802 015316 012737 001000 001420 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
803 015324 012737 104254 001416 MOV #BUFONE,RMBAO ;BUS ADDRESS
804 015332 012737 177376 001414 MOV #-258.,RMWCO ;WORD COUNT = 1 SECTOR
805 015340 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
806 015346 012737 000063 001412 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
807
808 ;VERIFY THAT SECTOR IS NOT BAD
      015354 004737 034762 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
      015360 000405 BR 3$ ;GO TO 3$ IF NO ERROR
      015362 104401 066006 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
      015366 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
      015370 000137 016440 JMP 28$ ;GO TO 28$ IF ERROR
809 015374 3$:
810 015374 012737 067414 001174 MOV #ONES,$TMP0 ;STARTING ADDRESS OF PATTERN
811 015402 012737 000001 001176 MOV #1,$TMP1 ;RANGE OF PATTERN
812 015410 004737 037214 JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT
813
814 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
      015414 012702 001555 MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
816 015420 112722 000034 MOVB #RMDC,(R2)+
817 015424 112722 000006 MOVB #RMDA,(R2)+
818 015430 112722 000004 MOVB #RMBA,(R2)+
819 015434 112722 000002 MOVB #RMWC,(R2)+
820 015440 112722 000032 MOVB #RMOF,(R2)+
821 015444 112722 000000 MOVB #RMCS1,(R2)+
822 015450 112712 000200 MOVB #200,(R2) ;TERMINATE TABLE
823 015454 4$:
824
825 ;FORMAT THE DRIVE
826 015454 004737 040360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      015460 000404 BR 5$ ;GO TO 5$ IF NO ERROR
      015462 000240 NOP ;RETURN HERE IF ERROR
      015464 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
      015466 000137 016440 JMP 28$ ;GO TO 28$ IF ERROR
827 015472 5$:
828
829 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      015472 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
830
831 ;WAIT FOR COMMAND TO COMPLETE
      015476 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
832
833 ;GO READ STATUS FOR FORMAT OPERATION
834 015502 004737 040110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
      015506 000404 BR 6$ ;GO TO 6$ IF NO ERROR
      015510 000240 NOP ;RETURN HERE IF ERROR
      015512 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
      015514 000137 016440 JMP 28$ ;GO TO 28$ IF ERROR
835 015520 6$:
836
837 ;VERIFY NO ERRORS DURING FORMAT
838 015520 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
      015524 000405 BR 7$ ;GO TO 7$ IF NO ERROR
      015526 000240 NOP ;RETURN HERE IF ERROR
      015530 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
```



```

015532 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
015534 000137 016440 JMP 28$ :GO TO 28$ IF ERROR
839 015540 7$:
840
841 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
842 015540 012737 015550 001124 MOV #8$,$LPERR
843 015546 000410 BR 9$ ;SKIP TO WRITE OPERATION
844
845 ;*****
846 ;LOOP #2 WRITE,WRITE CHECK DATA
847
848 015550 8$:
849
850 ;PREPARE DEVICE FOR WRITE OPERATION
851
852 015550 004737 034032 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
015554 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 9$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 28$ IF ERROR
015556 000404 BR 9$
015560 000240 NOP
015562 104000 EMT
015564 000137 016440 JMP 28$
853 015570 9$:
854
855 ;WRITE DATA TO THE DRIVE
856 015570 10$:
857 015570 012737 177400 001414 MOV #-256.,RMWCO ;CHANGE WORD COUNT
858 015576 012737 104260 001416 MOV #BUFONE+4,RMBA0 ;CHANGE MEMORY ADDRESS
859 015604 012737 000061 001412 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
860 015612 012702 001555 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
861 015616 112722 000006 MOVB #RMDA,(R2)+
862 015622 112722 000034 MOVB #RMDC,(R2)+
863 015626 112722 000032 MOVB #RMOF,(R2)+
864 015632 112722 000004 MOVB #RMBA,(R2)+
865 015636 112722 000002 MOVB #RMWC,(R2)+
866 015642 112722 000000 MOVB #RMCS1,(R2)+
867 015646 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
868
869 015652 004737 040360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
015656 000404 BR 11$ ;GO TO 11$ IF NO ERROR
015660 000240 NOP ;RETURN HERE IF ERROR
015662 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
015664 000137 016440 JMP 28$ ;GO TO 28$ IF ERROR
870 015670 11$:
871
872 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
015670 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
873
874 ;WAIT FOR COMMAND TO COMPLETE
015674 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
875

```



```

876                                     :GO READ STATUS FOR WRITE COMMAND
877 015700 004737 040110                JSR   PC,GET           :GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR     12$           :GO TO 12$ IF NO ERROR
                                NOP                    :RETURN HERE IF ERROR
                                EMT                    :ERROR # DEFINED BY GET SUBROUTINE
                                JMP     28$           :GO TO 28$ IF ERROR
878 015716                                12$:
879
880                                     :CHECK FOR ERRORS DURING WRITE OPERATION
881 015716 004737 041126                JSR   PC,PRIERR       :GO CHECK FOR PRIMARY ERRORS
                                BR     13$           :GO TO 13$ IF NO ERROR
                                NOP                    :RETURN HERE IF ERROR
                                EMT                    :ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR   PC,a(SP)+        :GO BACK FOR MORE ERROR CHECKS
                                JMP     28$           :GO TO 28$ IF ERROR
882 015736                                13$:
883 015736 004737 054166                JSR   PC,DTASTS       :GO VERIFY RESULTS OF DATA TRANSFER
                                BR     14$           :GO TO 14$ IF NO ERROR
                                NOP                    :RETURN HERE IF ERROR
                                EMT                    :ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR   PC,a(SP)+        :GO BACK FOR MORE ERROR CHECKS
                                JMP     28$           :GO TO 28$ IF ERROR
884 015756                                14$:
885 015756 004737 041760                JSR   PC,SECERR       :GO CHECK FOR SECONDARY ERRORS
                                BR     15$           :GO TO 15$ IF NO ERROR
                                NOP                    :RETURN HERE IF ERROR
                                EMT                    :ERROR # DEFINED BY SECERR SUBROUTINE
                                JSR   PC,a(SP)+        :GO BACK FOR MORE ERROR CHECKS
                                JMP     28$           :GO TO 28$ IF ERROR
886 015776                                15$:
887
888                                     :CHANGE LOOP ADDRESSES
889 015776 012737 016010 001124        MOV   #16$,$LPERR
890
891                                     :*****
892                                     :LOOP #3      WRITE CHECK DATA
893
894 016004 012703 000001                MOV   #1,R3           :R3+WCE BIT POSITION
895 016010                                16$:
896 016010 040337 105256                BIC   R3,BUFTWO-2    :CHANGE LAST WORD OF BUFFER
897
898                                     :PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
899 016014 004737 034032                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     17$           :GO TO 17$ IF NO ERROR
                                NOP                    :RETURN HERE IF ERROR
                                EMT                    :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                JMP     28$           :GO TO 28$ IF ERROR

                                016022 000404
                                016024 000240
                                016026 104000
                                016030 000137 016440
                                016034
900                                     17$:
901

```

```

902                                     :READ DATA FROM DEVICE
903 016034                               18$:
904 016034 012737 000051 001412      MOV    #WCD!GO,RMCS10      ;WRITE CHECK DATA DATA COMMAND
905 016042 012702 001555                MOV    #PUT!NX,R2        ;LOAD PUT REGISTER INDEX TABLE
906 016046 112722 000006                MOVB   #RMDA,(R2)+
907 016052 112722 000032                MOVB   #RMOF,(R2)+
908 016056 112722 000034                MOVB   #RMDC,(R2)+
909 016062 112722 000004                MOVB   #RMBA,(R2)+
910 016066 112722 000002                MOVB   #RPMC,(R2)+
911 016072 112722 000000                MOVB   #RMCS1,(R2)+
912 016076 112712 000200                MOVB   #200,(R2)
913
914 016102 004737 040360                JSR    PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    016106 000404                        BR     19$              ;GO TO 19$ IF NO ERROR
    016110 000240                        NOP
    016112 104000                        EMT
    016114 000137 016440                JMP    28$              ;ERROR # DEFINED BY PUT SUBROUTINE
    ;GO TO 28$ IF ERROR
915 016120                               19$:
916
917                                     ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
    JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
918
919                                     ;WAIT FOR COMMAND TO COMPLETE
    JSR    PC,TIMOUT     ;GO TO TIMOUT SUBROUTINE
920
921                                     ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
922 016130 004737 040110                JSR    PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
    016134 000404                        BR     20$              ;GO TO 20$ IF NO ERROR
    016136 000240                        NOP
    016140 104000                        EMT
    016142 000137 016440                JMP    28$              ;ERROR # DEFINED BY GET SUBROUTINE
    ;GO TO 28$ IF ERROR
923 016146                               20$:
924
925                                     ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
926 016146 004737 041126                JSR    PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
    016152 000405                        BR     21$              ;GO TO 21$ IF NO ERROR
    016154 000240                        NOP
    016156 104000                        EMT
    016160 004736                        JSR    PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
    016162 000137 016440                JMP    28$              ;GO TO 28$ IF ERROR
927 016166                               21$:
928 016166 032737 040000 001346        BIT    #WCE,RMCS2I     ;WAS "WCE" DETECTED??
929 016174 001030                        BNE    23$              ;YES!!
930 016176 004737 054166                JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
    016202 000405                        BR     22$              ;GO TO 22$ IF NO ERROR
    016204 000240                        NOP
    016206 104000                        EMT
    016210 004736                        JSR    PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
    016212 000137 016440                JMP    28$              ;GO TO 28$ IF ERROR
931 016216                               22$:
932 016216 013737 001346 001140        MOV    RMCS2I,$GDDAT   ;EXPECTED STATUS
933 016224 052737 040000 001140        BIS    #WCE,$GDDAT
934 016232 013737 001346 001142        MOV    RMCS2I,$BDDAT   ;RECEIVED STATUS
935 016240 010337 001174                MOV    R3,$TMP0        ;FAILING BIT POSITION
936 016244 012737 105256 001176        MOV    #BUF TWO-2,$TMP1 ;FAILING ADDRESS
937 016252 104337                        EMT
938 016254 000471                        BR     28$
    
```



```

939
940 016256      23$:
941 016256 112737 000022 001526   MOVB  #RMDB,GETINX   ;SETUP GET INDEX TABLE
    016264 112737 000200 001527   MOVB  #200,GETINX+1 ;SETUP TERMINATOR BYTE
    016272 012737 016440 001360   MOV   #28$,RMDBI    ;SET RMDB INPUT BUFFER = 28$
    016300 004737 040110           JSR   PC,GET        ;GO READ RMDB VIA GET SUBROUTINE
    016304 000402           BR    24$          ;GO TO 24$ IF NO ERROR
    016306 000240           NOP                    ;RETURN HERE IF ERROR
    016310 104000           EMT                    ;ERROR DEFINED BY GET SUBROUTINE

942
943 016312      24$:
944 016312 013737 001360 001142   MOV   RMDBI,$BDDAT  ;RECEIVED DATA
945 016320 013737 105256 001140   MOV   BUFTWO-2,$GDDAT ;EXPECTED DATA
946 016326 050337 001140           BIS   R3,$GDDAT
947 016332 012737 105256 001134   MOV   #BUFTWO-2,$GDADR ;EXPECTED ADDRESS
948 016340 013737 001342 001136   MOV   RMBAI,$BDADR  ;RECEIVED ADDRESS
949 016346 162737 000002 001136   SUB   #2,$BDADR     ;CORRECT MEMORY ADDRESS
950 016354 023737 001134 001136   CMP   $GDADR,$BDADR ;ADDRESSES OK??
951 016362 001402           BEQ   25$          ;YES!!
952 016364 104340           EMT 340
953 016366 000424           BR    28$
954 016370 023737 001140 001142 25$:  CMP   $GDDAT,$BDDAT ;DATA OK??
955 016376 001402           BEQ   26$          ;YES!!
956 016400 104341           EMT 341
957 016402 000416           BR    28$
958 016404      26$:
959
960 016404 004737 041760           JSR   PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
    016410 000405           BR    27$          ;GO TO 27$ IF NO ERROR
    016412 000240           NOP                    ;RETURN HERE IF ERROR
    016414 104000           EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
    016416 004736           JSR   PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
    016420 000137 016440           JMP   28$          ;GO TO 28$ IF ERROR

961 016424      27$:
962 016424 050337 105256           BIS   R3,BUFTWO-2  ;RESTORE DATA PATTERN
963 016430 006303           ASL   R3            ;SHIFT TO NEXT BIT
964 016432 001402           BEQ   28$          ;EXIT IF DONE
965 016434 000137 016010           JMP   16$          ;REPEAT TEST FOR NEXT DATA BIT
966 016440      28$:
967
968
*****
;*TEST 10 WRITE, WRITE CHECK MULTIPLE SECTORS
*****
TST10:
016440
016440 000004           SCOPE                ;SCOPE CALL
016442 000240           NOP                  ;START OF TEST
016444 012706 001100           MOV   #STACK,SP     ;INITIALIZE STACK POINTER
016450 013700 001276           MOV   $BASE,R0      ;R0 = UNIBUS ADDRESS
016454 013701 001466           MOV   TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
016460 012737 000010 001226   MOV   #10,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX

969
970
971 *****
972 ;LOOP #1 FORMAT,WRITE,WRITE CHECK
973
974 016466      1$:
975 016466 012737 010000 001444   MOV   #FMT16,RMOFO ;16 BIT FORMAT
  
```



```

976                                     ;LOAD PARAMETERS AND GENERATE DATA BUFFER
977 016474                               2$:
978 016474 012737 001057 001446          MOV #559.,RMDCO           ;CYLINDER = 559.
979 016502 012737 001000 001420          MOV #TA2,RMDAO          ;TRACK = 2, SECTOR = 0
980 016510 012737 104254 001416          MOV #BUFONE,RMBAO       ;BUS ADDRESS
981 016516 012737 176774 001414          MOV #-258.*2,RMWCO      ;WORD COUNT FOR 2 SECTORS (2'S COMP)
982 016524 012737 000063 001412          MOV #WH!GO,RMCS10       ;WRITE HEADER AND DATA COMMAND
983
984
985 016532 004737 034032                 ;PREPARE THE DEVICE FOR FORMAT OPERATION
    016536 154130                         JSR PC,TSTPRP           ;PREPARE DEVICE FOR TEST
    .WORD 154130                          .WORD 154130           ;TASK DESCRIPTOR AS FOLLOWS:
    ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
    ;CLEAR CONTROLLER & SELECT DEVICE
    ;VERIFY CONTROLLER CLEAR OPERATION
    ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
    ;VERIFY PACK ACKNOWLEDGE
    ;RECALIBRATE IF "SKI" OR "PIP" IS SET
    ;VERIFY RECALIBRATION
    ;GO TO 3$ IF NO ERROR
    ;RETURN HERE IF ERROR
    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    ;GO TO 25$ IF ERROR
    BR 3$
    NOP
    EMT
    JMP 25$
986 016540 000404                         3$:
987 016542 000240
988 016544 104000
    016546 000137 017466
    016552
    ;VERIFY THAT SECTOR IS NOT BAD
    JSR PC,BADSCT         ;CALL BAD SECTOR MODULE
    BR 4$                 ;GO TO 4$ IF NO ERROR
    TYPE ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
    EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
    JMP 25$                ;GO TO 25$ IF ERROR
989 016552 004737 034762
    016556 000405
    016560 104401 066006
    016564 104000
    016566 000137 017466
    4$:
990 016572 012737 067456 001174          MOV #ZEROS,$TMP0        ;STARTING ADDRESS OF PATTERN
991 016600 012737 000001 001176          MOV #1,$TMP1           ;RANGE OF PATTERN
992 016606 004737 037214                 JSR PC,GENBUF          ;GO GENERATE BUFFER FOR FORMAT
993
994                                     ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
995 016612 012702 001555                 MOV #PUTINX,R2         ;R2 = BYTE ENTRY POSITION
996 016616 112722 000034                 MOV #RMDC,(R2)+
997 016622 112722 000006                 MOV #RMDA,(R2)+
998 016626 112722 000004                 MOV #RMBA,(R2)+
999 016632 112722 000002                 MOV #RMWC,(R2)+
1000 016636 112722 000032                 MOV #RMOF,(R2)+
1001 016642 112722 000000                 MOV #RMCS1,(R2)+
1002 016646 112712 000200                 MOV #200,(R2)         ;TERMINATE TABLE
1003 016652
1004
1005                                     ;FORMAT THE DRIVE
1006 016652 004737 040360                 JSR PC,PUT             ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    016656 000404                         BR 6$                 ;GO TO 6$ IF NO ERROR
    016660 000240                         NOP                    ;RETURN HERE IF ERROR
    016662 104000                         EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
    016664 000137 017466                 JMP 25$                ;GO TO 25$ IF ERROR
1007 016670
1008
1009                                     ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1010 016670 004737 040024                 JSR PC,GETSTS          ;GO TO GETSTS SUBROUTINE
  
```

```

1011          ;WAIT FOR COMMAND TO COMPLETE
016674 004737 040732      JSR      PC,TIMOUT      ;GO TO TIMEOUT SUBROUTINE
1012
1013          ;GO READ STATUS FOR FORMAT OPERATION
1014 016700 004737 040110      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
016704 000404              BR      7$              ;GO TO 7$ IF NO ERROR
016706 000240              NOP              ;RETURN HERE IF ERROR
016710 104000              EMT              ;ERROR # DEFINED BY GET SUBROUTINE
016712 000137 017466      JMP      25$            ;GO TO 25$ IF ERROR
1015 016716      7$:
1016
1017          ;VERIFY NO ERRORS DURING FORMAT
1018 016716 004737 054166      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
016722 000405              BR      8$              ;GO TO 8$ IF NO ERROR
016724 000240              NOP              ;RETURN HERE IF ERROR
016726 104000              EMT              ;ERROR # DEFINED BY DTASTS SUBROUTINE
016730 004736              JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
016732 000137 017466      JMP      25$            ;GO TO 25$ IF ERROR
1019 016736      8$:
1020
1021          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
1022 016736 012737 017010 001124  MOV      #9$,$LPERR
1023
1024          ;REGENERATE DATA BUFFER
1025 016744 012737 177000 001414  MOV      #-256.*2,RMWC0 ;CHANGE WORD COUNT
1026 016752 012737 104254 001416  MOV      #BUFONE,RMBA0 ;CHANGE BUS ADDRESS
1027 016760 012737 000060 001412  MOV      #WD,RMCS10    ;WRITE DATA
1028 016766 012737 067414 001174  MOV      #ONES,$TMP0   ;STARTING ADDRESS OF PATTERN
1029 016774 012737 000001 001176  MOV      #1,$TMP1     ;RANGE OF PATTERN
1030 017002 004737 037214      JSR      PC,GENBUF
1031 017006 000410              BR      10$            ;SKIP TO WRITE OPERATION
1032
1033          ;*****
1034          ;LOOP #2      WRITE,WRITE CHECK
1035
1036 017010      9$:
1037
1038          ;PREPARE DEVICE FOR WRITE OPERATION
1039
1040 017010 004737 034032      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
017014 154130      .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 10$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 25$ IF ERROR
                                BR      10$
                                NOP
                                EMT
                                JMP      25$
1041 017016 000404              BR      10$
1042 017020 000240              NOP
1043 017022 104000              EMT
1044 017024 000137 017466      JMP      25$
1041 017030      10$:
1042
1043          ;WRITE DATA TO THE DRIVE
1044 017030      11$:
1045 017030 012737 000061 001412  MOV      #WD!GO,RMCS10 ;WRITE DATA COMMAND
  
```



```

1046 017036 012702 001555      MOV      #PUTINX,R2          ;LOAD PUT REGISTER INDEX TABLE
1047 017042 112722 000006      MOVB    #RMDA,(R2)+
1048 017046 112722 000034      MOVB    #RMDC,(R2)+
1049 017052 112722 000032      MOVB    #RMOF,(R2)+
1050 017056 112722 000004      MOVB    #RMB A,(R2)+
1051 017062 112722 000002      MOVB    #RMWC,(R2)+
1052 017066 112722 000000      MOVB    #RMCS1,(R2)+
1053 017072 112722 000200      MOVB    #200,(R2)+          ;TERMINATE TABLE
1054
1055 017076 004737 040360      JSR     PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      017102 000404      BR     12$                 ;GO TO 12$ IF NO ERROR
      017104 000240      NOP
      017106 104000      EMT
      017110 000137 017466      JMP    25$                 ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 25$ IF ERROR
1056 017114      12$:
1057
1058      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR     PC,GETSTS        ;GO TO GETSTS SUBROUTINE
1059
1060      ;WAIT FOR COMMAND TO COMPLETE
      JSR     PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
1061
1062      ;GO READ STATUS FOR WRITE COMMAND
1063 017124 004737 040110      JSR     PC,GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
      017130 000404      BR     13$                 ;GO TO 13$ IF NO ERROR
      017132 000240      NOP
      017134 104000      EMT
      017136 000137 017466      JMP    25$                 ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 25$ IF ERROR
1064 017142      13$:
1065
1066      ;CHECK FOR ERRORS DURING WRITE OPERATION
1067 017142 004737 041126      JSR     PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
      017146 000405      BR     14$                 ;GO TO 14$ IF NO ERROR
      017150 000240      NOP
      017152 104000      EMT
      017154 004736      JSR     PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
      017156 000137 017466      JMP    25$                 ;GO TO 25$ IF ERROR
1068 017162      14$:
1069 017162 004737 054166      JSR     PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
      017166 000405      BR     15$                 ;GO TO 15$ IF NO ERROR
      017170 000240      NOP
      017172 104000      EMT
      017174 004736      JSR     PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
      017176 000137 017466      JMP    25$                 ;GO TO 25$ IF ERROR
1070 017202      15$:
1071 017202 004737 041760      JSR     PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
      017206 000405      BR     16$                 ;GO TO 16$ IF NO ERROR
      017210 000240      NOP
      017212 104000      EMT
      017214 004736      JSR     PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
      017216 000137 017466      JMP    25$                 ;GO TO 25$ IF ERROR
1072 017222      16$:
1073
1074      ;CHANGE LOOP ADDRESSES
1075 017222 012737 017232 001124      MOV     #17$,SLPERR
1076 017230 000410      BR     18$
1077      ;SKIP TO NEXT OPERATION

```



```

1078 ::*****
1079 :LOOP #3 WRITE CHECK
1080
1081 017232 17$:
1082
1083 :PREPARE DEVICE FOR READ OPERATION
1084 017232 004737 034032 JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
      017236 154130 .WORD 154130 :TASK DESCRIPTOR AS FOLLOWS:
                                           :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                           :CLEAR CONTROLLER & SELECT DEVICE
                                           :VERIFY CONTROLLER CLEAR OPERATION
                                           :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                           :VERIFY PACK ACKNOWLEDGE
                                           :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                           :VERIFY RECALIBRATION
                                           :GO TO 18$ IF NO ERROR
                                           :RETURN HERE IF ERROR
                                           :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                           :GO TO 25$ IF ERROR
      017240 000404 BR 18$
      017242 000240 NOP
      017244 104000 EMT
      017246 000137 017466 JMP 25$
1085 017252 18$:
1086
1087 :READ DATA FROM DEVICE
1088 017252 19$:
1089 017252 012737 000051 001412 MOV #WCD!GO,RMCS10 :READ DATA COMMAND
1090 017260 012702 001555 MOV #PUTINX,R2 :LOAD PUT REGISTER INDEX TABLE
1091 017264 112722 000006 MOV #RMDA,(R2)+
1092 017270 112722 000032 MOV #RMOF,(R2)+
1093 017274 112722 000034 MOV #RMDC,(R2)+
1094 017300 112722 000004 MOV #RMBA,(R2)+
1095 017304 112722 000002 MOV #RMWC,(R2)+
1096 017310 112722 000000 MOV #RMCS1,(R2)+
1097 017314 112712 000200 MOV #200,(R2)
1098
1099 017320 004737 040360 JSR PC,PUT :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      017324 000404 BR 20$ :GO TO 20$ IF NO ERROR
      017326 000240 NOP :RETURN HERE IF ERROR
      017330 104000 EMT :ERROR # DEFINED BY PUT SUBROUTINE
      017332 000137 017466 JMP 25$ :GO TO 25$ IF ERROR
1100 017336 20$:
1101
1102 :SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR PC,GETSTS :GO TO GETSTS SUBROUTINE
1103
1104 :WAIT FOR COMMAND TO COMPLETE
      JSR PC,TIMOUT :GO TO TIMOUT SUBROUTINE
1105
1106 :GO READ STATUS FOR WRITE CHECK OPERATION
1107 017346 004737 040110 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
      017352 000404 BR 21$ :GO TO 21$ IF NO ERROR
      017354 000240 NOP :RETURN HERE IF ERROR
      017356 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
      017360 000137 017466 JMP 25$ :GO TO 25$ IF ERROR
1108 017364 21$:
1109
1110 :CHECK FOR ERRORS DURING WRITE CHECK OPERATION
1111 017364 004737 041126 JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
      017370 000405 BR 22$ :GO TO 22$ IF NO ERROR
  
```

```

017372 000240      NOP      :RETURN HERE IF ERROR
017374 104000      EMT      :ERROR # DEFINED BY PRIERR SUBROUTINE
017376 004736      JSR      PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
017400 000137 017466 JMP      25$      :GO TO 25$ IF ERROR
1112 017404      22$:      JSR      PC,DTASTS :GO VERIFY RESULTS OF DATA TRANSFER
1113 017404 004737 054166 BR      23$      :GO TO 23$ IF NO ERROR
017410 000405      NOP      :RETURN HERE IF ERROR
017412 000240      EMT      :ERROR # DEFINED BY DTASTS SUBROUTINE
017414 104000      JSR      PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
017416 004736      JMP      25$      :GO TO 25$ IF ERROR
1114 017424      23$:      JSR      PC,SECERR :GO CHECK FOR SECONDARY ERRORS
1115 017424 004737 041760 BR      24$      :GO TO 24$ IF NO ERROR
017430 000405      NOP      :RETURN HERE IF ERROR
017432 000240      EMT      :ERROR # DEFINED BY SECERR SUBROUTINE
017434 104000      JSR      PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
017436 004736      JMP      25$      :GO TO 25$ IF ERROR
1116 017444      24$:      BIT      #SSEI,RMOFO :TEST 16 BIT MODE W/ SSEI SET ?
1117 017444 032737 001000 001444 BNE     25$      :YES !!
1118 017452 001005      BIS      #SSEI,RMOFO :SET SSEI IN OFFSET AND
1119 017454 052737 001000 001444 JMP      2$        :TEST AGAIN
1120 017462 000137 016474
1121 017466
1122
1123

```

```

:*****
:*TEST 11      WRITE, READ W/ IMPLIED SEEK
:*****

```

```

TST11:
017466 000004      SCOPE     :SCOPE CALL
017466 000240      NOP      :START OF TEST
017470 012706 001100 MOV      #STACK,SP :INITIALIZE STACK POINTER
017472 013700 001276 MOV      $BASE,R0  :R0 = UNIBUS ADDRESS
017502 013701 001466 MOV      TSTQUE,R1 : (R1) = DEVICE BEING TESTED
017506 012737 000011 001226 MOV      #11,$TESTN :SET TEST NUMBER IN APT MAIL BOX

```

```

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

```

```

1124
1125
1126
1127
1128 017514      1$:
1129
1130
1131 017514 004737 034032 :PREPARE THE DEVICE FOR FORMAT OPERATION
017520 154130      JSR      PC,TSTPRP :PREPARE DEVICE FOR TEST
                          .WORD 154130 :TASK DESCRIPTOR AS FOLLOWS:
                          :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                          :CLEAR CONTROLLER & SELECT DEVICE
                          :VERIFY CONTROLLER CLEAR OPERATION
                          :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                          :VERIFY PACK ACKNOWLEDGE
                          :RECALIBRATE IF "SKI" OR "PIP" IS SET
                          :VERIFY RECALIBRATION
017522 000404      BR      2$        :GO TO 2$ IF NO ERROR
017524 000240      NOP      :RETURN HERE IF ERROR
017526 104000      EMT      :ERROR # DEFINED BY TSTPRP SUBROUTINE
017530 000137 020706 JMP      31$      :GO TO 31$ IF ERROR

```

```

:LOAD PARAMETERS AND GENERATE DATA BUFFER

```

```

1132
1133

```



```

1134 017534          2$:
1135 017534 012737 001057 001446      MOV      #559.,RMDCO      ;CYLINDER = 559.
1136 017542 013737 001446 020710      MOV      RMDCO,32$      ;SAVE DESIRED CYLINDER
1137 017550 012737 001000 001420      MOV      #TA2,RMDAO     ;TRACK = 2, SECTOR = 0
1138 017556 012737 104254 001416      MOV      #BUFONE,RMBAO  ;BUS ADDRESS
1139 017564 012737 177376 001414      MOV      #-258.,RMWCO   ;2 + 256. WORDS (2'S COMP)
1140 017572 012737 010000 001444      MOV      #FMT16,RMOFO   ;16 BIT FORMAT
1141 017600 012737 000063 001412      MOV      #WH!GO,RMCS10  ;WRITE HEADER AND DATA COMMAND
1142
1143          ;VERIFY THAT SECTOR IS NOT BAD
          JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
          BR      3$              ;GO TO 3$ IF NO ERROR
          TYPE      ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
          EMT              ;ERROR # DEFINED BY BADSCT SUBROUTINE
          JMP      31$           ;GO TO 31$ IF ERROR
1144 017606 004737 034762
1145 017612 000405
1146 017614 104401 066006
1147 017620 104000
1148 017622 000137 020706
1149          3$:
1150 017626 012737 067414 001174      MOV      #ONES,$TMP0    ;STARTING ADDRESS OF PATTERN
1151 017634 012737 000001 001176      MOV      #1,$TMP1       ;RANGE OF PATTERN
1152 017642 004737 037214              JSR      PC,GENBUF      ;GO GENERATE BUFFER FOR FORMAT
1153
1154          ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
          MOV      #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
1155 017646 012702 001555      MOV      #RMDC,(R2)+
1156 017652 112722 000034      MOV      #RMDA,(R2)+
1157 017656 112722 000006      MOV      #RMBA,(R2)+
1158 017662 112722 000004      MOV      #RMWC,(R2)+
1159 017666 112722 000002      MOV      #RMOF,(R2)+
1160 017672 112722 000032      MOV      #RMCS1,(R2)+
1161 017676 112722 000000      MOV      #200,(R2)      ;TERMINATE TABLE
1162 017702 112712 000200
1163 017706
1164          4$:
1165          ;FORMAT THE DRIVE
          JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
1166 017712 000404      BR      5$              ;GO TO 5$ IF NO ERROR
1167 017714 000240      NOP              ;RETURN HERE IF ERROR
1168 017716 104000      EMT              ;ERROR # DEFINED BY PUT SUBROUTINE
1169 017720 000137 020706      JMP      31$           ;GO TO 31$ IF ERROR
1170 017724
1171          5$:
1172          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
          JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
1173 017724 004737 040024
1174          ;WAIT FOR COMMAND TO COMPLETE
          JSR      PC,TIMOUT      ;GO TO TIMEOUT SUBROUTINE
1175 017730 004737 040732
1176          ;GO READ STATUS FOR FORMAT OPERATION
          JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
1177 017734 004737 040110      BR      6$              ;GO TO 6$ IF NO ERROR
1178 017740 000404      NOP              ;RETURN HERE IF ERROR
1179 017742 000240      EMT              ;ERROR # DEFINED BY GET SUBROUTINE
1180 017744 104000      JMP      31$           ;GO TO 31$ IF ERROR
1181 017746 000137 020706
1182          6$:
1183          ;VERIFY NO ERRORS DURING FORMAT
          JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
1184 017752 004737 054166      BR      7$              ;GO TO 7$ IF NO ERROR
1185 017756 000405      NOP              ;RETURN HERE IF ERROR
1186 017760 000240
  
```



```

017762 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
017764 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017766 000137 020706 JMP 31$ ;GO TO 31$ IF ERROR
1174 017772 7$:
1175
1176 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
1177 017772 012737 020002 001124 MOV #8$,$LPERR
1178 020000 000410 BR 9$ ;SKIP TO WRITE OPERATION
1179
1180 ;*****
1181 ;LOOP #2 WRITE,READ
1182
1183 020002 8$:
1184
1185 ;PREPARE DEVICE FOR WRITE OPERATION
1186 020002 004737 034032 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
020006 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 9$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 31$ IF ERROR
020010 000404 BR 9$
020012 000240 NOP
020014 104000 EMT
020016 000137 020706 JMP 31$
1187 020022 9$:
1188 020022 012737 000000 001446 MOV #0,RMDCO ;SEEK TO FIRST CYLINDER
1189 020030 012737 000005 001412 MOV #SEEK!GO,RMCS10 ;WRITE SEEK COMMAND
1190
1191 020036 004737 040360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
020042 000404 BR 10$ ;GO TO 10$ IF NO ERROR
020044 000240 NOP ;RETURN HERE IF ERROR
020046 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
020050 000137 020706 JMP 31$ ;GO TO 31$ IF ERROR
1192 020054 10$:
1193
1194 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1195 020054 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
1196
1197 ;WAIT FOR COMMAND TO COMPLETE
1198 020060 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1199
1199 ;GO GET REGISTER STATUS
020064 004737 040110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
020070 000404 BR 11$ ;GO TO 11$ IF NO ERROR
020072 000240 NOP ;RETURN HERE IF ERROR
020074 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
020076 000137 020706 JMP 31$ ;GO TO 31$ IF ERROR
1200 020102 11$:
1201
1202 ;VERIFY RESULTS OF SEEK COMMAND
1203 020102 004737 046550 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
020106 000405 BR 12$ ;GO TO 12$ IF NO ERROR
020110 000240 NOP ;RETURN HERE IF ERROR

```

1204	020112	104000			EMT			:ERROR # DEFINED BY SEKSTS SUBROUTINE
1205	020114	004736			JSR	PC,a(SP)+		:GO BACK FOR MORE ERROR CHECKS
1206	020116	000137	020706		JMP	31\$:GO TO 31\$ IF ERROR
1207	020122						12\$:	
1208	020122	013737	020710	001446			13\$:	:SETUP PARAMETERS AND EXECUTE WRITE DATA COMMAND
1209	020130	012737	177400	001414	MOV	32\$,RMDCO		:RESTORE CYLINDER
1210	020136	012737	104260	001416	MOV	#-256,RMWC0		:CHANGE WORD COUNT
1211	020144	012737	000061	001412	MOV	#BUFONÉ+4,RMBA0		:CHANGE BUS ADDRESS
1212	020152	012702	001555		MOV	#WD!GO,RMCS10		:WRITE DATA COMMAND
1213	020156	112722	000006		MOV	#PUTINX,R2		:LOAD PUT REGISTER INDEX TABLE
1214	020162	112722	000034		MOVB	#RMDA,(R2)+		
1215	020166	112722	000032		MOVB	#RMDC,(R2)+		
1216	020172	112722	000004		MOVB	#RMOF,(R2)+		
1217	020176	112722	000002		MOVB	#RMBA,(R2)+		
1218	020202	112722	000000		MOVB	#RMWC,(R2)+		
1219	020206	112722	000200		MOVB	#RMCS1,(R2)+		:TERMINATE TABLE
1221	020212	004737	040360		JSR	PC,PUT		:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	020216	000404			BR	14\$:GO TO 14\$ IF NO ERROR
	020220	000240			NOP			:RETURN HERE IF ERROR
	020222	104000			EMT			:ERROR # DEFINED BY PUT SUBROUTINE
	020224	000137	020706		JMP	31\$:GO TO 31\$ IF ERROR
1222	020230						14\$:	
1223								:WAIT FOR COMMAND TO COMPLETE
1224	020230	004737	040732		JSR	PC,TIMOUT		:GO TO TIMOUT SUBROUTINE
1225								:GO READ STATUS FOR WRITE COMMAND
1226	020234	004737	040110		JSR	PC,GET		:GO READ REGISTER(S) WITH GET SUBROUTINE
1227	020240	000404			BR	15\$:GO TO 15\$ IF NO ERROR
	020242	000240			NOP			:RETURN HERE IF ERROR
	020244	104000			EMT			:ERROR # DEFINED BY GET SUBROUTINE
	020246	000137	020706		JMP	31\$:GO TO 31\$ IF ERROR
1228	020252						15\$:	
1229								:CHECK FOR ERRORS DURING WRITE OPERATION
1230	020252	004737	041126		JSR	PC,PRIERR		:GO CHECK FOR PRIMARY ERRORS
1231	020256	000405			BR	16\$:GO TO 16\$ IF NO ERROR
	020260	000240			NOP			:RETURN HERE IF ERROR
	020262	104000			EMT			:ERROR # DEFINED BY PRIERR SUBROUTINE
	020264	004736			JSR	PC,a(SP)+		:GO BACK FOR MORE ERROR CHECKS
	020266	000137	020706		JMP	31\$:GO TO 31\$ IF ERROR
1232	020272						16\$:	
1233	020272	004737	054166		JSR	PC,DTASTS		:GO VERIFY RESULTS OF DATA TRANSFER
	020276	000405			BR	17\$:GO TO 17\$ IF NO ERROR
	020300	000240			NOP			:RETURN HERE IF ERROR
	020302	104000			EMT			:ERROR # DEFINED BY DTASTS SUBROUTINE
	020304	004736			JSR	PC,a(SP)+		:GO BACK FOR MORE ERROR CHECKS
	020306	000137	020706		JMP	31\$:GO TO 31\$ IF ERROR
1234	020312						17\$:	
1235	020312	004737	041760		JSR	PC,SECERR		:GO CHECK FOR SECONDARY ERRORS
	020316	000405			BR	18\$:GO TO 18\$ IF NO ERROR
	020320	000240			NOP			:RETURN HERE IF ERROR
	020322	104000			EMT			:ERROR # DEFINED BY SECERR SUBROUTINE


```

1236 020324 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
1237 020326 000137 020706 JMP 31$ ;GO TO 31$ IF ERROR
1238 ;CHANGE LOOP ADDRESSES
1239 020332 012737 020342 001124 MOV #19$,$LPERR
1240 020340 000410 BR 20$ ;SKIP TO NEXT OPERATION
1241
1242 ::*****
1243 ;LOOP #3 READ
1244
1245 020342 19$:
1246 ;PREPARE DEVICE FOR READ OPERATION
1247 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
1248 020342 004737 034032 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 20$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 31$ IF ERROR
020350 000404 BR 20$
020352 000240 NOP
020354 104000 EMT
020356 000137 020706 JMP 31$
1249 020362 20$:
1250 020362 012737 000000 001446 MOV #0,RMDCO ;SEEK TO FIRST CYLINDER
1251 020370 012737 000005 001412 MOV #SEEK!GO,RMCS10 ;WRITE SEEK COMMAND
1252
1253 020376 004737 040360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
020402 000404 BR 21$ ;GO TO 21$ IF NO ERROR
020404 000240 NOP ;RETURN HERE IF ERROR
020406 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
020410 000137 020706 JMP 31$ ;GO TO 31$ IF ERROR
1254 020414 21$:
1255
1256 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
020414 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
1257
1258 ;WAIT FOR COMMAND TO COMPLETE
020420 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1259
1260 ;GO GET REGISTER STATUS
1261 020424 004737 040110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
020430 000404 BR 22$ ;GO TO 22$ IF NO ERROR
020432 000240 NOP ;RETURN HERE IF ERROR
020434 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
020436 000137 020706 JMP 31$ ;GO TO 31$ IF ERROR
1262 020442 22$:
1263
1264 ;VERIFY RESULTS OF SEEK COMMAND
1265 020442 004737 046550 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
020446 000405 BR 23$ ;GO TO 23$ IF NO ERROR
020450 000240 NOP ;RETURN HERE IF ERROR
020452 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
  
```



```

020454 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
020456 000137 020706   JMP    31$          ;GO TO 31$ IF ERROR

1266
1267 020462          23$:
1268
1269                ;SETUP PARAMETERS AND EXECUTE READ DATA COMMAND
1270 020462          24$:
1271 020462 013737 020710 001446   MOV    32$,RMDCO      ;RESTORE CYLINDER
1272 020470 012737 000071 001412   MOV    #RD!GO,RMCS10 ;READ DATA COMMAND
1273 020476 012737 105264 001416   MOV    #BUFTWO+4,RMBA0 ;LOAD STARTING BUFFER ADDRESS
1274 020504 012702 001555          MOV    #PUTINX,R2     ;LOAD PUT REGISTER INDEX TABLE
1275 020510 112722 000006          MOVB   #RMDA,(R2)+
1276 020514 112722 000032          MOVB   #RMOF,(R2)+
1277 020520 112722 000034          MOVB   #RMDC,(R2)+
1278 020524 112722 000004          MOVB   #RMBA,(R2)+
1279 020530 112722 000002          MOVB   #RMWC,(R2)+
1280 020534 112722 000000          MOVB   #RMCS1,(R2)+
1281 020540 112712 000200          MOVB   #200,(R2)

1283 020544 004737 040360          JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
020550 000404          BR     25$          ;GO TO 25$ IF NO ERROR
020552 000240          NOP                    ;RETURN HERE IF ERROR
020554 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
020556 000137 020706   JMP    31$          ;GO TO 31$ IF ERROR

1284 020562          25$:
1285
1286                ;WAIT FOR COMMAND TO COMPLETE
020562 004737 040732          JSR    PC,TIMOUT    ;GO TO TIMEOUT SUBROUTINE

1287
1288                ;GO READ STATUS FOR READ OPERATION
1289 020566 004737 040110          JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
020572 000404          BR     26$          ;GO TO 26$ IF NO ERROR
020574 000240          NOP                    ;RETURN HERE IF ERROR
020576 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
020600 000137 020706   JMP    31$          ;GO TO 31$ IF ERROR

1290 020604          26$:
1291
1292                ;CHECK FOR ERRORS DURING READ OPERATION
1293 020604 004737 041126          JSR    PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
020610 000405          BR     27$          ;GO TO 27$ IF NO ERROR
020612 000240          NOP                    ;RETURN HERE IF ERROR
020614 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
020616 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
020620 000137 020706   JMP    31$          ;GO TO 31$ IF ERROR

1294 020624          27$:
1295 020624 004737 054166          JSR    PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
020630 000405          BR     28$          ;GO TO 28$ IF NO ERROR
020632 000240          NOP                    ;RETURN HERE IF ERROR
020634 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
020636 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
020640 000137 020706   JMP    31$          ;GO TO 31$ IF ERROR

1296 020644          28$:
1297 020644 004737 041760          JSR    PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
020650 000405          BR     29$          ;GO TO 29$ IF NO ERROR
020652 000240          NOP                    ;RETURN HERE IF ERROR
020654 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
020656 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS

```

```

1298 020660 000137 020706          29$: JMP      31$          ;GO TO 31$ IF ERROR
1299 020664 004737 037462          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
020670 104260
020672 105264
020674 000404                      BR       30$          ;GO TO 30$ IF NO ERROR
020676 000240                      NOP
020700 104000                      EMT
020702 000137 020706          JMP      31$          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
                                ;GO TO 31$ IF ERROR
1300 020706          30$:
1301
1302 020706          31$:
1303 020706 000401                      BR       33$
1304
1305 020710 000000          32$: .WORD    0          ;STORAGE FOR DATA CYLINDER
1306
1307 020712          33$:
1308
1309
::*****
::*TEST 12      READ, WRITE CHECK W/ HEAD SWITCHING
::*****
TST12:
020712 000004          SCOPE          ;SCOPE CALL
020714 000240          NOP           ;START OF TEST
020716 012706 001100      MOV      #STACK,SP ;INITIALIZE STACK POINTER
020722 013700 001276      MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
020726 013701 001466      MOV      TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
020732 012737 000012 001226 MOV      #12,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

::*****
::LOOP #1      READ,WRITE CHECK
::LOAD PARAMETERS AND GENERATE DATA BUFFER
1310
1311
1312
1313
1314
1315 020740          1$:
1316 020740 012737 001036 001420      MOV      #TA2!30.,RMDAO ;TRACK = 2, SECTOR = 30.
1317 020746 012737 010000 001444      MOV      #FMT16,RMFO   ;16 BIT FORMAT
1318 020754          2$:
1319 020754 012737 001057 001446      MOV      #559.,RMDCO   ;CYLINDER = 559.
1320 020762 012737 104254 001416      MOV      #BUFOFF,RMBAO ;BUS ADDRESS
1321 020770 012737 177000 001414      MOV      #-256.*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2'S COMP)
1322 020776 012737 000070 001412      MOV      #RD,RMCS10   ;READ DATA COMMAND
1330 021004          3$:
1331
1332
1333 021004 004737 034032      ;PREPARE THE DEVICE FOR FORMAT OPERATION
                                JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
                                .WORD    154130 ;TASK DESCRIPTOR AS FOLLOWS:
                                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                                ;CLEAR CONTROLLER & SELECT DEVICE
                                                ;VERIFY CONTROLLER CLEAR OPERATION
                                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                                ;VERIFY PACK ACKNOWLEDGE
                                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                                ;VERIFY RECALIBRATION
021010 154130          BR       4$          ;GO TO 4$ IF NO ERROR
021012 000404          NOP           ;RETURN HERE IF ERROR
021014 000240          EMT
021016 104000          EMT
021020 000137 021734          JMP      26$        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 26$ IF ERROR

```



```

1334 021024          4$:
1335
1336                ;SETUP PARAMETERS AND SEEK TO GET DRIVE ON CYLINDER
1337 021024 012737 000005 001412  MOV    #SEEK!GO, RMCS10      ;LOAD SEEK COMMAND
1338 021032 012702 001555          MOV    #PUTINX, R2          ;R2 = BYTE ENTRY POSITION
1339 021036 112722 000034          MOVB  #RMDC, (R2)+
1340 021042 112722 000006          MOVB  #RMDA, (R2)+
1341 021046 112722 000004          MCVB  #RMBA, (R2)+
1342 021052 112722 000002          MOVB  #RMWC, (R2)+
1343 021056 112722 000032          MOVB  #RMOF, (R2)+
1344 021062 112722 000000          MOVB  #RMCS1, (R2)+
1345 021066 112712 000200          MOVB  #200, (R2)          ;TERMINATE TABLE
1346 021072          5$:
1347
1348 021072 004737 040360          JSR   PC, PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021076 000404          BR    6$                ;GO TO 6$ IF NO ERROR
      021100 000240          NOP                    ;RETURN HERE IF ERROR
      021102 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      021104 000137 021734          JMP   26$               ;GO TO 26$ IF ERROR
1349 021110          6$:
1350
1351                ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      021110 004737 040024          JSR   PC, GETSTS        ;GO TO GETSTS SUBROUTINE
1352
1353                ;WAIT FOR COMMAND TO COMPLETE
      021114 004737 040732          JSR   PC, TIMEOUT       ;GO TO TIMEOUT SUBROUTINE
1354
1355                ;GO READ REGISTER STATUS
      021120 004737 040110          JSR   PC, GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021124 000404          BR    7$                ;GO TO 7$ IF NO ERROR
      021126 000240          NOP                    ;RETURN HERE IF ERROR
      021130 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      021132 000137 021734          JMP   26$               ;GO TO 26$ IF ERROR
1357 021136          7$:
1358
1359                ;GO VERIFY RESULTS OF SEEK
      021136 004737 046550          JSR   PC, SEKSTS        ;GO VERIFY RESULTS OF SEEK OPERATION
      021142 000405          BR    8$                ;GO TO 8$ IF NO ERROR
      021144 000240          NOP                    ;RETURN HERE IF ERROR
      021146 104000          EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      021150 004736          JSR   PC, @ (SP)+       ;GO BACK FOR MORE ERROR CHECKS
      021152 000137 021734          JMP   26$               ;GO TO 26$ IF ERROR
1361 021156          8$:
1362
1363                ;READ DATA FROM THE DRIVE
1364 021156          9$:
1365 021156 012737 000071 001412  MOV    #RD!GO, RMCS10      ;READ DATA COMMAND
1366 021164 012702 001555          MOV    #PUTINX, R2        ;LOAD PUT REGISTER INDEX TABLE
1367 021170 112722 000006          MOVB  #RMDA, (R2)+
1370 021174 112722 000034          MOVB  #RMDC, (R2)+
1371 021174 112722 000034          MOVB  #RMDA, (R2)+
1372 021200 112722 000032          MOVB  #RMOF, (R2)+
1373 021204 112722 000004          MOVB  #RMBA, (R2)+
1374 021210 112722 000002          MOVB  #RMWC, (R2)+
1375 021214 112722 000000          MOVB  #RMCS1, (R2)+
1376 021220 112722 000200          MOVB  #200, (R2)+          ;TERMINATE TABLE
1377
1378 021224 004737 040360          JSR   PC, PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
  
```



```

021230 000404 BR 10$ :GO TO 10$ IF NO ERROR
021232 000240 NOP :RETURN HERE IF ERROR
021234 104000 EMT :ERROR # DEFINED BY PUT SUBROUTINE
021236 000137 021734 JMP 26$ :GO TO 26$ IF ERROR
1379 021242 10$:
1380
1381 :SETUP GET INDEX TABLE TO READ ALL REGISTERS
021242 004737 040024 JSR PC,GETSTS :GO TO GETSTS SUBROUTINE
1382
1383 :WAIT FOR COMMAND TO COMPLETE
021246 004737 040732 JSR PC,TIMOUT :GO TO TIMEOUT SUBROUTINE
1384
1385 :GO READ STATUS FOR READ COMMAND
1386 021252 004737 040110 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
021256 000404 BR 11$ :GO TO 11$ IF NO ERROR
021260 000240 NOP :RETURN HERE IF ERROR
021262 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
021264 000137 021734 JMP 26$ :GO TO 26$ IF ERROR
1387
1388 :CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
1389 :DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
1390 021270 11$:
1391 021270 032737 000040 001400 BIT #SSE,RMER2I :IS 'SSE' BIT SET ?
1392 021276 001030 BNE 14$ :BR IF YES
1393
1394 :CHECK FOR ERRORS DURING READ OPERATION
1395 021300 004737 041126 JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
021304 000405 BR 12$ :GO TO 12$ IF NO ERROR
021306 000240 NOP :RETURN HERE IF ERROR
021310 104000 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
021312 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
021314 000137 021734 JMP 26$ :GO TO 26$ IF ERROR
1396 021320 12$:
1397 021320 004737 054166 JSR PC,DTASTS :GO VERIFY RESULTS OF DATA TRANSFER
021324 000405 BR 13$ :GO TO 13$ IF NO ERROR
021326 000240 NOP :RETURN HERE IF ERROR
021330 104000 EMT :ERROR # DEFINED BY DTASTS SUBROUTINE
021332 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
021334 000137 021734 JMP 26$ :GO TO 26$ IF ERROR
1398 021340 13$:
1399 021340 004737 041760 JSR PC,SECERR :GO CHECK FOR SECONDARY ERRORS
021344 000405 BR 14$ :GO TO 14$ IF NO ERROR
021346 000240 NOP :RETURN HERE IF ERROR
021350 104000 EMT :ERROR # DEFINED BY SECERR SUBROUTINE
021352 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
021354 000137 021734 JMP 26$ :GO TO 26$ IF ERROR
1400 021360 14$:
1401
1402 :CHANGE LOOP ADDRESSES
1403 021360 012737 021370 001124 MOV #15$,SLPERR
1404 021366 000410 BR 16$ :SKIP TO NEXT OPERATION
1405
1406 :*****
1407 :LOOP #2 WRITE CHECK DATA
1408
1409 021370 15$:
1410

```

```

1411                                     ;PREPARE DEVICE FOR READ OPERATION
1412 021370 004737 034032             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 16$ IF NO ERROR
                                                ;RETURN HERE IF ERROR
                                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                                ;GO TO 26$ IF ERROR
                                021376 000404             BR       16$
                                021400 000240             NOP
                                021402 104000             EMT
                                021404 000137 021734     JMP      26$
1413 021410                                     16$:
1414 021410 012737 000005 001412     MOV      #SEEK!GO,RMCS10      ;LOAD SEEK COMMAND
1415
1416 021416 004737 040360             JSR      PC,PUT               ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                021422 000404             BR       17$                 ;GO TO 17$ IF NO ERROR
                                021424 000240             NOP                           ;RETURN HERE IF ERROR
                                021426 104000             EMT                           ;ERROR # DEFINED BY PUT SUBROUTINE
                                021430 000137 021734     JMP      26$                 ;GO TO 26$ IF ERROR
1417 021434                                     17$:
1418
1419 021434 004737 040024             ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
                                JSR      PC,GETSTS         ;GO TO GETSTS SUBROUTINE
1420
1421 021440 004737 040732             ;WAIT FOR COMMAND TO COMPLETE
                                JSR      PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
1422
1423
1424 021444 004737 040110             ;GO READ REGISTER STATUS
                                JSR      PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                021450 000404             BR       18$                 ;GO TO 18$ IF NO ERROR
                                021452 000240             NOP                           ;RETURN HERE IF ERROR
                                021454 104000             EMT                           ;ERROR # DEFINED BY GET SUBROUTINE
                                021456 000137 021734     JMP      26$                 ;GO TO 26$ IF ERROR
1425 021462                                     18$:
1426
1427
1428 021462 004737 046550             ;GO VERIFY RESLUTS OF SEEK
                                JSR      PC,SEKSTS         ;GO VERIFY RESULTS OF SEEK OPERATION
                                021466 000405             BR       19$                 ;GO TO 19$ IF NO ERROR
                                021470 000240             NOP                           ;RETURN HERE IF ERROR
                                021472 104000             EMT                           ;ERROR # DEFINED BY SEKSTS SUBROUTINE
                                021474 004736             JSR      PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
                                021476 000137 021734     JMP      26$                 ;GO TO 26$ IF ERROR
1429 021502                                     19$:
1430
1431
1432 021502                                     ;READ DATA FROM DEVICE
                                20$:
1433 021502 012737 000051 001412     MOV      #WCD!GO,RMCS10      ;READ DATA COMMAND
1434 021510 012702 001555             MOV      #PUTINX,R2          ;LOAD PUT REGISTER INDEX TABLE
1435 021514 112722 000006             MOVB    #RMDA,(R2)+
1436 021520 112722 000032             MOVB    #RMOF,(R2)+
1437 021524 112722 000034             MOVB    #RMDC,(R2)+
1438 021530 112722 000004             MOVB    #RMBA,(R2)+
1439 021534 112722 000002             MOVB    #RMWC,(R2)+
1440 021540 112722 000000             MOVB    #RMCS1,(R2)+
  
```



```

1441 021544 112712 000200          MOVB   #200,(R2)
1442
1443 021550 004737 040360          JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      21$          ;GO TO 21$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT         ;ERROR # DEFINED BY PUT SUBROUTINE
                                JMP      26$          ;GO TO 26$ IF ERROR
1444 021566
1445
1446          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
                                JSR    PC,GETSTS        ;GO TO GETSTS SUBROUTINE
1447 021566 004737 040024
1448          ;WAIT FOR COMMAND TO COMPLETE
                                JSR    PC,TIMOUT        ;GO TO TIMOUT SUBROUTINE
1449 021572 004737 040732
1450          ;GO READ STATUS FOR WRITE CHECK OPERATION
1451 021576 004737 040110          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR      22$          ;GO TO 22$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT         ;ERROR # DEFINED BY GET SUBROUTINE
                                JMP      26$          ;GO TO 26$ IF ERROR
1452 021610 000137 021734
1453          ;CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
1454          ;DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
1455 021614
1456 021614 032737 000040 001400    22$:  BIT    #SSE,RMER2I          ;IS 'SSE' BIT SET ?
1457 021622 001030                    BNE    25$          ;BR IF YES
1458
1459          ;CHECK FOR ERRORS DURING WRITE CHECK OPERATION
1460 021624 004737 041126          JSR    PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
                                BR      23$          ;GO TO 23$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT         ;ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR    PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
                                JMP      26$          ;GO TO 26$ IF ERROR
1461 021644
1462 021644 004737 054166          23$:  JSR    PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR      24$          ;GO TO 24$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT         ;ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR    PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
                                JMP      26$          ;GO TO 26$ IF ERROR
1463 021664
1464 021664 004737 041760          24$:  JSR    PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
                                BR      25$          ;GO TO 25$ IF NO ERROR
                                NOP          ;RETURN HERE IF ERROR
                                EMT         ;ERROR # DEFINED BY SECERR SUBROUTINE
                                JSR    PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
                                JMP      26$          ;GO TO 26$ IF ERROR
1465 021700 000137 021734
1466 021704
1467 021704 032737 001000 001444    25$:  BIT    #SSEI,RMOFO        ;TEST 16 BIT MODE W/ SSEI SET ?
1468 021712 001010                    BNE    26$          ;YES !!
1469 021714 012737 001037 001420    MOV    #TA2!31,RMDAO    ;TRACK = 1, SECTOR = 31.
1470 021722 052737 001000 001444    BIS    #SSEI,RMOFO      ;SET SSEI BIT IN OFFSET AND
1471 021730 000137 020754          JMP    2$              ;TEST AGAIN.
1472

```


1473

 :*TEST 13 READ, WRITE CHECK W/ MID-TRANSFER SEEK

 TST13:

021734	000004				SCOPE	:SCOPE CALL
021734	000240				NOP	:START OF TEST
021736	000240				MOV #STACK,SP	:INITIALIZE STACK POINTER
021740	012706	001100			MOV \$BASE,R0	:R0 = UNIBUS ADDRESS
021744	013700	001276			MOV TSTQUE,R1	:(R1) = DEVICE BEING TESTED
021750	013701	001466			MOV #13,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
021754	012737	000013	001226			

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1492

1493

1494

1495

1496

1497

1498

1499

1500

1501

1502

1503

1504

1505

1506

1507

1508

1509

1510

1511

1512

1513

1514

1515

1516

 :LOOP #1 READ,WRITE CHECK

:LOAD PARAMETERS AND GENERATE DATA BUFFER

1\$:	MOV LSTRK,RMDAO	:SET LAST TRACK AND
	MOVB #30,RMDAO	:LAST SECTOR
	MOV #FMT16,RMOFO	:16 BIT FORMAT
2\$:	MOV #559,RMDCO	:CYLINDER = 559.
	MOV #-256.*2,RMWCO	:CHANGE WORD COUNT
	MOV #BUFONE,RMBAO	:CHANGE BUS ADDRSS
	MOV #RD,RMCS10	:READ DATA COMMAND

: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
	:GO TO 4\$ IF NO ERROR
	:RETURN HERE IF ERROR
	:ERROR # DEFINED BY TSTPRP SUBROUTINE
	:GO TO 19\$ IF ERROR

4\$:

:READ DATA FROM THE DRIVE

5\$:	MOV #RD!GO,RMCS10	:READ DATA COMMAND
	MOV #PUTINX,R2	:LOAD PUT REGISTER INDEX TABLE
	MOVB #RMDA,(R2)+	
	MOVB #RMDC,(R2)+	
	MOVB #RMOF,(R2)+	
	MOVB #RMBA,(R2)+	
	MOVB #RMWC,(R2)+	
	MOVB #RMCS1,(R2)+	
	MOVB #200,(R2)+	:TERMINATE TABLE

JSR PC,PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 6\$:GO TO 6\$ IF NO ERROR
NOP	:RETURN HERE IF ERROR

```

022132 104000          EMT          :ERROR # DEFINED BY PUT SUBROUTINE
022134 000137 022546   JMP          19$          :GO TO 19$ IF ERROR
1517 022140          6$:
1518
1519          :SETUP GET INDEX TABLE TO READ ALL REGISTERS
022140 004737 040024   JSR          PC,GETSTS    :GO TO GETSTS SUBROUTINE
1520
1521          :WAIT FOR COMMAND TO COMPLETE
022144 004737 040732   JSR          PC,TIMOUT    :GO TO TIMOUT SUBROUTINE
1522
1523          :GO READ STATUS FOR READ COMMAND
1524 022150 004737 040110   JSR          PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
022154 000404          BR          7$          :GO TO 7$ IF NO ERROR
022156 000240          NOP          :RETURN HERE IF ERROR
022160 104000          EMT          :ERROR # DEFINED BY GET SUBROUTINE
022162 000137 022546   JMP          19$          :GO TO 19$ IF ERROR
1525
1526          :CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
1527          :DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
1528 022166          7$:
1529 022166 032737 000040 001400   BIT          #SSE,RMER2I  :IS 'SSE' BIT SET ?
1530 022174 001030          BNE          10$          :BR IF YES
1531
1532          :CHECK FOR ERRORS DURING READ OPERATION
1533 022176 004737 041126   JSR          PC,PRIERR    :GO CHECK FOR PRIMARY ERRORS
022202 000405          BR          8$          :GO TO 8$ IF NO ERROR
022204 000240          NOP          :RETURN HERE IF ERROR
022206 104000          EMT          :ERROR # DEFINED BY PRIERR SUBROUTINE
022210 004736          JSR          PC,@(SP)+   :GO BACK FOR MORE ERROR CHECKS
022212 000137 022546   JMP          19$          :GO TO 19$ IF ERROR
1534 022216          8$:
1535 022216 004737 054166   JSR          PC,DTASTS    :GO VERIFY RESULTS OF DATA TRANSFER
022222 000405          BR          9$          :GO TO 9$ IF NO ERROR
022224 000240          NOP          :RETURN HERE IF ERROR
022226 104000          EMT          :ERROR # DEFINED BY DTASTS SUBROUTINE
022230 004736          JSR          PC,@(SP)+   :GO BACK FOR MORE ERROR CHECKS
022232 000137 022546   JMP          19$          :GO TO 19$ IF ERROR
1536 022236          9$:
1537 022236 004737 041760   JSR          PC,SECERR    :GO CHECK FOR SECONDARY ERROR
022242 000405          BR          10$         :GO TO 10$ IF NO ERROR
022244 000240          NOP          :RETURN HERE IF ERROR
022246 104000          EMT          :ERROR # DEFINED BY SECERR SUBROUTINE
022250 004736          JSR          PC,@(SP)+   :GO BACK FOR MORE ERROR CHECKS
022252 000137 022546   JMP          19$          :GO TO 19$ IF ERROR
1538 022256          10$:
1539
1540          :CHANGE LOOP ADDRESSES
1541 022256 012737 022266 001124   MOV          #11$,$LPERR
1542 022264 000410          BR          12$          :SKIP TO NEXT OPERATION
1543
1544          :*****
1545          :LOOP #2          WRITE CHECK DATA
1546
1547 022266          11$:
1548
1549          :PREPARE DEVICE FOR WRITE CHECK OPERATION
1550 022266 004737 034032   JSR          PC,TSTPRP   :PREPARE DEVICE FOR TEST
  
```



```

022272 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 12$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 19$ IF ERROR

1551 022274 000404 BR 12$
1552 022276 000240 NOP
1553 022300 104000 EMT
1554 022302 000137 022546 JMP 19$
1555 022306 012737 000051 001412 12$:
;WRITE CHECK DATA FROM DEVICE
13$:
MOV #WCD!GO, RMCS10 ;READ DATA COMMAND
MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
MOVB #RMDA, (R2)+
MOVB #RMOF, (R2)+
MOVB #RMDC, (R2)+
MOVB #RMBA, (R2)+
MOVB #RMWC, (R2)+
MOVB #RMCS1, (R2)+
MOVB #200, (R2)

1565 022354 004737 040360 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
022360 000404 BR 14$ ;GO TO 14$ IF NO ERROR
022362 000240 NOP ;RETURN HERE IF ERROR
022364 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
022366 000137 022546 JMP 19$ ;GO TO 19$ IF ERROR

1566 022372 14$:
;SETUP GET INDEX TABLE TO READ ALL REGISTERS
1567 JSR PC, GETSTS ;GO TO GETSTS SUBROUTINE
1568 022372 004737 040024

;WAIT FOR COMMAND TO COMPLETE
1569 JSR PC, TIMEOUT ;GO TO TIMEOUT SUBROUTINE
1570 022376 004737 040732

;GO READ STATUS FOR READ OPERATION
1571 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
1572 022402 004737 040110 BR 15$ ;GO TO 15$ IF NO ERROR
022406 000404 NOP ;RETURN HERE IF ERROR
022410 000240 EMT ;ERROR # DEFINED BY GET SUBROUTINE
022412 104000 JMP 19$ ;GO TO 19$ IF ERROR
022414 000137 022546

;CHECK TO SEE IF THIS SECTOR WAS SKIP SECTORED FIRST, IF SO
;DO NOT EXECUTE THIS TEST IN 16 BIT FORMAT MODE
1574 15$:
1575 BIT #SSE, RMER2I ;IS 'SSE' BIT SET ?
1576 BNE 18$ ;BR IF YES
1577 022420 032737 000040 001400
1578 022420 001030
1579 022426

;CHECK FOR ERRORS DURING READ OPERATION
1580 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
1581 022430 004737 041126 BR 16$ ;GO TO 16$ IF NO ERROR
022434 000405 NOP ;RETURN HERE IF ERROR
022436 000240 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
022440 104000
  
```



```

022442 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022444 000137 022546 JMP 19$ ;GO TO 19$ IF ERROR
1583 022450 16$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
1584 022450 004737 054166 BR 17$ ;GO TO 17$ IF NO ERROR
022454 000405 NOP ;RETURN HERE IF ERROR
022456 000240 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
022460 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022462 004736 JMP 19$ ;GO TO 19$ IF ERROR
1585 022470 17$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
1586 022470 004737 041760 BR 18$ ;GO TO 18$ IF NO ERROR
022474 000405 NOP ;RETURN HERE IF ERROR
022476 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
022500 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022502 004736 JMP 19$ ;GO TO 19$ IF ERROR
022504 000137 022546
1587 022510 18$: BIT #SSEI,RMOFO ;TEST 16 BIT MODE W/ SSEI SET ?
1588 022510 032737 001000 001444 BNE 19$ ;YES !!
1589 022516 001013 MOV LSTRK,RMDAO ;SET LAST TRACK AND
1590 022520 013737 001334 001420 MOVB #31,RMDAO ;LAST SECTOR
1591 022526 112737 000037 001420 BIS #SSEI,RMOFO ;SET SSEI BIT IN RMOF AND
1592 022534 052737 001000 001444 JMP 2$ ;TEST AGAIN.
1593 022542 000137 022004
1594 022546
1595
1596
:*****
:*TEST 14 WRITE, READ W/ HCE ERROR
:*****
TST14:
022546 000004 SCOPE ;SCOPE CALL
022550 000240 NOP ;START OF TEST
022552 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
022556 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
022562 013701 001466 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
022566 012737 000014 001226 MOV #14,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1597
1598
1599
1600
:*****
:LOOP #1 FORMAT,WRITE,READ
1601 022574 012704 000000 MOV #0,R4 ;R4 = 1ST HEADER WORD
1602 022600 012703 000001 1$: MOV #1,R3 ;R3 = HCE BIT
1603 022604 2$:
1604
1605
:PREPARE THE DEVICE FOR FORMAT OPERATION
1606 022604 004737 034032 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
022610 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
022612 000404 BR 3$ ;GO TO 3$ IF NO ERROR
022614 000240 NOP ;RETURN HERE IF ERROR
022616 104000 EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
022620 000137 024416 JMP 47$ ;GO TO 47$ IF ERROR
  
```

T14 WRITE, READ W/ HCE ERROR

```

1607
1608           ;LOAD PARAMETERS AND GENERATE DATA BUFFER
1609 022624           3$:
1610 022624 012737 001057 001446      MOV      #559.,RMDCO      ;CYLINDER = 559.
1611 022632 012737 001000 001420      MOV      #TA2,RMDAO      ;TRACK = 2, SECTOR = 0
1612 022640 012737 104254 001416      MOV      #BUFONE,RMBAO   ;BUS ADDRESS
1613 022646 012737 177376 001414      MOV      #-258.,RMWCO    ;2 + 256. WORDS (2'S COMP)
1614 022654 012737 010000 001444      MOV      #FMT16,RMFO    ;16 BIT FORMAT
1615 022662 012737 000063 001412      MOV      #WH!GO,RMCS10  ;WRITE HEADER AND DATA COMMAND
1616
1617           ;VERIFY THAT SECTOR IS NOT BAD
           JSR      PC,BADSCT ;CALL BAD SECTOR MODULE
           BR      4$        ;GO TO 4$ IF NO ERROR
           TYPE     ,SCTMSG   ;TYPE BAD SECTOR MESSAGE
           EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
           JMP      46$      ;GO TO 46$ IF ERROR
1618 022670 004737 034762
           JSR      PC,BADSCT ;CALL BAD SECTOR MODULE
           BR      4$        ;GO TO 4$ IF NO ERROR
           TYPE     ,SCTMSG   ;TYPE BAD SECTOR MESSAGE
           EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
           JMP      46$      ;GO TO 46$ IF ERROR
1619 022710 012737 067456 001174      MOV      #ZEROS,$TMP0    ;STARTING ADDRESS OF PATTERN
1620 022716 012737 000001 001176      MOV      #1,$TMP1       ;RANGE OF PATTERN
1621 022724 004737 037214              JSR      PC,GENBUF      ;GO GENERATE BUFFER FOR FORMAT
1622
1623           ;CHANGE HEADER WORD TO FORCE HCE DURING WRITE & READ
           BIT      R3,BUFONE(R4) ;SET OR RESET FOR HCE??
           BEQ      5$
           BIC      R3,BUFONE(R4) ;RESET BIT FOR HCE
           BR      6$
1624 022730 030364 104254
           BIS      R3,BUFONE(R4) ;SET FOR HCE
1625 022734 001403
1626 022736 040364 104254
1627 022742 000402
1628 022744 050364 104254
           BIS      R3,BUFONE(R4) ;SET FOR HCE
1629
1630           ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
           6$:
1631 022750              MOV      #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
1632 022750 012702 001555      MOV      #RMDC,(R2)+
1633 022754 112722 000034      MOV      #RMDA,(R2)+
1634 022760 112722 000006      MOV      #RMBA,(R2)+
1635 022764 112722 000004      MOV      #RMWC,(R2)+
1636 022770 112722 000002      MOV      #RMOF,(R2)+
1637 022774 112722 000032      MOV      #RMCS1,(R2)+
1638 023000 112722 000000      MOV      #RMCST,(R2)+
1639 023004 112712 000200      MOV      #200,(R2)     ;TERMINATE TABLE
1640 023010
           7$:
1641 023010 004737 040360      JSR      PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
           BR      8$        ;GO TO 8$ IF NO ERROR
           NOP      ;RETURN HERE IF ERROR
           EMT      ;ERROR # DEFINED BY PUT SUBROUTINE
           JMP      46$      ;GO TO 46$ IF ERROR
1642 023014 000404
1643 023016 000240
1644 023020 104000
           8$:
1645 023022 000137 024416      JMP      46$
1646 023026 004737 040024
           ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
           JSR      PC,GETSTS ;GO TO GETSTS SUBROUTINE
1647
1648           ;WAIT FOR COMMAND TO COMPLETE
           JSR      PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1649
           ;GO READ STATUS FOR FORMAT OPERATION
           JSR      PC,GET    ;GO READ REGISTER(S) WITH GET SUBROUTINE
           BR      9$        ;GO TO 9$ IF NO ERROR
           NOP      ;RETURN HERE IF ERROR
           EMT      ;ERROR # DEFINED BY GET SUBROUTINE
1649 023032 004737 040732
           JSR      PC,GET    ;GO READ REGISTER(S) WITH GET SUBROUTINE
           BR      9$        ;GO TO 9$ IF NO ERROR
           NOP      ;RETURN HERE IF ERROR
           EMT      ;ERROR # DEFINED BY GET SUBROUTINE
1649 023036 004737 040110
           JSR      PC,GET    ;GO READ REGISTER(S) WITH GET SUBROUTINE
           BR      9$        ;GO TO 9$ IF NO ERROR
           NOP      ;RETURN HERE IF ERROR
           EMT      ;ERROR # DEFINED BY GET SUBROUTINE
1649 023042 000404
1649 023044 000240
1649 023046 104000

```



```

1650 023050 000137 024416          9$:      JMP      46$          :GO TO 46$ IF ERROR
1651 023054
1652
1653 023054 004737 054166          :VERIFY NO ERRORS DURING FORMAT
      023060 000405          JSR      PC,DTASTS      :GO VERIFY RESULTS OF DATA TRANSFER
      023062 000240          BR       10$           :GO TO 10$ IF NO ERROR
      023064 104000          NOP
      023066 004736          EMT      :RETURN HERE IF ERROR
      023070 000137 024416          JSR      PC,@(SP)+     :ERROR # DEFINED BY DTASTS SUBROUTINE
      023074          JMP      46$           :GO BACK FOR MORE ERROR CHECKS
      :GO TO 46$ IF ERROR

1654 023074          10$:
1655
1656          :MOVE LOOP ADDRESSES TO NEXT OPERATION
1657 023074 012737 023104 001124      MOV      #11$, $LPERR
1658 023102 000410          BR       12$           :SKIP TO WRITE OPERATION
1659
1660          :*****
1661          :LOOP #2      WRITE,READ
1662
1663 023104          11$:
1664
1665          :PREPARE DEVICE FOR WRITE OPERATION
1666 023104 004737 034032      JSR      PC,TSTPRP     :PREPARE DEVICE FOR TEST
      023110 154130          .WORD   154130        :TASK DESCRIPTOR AS FOLLOWS:
      :SELECT DEVICE & VERIFY DEVICE AVAILABLE
      :CLEAR CONTROLLER & SELECT DEVICE
      :VERIFY CONTROLLER CLEAR OPERATION
      :PACK ACKNOWLEDGE IF VOLUME NOT VALID
      :VERIFY PACK ACKNOWLEDGE
      :RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      :VERIFY RECALIBRATION
      :GO TO 12$ IF NO ERROR
      :RETURN HERE IF ERROR
      :ERROR # DEFINED BY TSTPRP SUBROUTINE
      :GO TO 46$ IF ERROR

      023112 000404          BR       12$
      023114 000240          NOP
      023116 104000          EMT
      023120 000137 024416          JMP      46$
1667 023124          12$:
1668
1669          :WRITE DATA TO THE DRIVE
1670 023124          13$:
1671 023124 012737 104260 001416      MOV      #BUFONE+4,RMBAD :CHANGE BUS ADDRESS
1672 023132 012737 177400 001414      MOV      #-256.,RMWCO   :CHANGE WORD COUNT
1673 023140 012737 000061 001412      MOV      #WD!GO,RMCS10  :WRITE DATA COMMAND
1674 023146 012702 001555          MOV      #PUTINX,R2     :LOAD PUT REGISTER INDEX TABLE
1675 023152 112722 000006          MOVB    #RMDA,(R2)+
1676 023156 112722 000034          MOVB    #RMDC,(R2)+
1677 023162 112722 000032          MOVB    #RMOF,(R2)+
1678 023166 112722 000004          MOVB    #RMBA,(R2)+
1679 023172 112722 000002          MOVB    #RMWC,(R2)+
1680 023176 112722 000000          MOVB    #RMCS1,(R2)+
1681 023202 112722 000200          MOVB    #200,(R2)+
      :TERMINATE TABLE
1682
1683 023206 004737 040360          JSR      PC,PUT        :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      023212 000404          BR       14$          :GO TO 14$ IF NO ERROR
      023214 000240          NOP
      023216 104000          EMT      :RETURN HERE IF ERROR
      023220 000137 024416          JMP      46$          :ERROR # DEFINED BY PUT SUBROUTINE
      :GO TO 46$ IF ERROR
1684 023224          14$:

```



```

1685
1686      023224 004737 040024      :SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR      PC,GETSTS      :GO TO GETSTS SUBROUTINE
1687
1688      023230 004737 040732      :WAIT FOR COMMAND TO COMPLETE
      JSR      PC,TIMOUT      :GO TO TIMEOUT SUBROUTINE
1689
1690      :GO READ STATUS FOR WRITE COMMAND
1691      023234 004737 040110      JSR      PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
      023240 000404      BR      15$      :GO TO 15$ IF NO ERROR
      023242 000240      NOP      :RETURN HERE IF ERROR
      023244 104000      EMT      :ERROR # DEFINED BY GET SUBROUTINE
      023246 000137 024416      JMP      46$      :GO TO 46$ IF ERROR
1692      023252      15$:
1693
1694      :CHECK FOR ERRORS DURING WRITE OPERATION
1695      023252 004737 041126      JSR      PC,PRIERR      :GO CHECK FOR PRIMARY ERRORS
      023256 000405      BR      16$      :GO TO 16$ IF NO ERROR
      023260 000240      NOP      :RETURN HERE IF ERROR
      023262 104000      EMT      :ERROR # DEFINED BY PRIERR SUBROUTINE
      023264 004736      JSR      PC,@(SP)+      :GO BACK FOR MORE ERROR CHECKS
      023266 000137 024416      JMP      46$      :GO TO 46$ IF ERROR
1696
1697      :DECODE THE TYPE OF ERROR PRODUCED BY THE BIT POSITION OF R3
1698      023272      16$:
1699      023272 005704      TST      R4      :IS THIS THE FIRST HEADER WORD ?
1700      023274 001011      BNE      17$      :NO !!
1701
1702      023276 032703 020000      BIT      #SSF,R3      :SHOULD SSE BE SET ?
1703      023302 001042      BNE      20$      :YES !!
1704      023304 032703 010000      BIT      #FMT16,R3      :SHOULD FER BE SET ?
1705      023310 001067      BNE      22$      :YES !!
1706      023312 032703 140000      BIT      #MSE!USE,R3      :SHOULD BSE BE SET ?
1707      023316 001111      BNE      24$      :YES !!
1708
1709      :VERIFY THAT A HEADER COMPARE ERROR WAS DETECTED
1710      023320      17$:
1711      023320 032737 000200 001352      BIT      #HCE,RMER1I      :WAS HCE DETECTED ??
1712      023326 001132      BNE      26$      :YES !!
1713
1714      023330 004737 054166      JSR      PC,DTASTS      :GO VERIFY RESULTS OF DATA TRANSFER
      023334 000405      BR      18$      :GO TO 18$ IF NO ERROR
      023336 000240      NOP      :RETURN HERE IF ERROR
      023340 104000      EMT      :ERROR # DEFINED BY DTASTS SUBROUTINE
      023342 004736      JSR      PC,@(SP)+      :GO BACK FOR MORE ERROR CHECKS
      023344 000137 024416      JMP      46$      :GO TO 46$ IF ERROR
1715      023350      18$:
1716      023350 013737 001352 001142      MOV      RMER1I,SBDDAT      :RECEIVED STATUS
1717      023356 012737 000200 001140      MOV      #HCE,$GDDAT      :EXPECTED STATUS
1718      023364 010437 001174      MOV      R4,STMP0      :R4 = HEADER WORD NUMBER AND
1719      023370 001002      BNE      19$      :ADJUST NUMBER.
1720      023372 005237 001174      INC      STMP0
1721      023376 010337 001176      19$: MOV      R3,STMP1      :R3 = BIT POSIITON
1722      023402 104344      EMT      344
1723      023404 000137 024416      JMP      46$
1724
1725      :VERIFY THAT SSE IS SET

```

```
1726 023410          20$:
1727 023410 032737 000040 001400 BIT #SSE,RMER2I ;IS SSE SET ?
1728 023416 001076          BNE 26$ ;YES !!
1729
1730 023420 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    023424 000405          BR 21$ ;GO TO 21$ IF NO ERROR
    023426 000240          NOP ;RETURN HERE IF ERROR
    023430 104000          EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
    023432 004736          JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    023434 000137 024416 JMP 46$ ;GO TO 46$ IF ERROR
1731 023440
1732 023440 013737 001400 001142 21$: MOV RMER2I,$BDDAT ;GET RECIEVED STATUS
1733 023446 013737 001400 001140 MOV RMER2I,$GDDAT ;GET EXPECTED STATUS AND
1734 023454 052737 000040 001140 BIS #SSE,$GDDAT ;SET SSE.
1735 023462 104224          EMT 224
1736 023464 000137 024416 JMP 46$
1737
1738 ;VERIFY THAT A FORMAT ERROR WAS DETECTED
1739 023470          22$:
1740 023470 032737 000020 001352 BIT #FER,RMER1I ;WAS FER DETECTED ??
1741 023476 001046          BNE 26$ ;YES !!
1742
1743 023500 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    023504 000405          BR 23$ ;GO TO 23$ IF NO ERROR
    023506 000240          NOP ;RETURN HERE IF ERROR
    023510 104000          EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
    023512 004736          JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    023514 000137 024416 JMP 46$ ;GO TO 46$ IF ERROR
1744 023520
1745 023520 012737 000020 001140 23$: MOV #FER,$GDDAT ;EXPECTED STATUS
1746 023526 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
1747 023534 104343          EMT 343
1748 023536 000137 024416 JMP 46$
1749
1750 ;VERIFY THAT A BAD SECTOR ERROR WAS DETECTED
1751 023542          24$:
1752 023542 032737 100000 001400 BIT #BSE,RMER2I ;WAS BSE DETECTED ??
1753 023550 001021          BNE 26$ ;YES !!
1754
1755 023552 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    023556 000405          BR 25$ ;GO TO 25$ IF NO ERROR
    023560 000240          NOP ;RETURN HERE IF ERROR
    023562 104000          EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
    023564 004736          JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    023566 000137 024416 JMP 46$ ;GO TO 46$ IF ERROR
1756 023572
1757 023572 013737 001400 001142 25$: MOV RMER2I,$BDDAT ;RECEIVED STATUS
1758 023600 012737 100000 001140 MOV #BSE,$GDDAT ;EXPECTED STAIUS
1759 023606 104345          EMT 345
1760 023610 000137 024416 JMP 46$
1761
1762 ;CHECK FOR OTHER ERRORS
1763 023614          26$:
1764 023614 004737 041760 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    023620 000405          BR 27$ ;GO TO 27$ IF NO ERROR
    023622 000240          NOP ;RETURN HERE IF ERROR
    023624 104000          EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
```



```

1765 023626 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      023630 000137 024416    JMP    46$           ;GO TO 46$ IF ERROR
1766 023634
1767
1768 023634 012737 023642 001124 ;CHANGE LOOP ADDRESSES
      MOV    #28$,SLPERR
1769
1770 ::*****
1771 ;LOOP #3      READ
1772
1773 023642 28$:
1774
1775 ;PREPARE DEVICE FOR READ OPERATION
1776 023642 004737 034032    JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      023646 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 29$ IF NO ERROR
                          ;RETURN HERE IF ERROR
                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                          ;GO TO 46$ IF ERROR
      BR     29$
      NOP
      EMT
      JMP    46$
1777 023650 000404          BR     29$
1778 023652 000240          NOP
1779 023654 104000          EMT
1780 023656 000137 024416    JMP    46$
1781 023662 29$:
1782 023662 012737 105264 001416 ;READ DATA FROM DEVICE
1783 023670 012737 177400 001414 30$:
1784 023676 012737 000071 001412    MOV    #BUFTWO+4,RMBAO ;CHANGE BUS ADDRESS
1785 023704 012702 001555          MOV    #-256.,RMWCO    ;CHANGE WORD COUNT
1786 023710 112722 000006          MOV    #RD!GO,RMCS10  ;READ DATA COMMAND
1787 023714 112722 000032          MOV    #PUTINX,R2     ;LOAD PUT REGISTER INDEX TABLE
1788 023720 112722 000034          MOVB  #RMDA,(R2)+
1789 023724 112722 000004          MOVB  #RMOF,(R2)+
1790 023730 112722 000002          MOVB  #RMDC,(R2)+
1791 023734 112722 000000          MOVB  #RMDA,(R2)+
1792 023740 112712 000200          MOVB  #RMWC,(R2)+
1793 023744 004737 040360          MOVB  #RMCS1,(R2)+
1794 023750 000404          MOVB  #200,(R2)
1795 023752 000240          JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
1796 023754 104000          BR     31$          ;GO TO 31$ IF NO ERROR
1797 023756 000137 024416    NOP              ;RETURN HERE IF ERROR
1798 023762 004737 040024    EMT              ;ERROR # DEFINED BY PUT SUBROUTINE
1799 023762 004737 040024    JMP    46$          ;GO TO 46$ IF ERROR
1800
1801 023762 004737 040024 31$:
      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR    PC,GETSTS ;GO TO GETSTS SUBROUTINE
1802
1803 023766 004737 040732 ;WAIT FOR COMMAND TO COMPLETE
      JSR    PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1804
1805 ;GO READ STATUS FOR READ OPERATION
1806 023772 004737 040110    JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
  
```


T14

WRITE, READ W/ HCE ERROR

023776	000404			BR	32\$:GO TO 32\$ IF NO ERROR
024000	000240			NOP		:RETURN HERE IF ERROR
024002	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
024004	000137	024416		JMP	46\$:GO TO 46\$ IF ERROR
1802	024010				32\$:	
1803						
1804						
1805	024010	004737	041126	JSR	PC,PRIERR	:CHECK FOR ERRORS DURING READ OPERATION :GO CHECK FOR PRIMARY ERRORS
	024014	000405		BR	33\$:GO TO 33\$ IF NO ERROR
	024016	000240		NOP		:RETURN HERE IF ERROR
	024020	104000		EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	024022	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	024024	000137	024416	JMP	46\$:GO TO 46\$ IF ERROR
1806	024030				33\$:	
1807	024030	005704		TST	R4	:IS THIS THE FIRST HEADER WORD ?
1808	024032	001011		BNE	34\$:NO !!
1809						
1810	024034	032703	020000	BIT	#SSF,R3	:SHOULD SSE BE SET ?
1811	024040	001042		BNE	37\$:YES !!
1812	024042	032703	010000	BIT	#FMT16,R3	:SHOULD FER BE SET ?
1813	024046	001067		BNE	39\$:YES !!
1814	024050	032703	140000	BIT	#MSE!USE,R3	:SHOULD BSE BE SET ?
1815	024054	001111		BNE	41\$:YES !!
1816						
1817						
1818	024056					:VERIFY THAT A HEADER COMPARE ERROR WAS DETECTED
1819	024056	032737	000200	001352	34\$:	
1820	024064	001132		BIT	#HCE,RMER11	:WAS HCE DETECTED ??
1821				BNE	43\$:YES !!
1822	024066	004737	054166	JSR	PC,DTASTS	:GO VERIFY RESULTS OF DATA TRANSFER
	024072	000405		BR	35\$:GO TO 35\$ IF NO ERROR
	024074	000240		NOP		:RETURN HERE IF ERROR
	024076	104000		EMT		:ERROR # DEFINED BY DTASTS SUBROUTINE
	024100	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	024102	000137	024416	JMP	46\$:GO TO 46\$ IF ERROR
1823	024106				35\$:	
1824	024106	013737	001352	001142		:RECEIVED STATUS
1825	024114	012737	000200	001140		:EXPECTED STATUS
1826	024122	010437	001174			:R4 = HEADER WORD NUMBER AND
1827	024126	001002				:ADJUST NUMBER.
1828	024130	005237	001174			
1829	024134	010337	001176		36\$:	:R3 = BIT POSITION
1830	024140	104344		MOV	R3,STMP1	
1831	024142	000137	024416	EMT	344	
1832				JMP	46\$	
1833						
1834	024146					:VERIFY THAT SSE BIT IS SET
1835	024146	032737	000040	001400	37\$:	
1836	024154	001076		BIT	#SSE,RMER21	:IS SSE SET ?
1837				BNE	43\$:YES !!
1838	024156	004737	054166	JSR	PC,DTASTS	:GO VERIFY RESULTS OF DATA TRANSFER
	024162	000405		BR	38\$:GO TO 38\$ IF NO ERROR
	024164	000240		NOP		:RETURN HERE IF ERROR
	024166	104000		EMT		:ERROR # DEFINED BY DTASTS SUBROUTINE
	024170	004736		JSR	PC,@(SP)+	:GO BACK FOR MORE ERROR CHECKS
	024172	000137	024416	JMP	46\$:GO TO 46\$ IF ERROR
1839	024176				38\$:	

T14 WRITE, READ W/ HCE ERROR

```

1840 024176 013737 001400 001142      MOV      RMER2I,$BDDAT      :GET RECIEVED STATUS
1841 024204 013737 001400 001140      MOV      RMER2I,$GDDAT      :GET EXPECTED STATUS AND
1842 024212 052737 000040 001140      BIS      #SSE,$GDDAT        :SET SSE.
1843 024220 104224                      EMT      224
1844 024222 000137 024416                      JMP      46$
1845
1846                                     :VERIFY THAT A FORMAT ERROR WAS DETECTED
1847 024226                                     39$:
1848 024226 032737 000020 001352      BIT      #FER,RMER1I        :WAS FER DETECTED ??
1849 024234 001046                      BNE      43$                :YES !!
1850
1851 024236 004737 054166                      JSR      PC,DTASTS          :GO VERIFY RESULTS OF DATA TRANSFER
                                024242 000405                      BR       40$                :GO TO 40$ IF NO ERROR
                                024244 000240                      NOP                      :RETURN HERE IF ERROR
                                024246 104000                      EMT                      :ERROR # DEFINED BY DTASTS SUBROUTINE
                                024250 004736                      JSR      PC,@(SP)+          :GO BACK FOR MORE ERROR CHECKS
                                024252 000137 024416                      JMP      46$                :GO TO 46$ IF ERROR
1852 024256                                     40$:
1853 024256 012737 000020 001140      MOV      #FER,$GDDAT        :EXPECTED STATUS
1854 024264 013737 001352 001142      MOV      RMER1I,$BDDAT      :RECEIVED STATUS
1855 024272 104343                      EMT      343
1856 024274 000137 024416                      JMP      46$
1857
1858                                     :VERIFY THAT A BAD SECTOR ERROR WAS DETECTED
1859 024300                                     41$:
1860 024300 032737 100000 001400      BIT      #BSE,RMER2I        :WAS BSE DETECTED ??
1861 024306 001021                      BNE      43$                :YES !!
1862
1863 024310 004737 054166                      JSR      PC,DTASTS          :GO VERIFY RESULTS OF DATA TRANSFER
                                024314 000405                      BR       42$                :GO TO 42$ IF NO ERROR
                                024316 000240                      NOP                      :RETURN HERE IF ERROR
                                024320 104000                      EMT                      :ERROR # DEFINED BY DTASTS SUBROUTINE
                                024322 004736                      JSR      PC,@(SP)+          :GO BACK FOR MORE ERROR CHECKS
                                024324 000137 024416                      JMP      46$                :GO TO 46$ IF ERROR
1864 024330                                     42$:
1865 024330 013737 001400 001142      MOV      RMER2I,$BDDAT      :RECEIVED STATUS
1866 024336 012737 100000 001140      MOV      #BSE,$GDDAT        :EXPECTED STAIUS
1867 024344 104345                      EMT      345
1868 024346 000137 000350                      JMP      350
1869
1870                                     :CHECK FOR OTHER ERRORS
1871 024352                                     43$:
1872 024352 004737 041760                      JSR      PC,SECERR          :GO CHECK FOR SECONDARY ERRORS
                                024356 000405                      BR       44$                :GO TO 44$ IF NO ERROR
                                024360 000240                      NOP                      :RETURN HERE IF ERROR
                                024362 104000                      EMT                      :ERROR # DEFINED BY SECERR SUBROUTINE
                                024364 004736                      JSR      PC,@(SP)+          :GO BACK FOR MORE ERROR CHECKS
                                024366 000137 024416                      JMP      46$                :GO TO 46$ IF ERROR
1873 024372                                     44$:
1874 024372 006303                      ASL      R3                  :SHIFT TO NEXT BIT
1875 024374 001006                      BNE      45$                :REPEAT IF NOT DONE
1876
1877 024376 005704                      TST      R4                  :SECOND HEADER WORD DONE??
1878 024400 001006                      BNE      46$                :YES!!
1879 024402 012704 000002                      MOV      #2,R4              :SETUP FOR 2ND HEADER WORD
1880 024406 012703 000001                      MOV      #1,R3              :SETUP FOR HCE
1881 024412                                     45$:

```


1882 024412 000137 022604
 1883 024416
 1884
 1885 024416
 1886
 1887

46\$: JMP 2\$
 47\$:

 :*TEST 15 WRITE, READ W/ HCI & SSEI

 TST15:

024416
 024416 000004
 024420 000240
 024422 012706 001100
 024426 013700 001276
 024432 013701 001466
 024436 012737 000015 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 #15,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

1888
 1889
 1890
 1891
 1892 024444 012704 000000
 1893 024450 012703 000001
 1894 024454
 1895
 1896
 1897 024454 004737 034032
 024460 154130

 :LOOP #1 FORMAT,WRITE,READ

1\$: MOV #0,R4 :HEADER WORD
 MOV #1,R3 :HCE BIT
 2\$:

: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 3\$ IF NO ERROR
 :RETURN HERE IF ERROR
 :ERROR # DEFINED BY TSTPRP SUBROUTINE
 :GO TO 38\$ IF ERROR

024462 000404
 024464 000240
 024466 104000
 024470 000137 025720

BR 3\$
 NOP
 EMT
 JMP 38\$

1898
 1899
 1900 024474
 1901 024474 012737 001057 001446
 1902 024502 012737 001000 001420
 1903 024510 012737 104254 001416
 1904 024516 012737 177376 001414
 1905 024524 012737 010000 001444
 1906 024532 012737 000063 001412
 1907
 1908

:LOAD PARAMETERS AND GENERATE DATA BUFFER

3\$: MOV #559,RMDCO :CYLINDER = 559.
 MOV #TA2,RMDAO :TRACK = 2, SECTOR = 0
 MOV #BUFONE,RMBAO :BUS ADDRESS
 MOV #-258,RMWCO :2 + 256. WORDS (2' COMP)
 MOV #FMT16,RMFOFO :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 4\$:GO TO 4\$ IF NO ERROR
 TYPE ,SCTMSG :TYPE BAD SECTOR MESSAGE
 EMT :ERROR # DEFINED BY BADSCT SUBROUTINE
 JMP 37\$:GO TO 37\$ IF ERROR

024540 004737 034762
 024544 000405
 024546 104401 066006
 024552 104000
 024554 000137 025720
 1909 024560
 1910 024560 012737 067414 001174
 1911 024566 012737 000001 001176
 1912 024574 004737 037214

4\$: MOV #ONES,\$TMP0 :STARTING ADDRESS OF PATTERN
 MOV #1,\$TMP1 :RANGE OF PATTERN
 JSR PC,GENBUF :GO GENERATE BUFFER FOR FORMAT


```

1913
1914 ;CHANGE HEADER WORD TO FORCE ERROR DURING WRITE
1915 024600 030364 104254 BIT R3,BUFONE(R4) ;SET OR RESET BIT??
1916 024604 001403 BEQ 5$
1917 024606 040364 104254 BIC R3,BUFONE(R4) ;RESET BIT
1918 024612 000402 BR 6$
1919 024614 050364 104254 5$: BIS R3,BUFONE(R4) ;SET BIT
1920
1921 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
1922 024620 6$:
1923 024620 012702 001555 MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
1924 024624 112722 000034 MOVB #RMDC,(R2)+
1925 024630 112722 000006 MOVB #RMDA,(R2)+
1926 024634 112722 000004 MOVB #RMBA,(R2)+
1927 024640 112722 000002 MOVB #RMWC,(R2)+
1928 024644 112722 000032 MOVB #RMOF,(R2)+
1929 024650 112722 000000 MOVB #RMC$1,(R2)+
1930 024654 112712 000200 MOVB #200,(R2) ;TERMINATE TABLE
1931 024660
1932
1933 ;FORMAT THE DRIVE
1934 024660 004737 040360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
024664 000404 BR 8$ ;GO TO 8$ IF NO ERROR
024666 000240 NOP ;RETURN HERE IF ERROR
024670 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
024672 000137 025720 JMP 37$ ;GO TO 37$ IF ERROR
1935 024676
1936
1937 ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
024676 004737 040024 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
1938
1939 ;WAIT FOR COMMAND TO COMPLETE
024702 004737 040732 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
1940
1941 ;GO READ STATUS FOR FORMAT OPERATION
1942 024706 004737 040110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
024712 000404 BR 9$ ;GO TO 9$ IF NO ERROR
024714 000240 NOP ;RETURN HERE IF ERROR
024716 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
024720 000137 025720 JMP 37$ ;GO TO 37$ IF ERROR
1943 024724
1944
1945 ;VERIFY NO ERRORS DURING FORMAT
1946 024724 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
024730 000405 BR 10$ ;GO TO 10$ IF NO ERROR
024732 000240 NOP ;RETURN HERE IF ERROR
024734 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
024736 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
024740 000137 025720 JMP 37$ ;GO TO 37$ IF ERROR
1947 024744
1948
1949 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
1950 024744 012737 024754 001124 MOV #11$, $LPERR
1951 024752 000410 BR 12$ ;SKIP TO WRITE OPERATION
1952
1953 ;*****
1954 ;LOOP #2 WRITE,READ
  
```

1955									
1956	024754					11\$:			
1957									
1958									
1959	024754	004737	034032						
	024760	154130							
	024762	000404							
	024764	000240							
	024766	104000							
	024770	000137	025720						
1960	024774								
1961									
1962									
1963	024774								
1964	024774	012737	013000	001444					
1965									
1966	025002	012737	000061	001412					
1967	025010	012737	104260	001416					
1968	025016	012702	001555						
1969	025022	112722	000006						
1970	025026	112722	000034						
1971	025032	112722	000032						
1972	025036	112722	000004						
1973	025042	112722	000002						
1974	025046	112722	000000						
1975	025052	112722	000200						
1976									
1977	025056	004737	040360						
	025062	000404							
	025064	000240							
	025066	104000							
	025070	000137	025720						
1978	025074								
1979									
1980									
	025074	004737	040024						
1981									
1982									
	025100	004737	040732						
1983									
1984									
1985	025104	004737	040110						
	025110	000404							
	025112	000240							
	025114	104000							
	025116	000137	025720						
1986	025122								
1987									
1988									
1989	025122	004737	041126						

```

11$:
:PREPARE DEVICE FOR WRITE OPERATION
      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      .WORD   154130          ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 12$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 37$ IF ERROR

      BR      12$
      NOP
      EMT
      JMP     37$

12$:
:WRITE DATA TO THE DRIVE
13$:
      MOV     #FMT16!HCI!SSEI,RMOFO ;INHIBIT HEADER COMPARE AND SKIP
                                ;SKIP SECTOR ERRORS
                                ;WRITE DATA COMMAND
                                ;LOAD STARTING BUFFER ADDRESS
                                ;LOAD PUT REGISTER INDEX TABLE

      MOV     #WD!GO,RMCS10
      MOV     #BUFONE+4,RMBAO
      MOV     #PUTINX,R2
      MOVB   #RMDA,(R2)+
      MOVB   #RMDC,(R2)+
      MOVB   #RMOF,(R2)+
      MOVB   #RMBA,(R2)+
      MOVB   #RMWC,(R2)+
      MOVB   #RMCS1,(R2)+
      MOVB   #200,(R2)+
                                ;TERMINATE TABLE

      JSR     PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR     14$              ;GO TO 14$ IF NO ERROR
      NOP
      EMT
      JMP     37$              ;GO TO 37$ IF ERROR

14$:
: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
      JSR     PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR     15$              ;GO TO 15$ IF NO ERROR
      NOP
      EMT
      JMP     37$              ;GO TO 37$ IF ERROR

15$:
:CHECK FOR ERRORS DURING WRITE OPERATION
      JSR     PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
  
```



```

025126 000405 BR 16$ ;GO TO 16$ IF NO ERROR
025130 000240 NOP ;RETURN HERE IF ERROR
025132 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
025134 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
025136 000137 JMP 37$ ;GO TO 37$ IF ERROR
1990 025142 005037 025720 16$: CLR $GDDAT ;EXPECTED STATUS
1991
1992 025146 032737 000220 001352 BIT #HCE!FER,RMER1I ;ANY ERROR??
1993 025154 001407 BEQ 17$
1994 025156 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
1995 025164 042737 177557 001142 BIC #^C<HCE!FER>,$BDDAT ;CLEAR DONT CARES
1996 025172 000412 BR 18$
1997 025174
1998 025174 013737 001400 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
1999 025202 032737 100000 001400 BIT #BSE,RMER2I ;ANY BAD SECTOR ERROR ?
2000 025210 001406 BEQ 19$ ;NO !!
2001 025212 042737 077777 001142 BIC #^CBSE,$BDDAT ;SAVE BSE FOR ERROR
2002 025220
2003 025220 104346 EMT 346
2004 025222 000137 025720 JMP 37$
2005 025226
2006 025226 032737 000040 001400 BIT #SSE,RMER2I ;ANY SKIP SECTOR ERROR ?
2007 025234 001406 BEQ 20$
2008 025236 043737 177737 001142 BIC ^CSSE,$BDDAT ;SAVE SSE FOR ERROR
2009 025244 104227 EMT 227
2010 025246 000137 025720 JMP 37$
2011 025252
2012 025252 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
025256 000405 BR 21$ ;GO TO 21$ IF NO ERROR
025260 000240 NOP ;RETURN HERE IF ERROR
025262 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
025264 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
025266 000137 025720 JMP 37$ ;GO TO 37$ IF ERROR
2013 025272
2014 025272 004737 041760 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
025276 000405 BR 22$ ;GO TO 22$ IF NO ERROR
025300 000240 NOP ;RETURN HERE IF ERROR
025302 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
025304 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
025306 000137 025720 JMP 37$ ;GO TO 37$ IF ERROR
2015 025312
2016
2017 ;CHANGE LOOP ADDRESSES
2018 025312 012737 025322 001124 MOV #23$,$LPERR
2019 025320 000410 BR 24$
2020
2021 ;*****
2022 ;LOOP #3 READ
2023
2024 025322
2025
2026
2027 025322 004737 034032 ;PREPARE DEVICE FOR READ OPERATION
025326 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 24$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 37$ IF ERROR
025330 000404 BR 24$
025332 000240 NOP
025334 104000 EMT
025336 000137 025720 JMP 37$
2028 025342 24$:
2029
2030 :READ DATA FROM DEVICE
2031 025342 25$:
2032 025342 012737 000071 001412 MOV #RD!GO,RMCS10 :READ DATA COMMAND
2033 025350 012737 105264 001416 MOV #BUFTWO+4,RMBAO :LOAD STARTING BUFFER ADDRESS
2034 025356 012702 001555 MOV #PUTINX,R2 :LOAD PUT REGISTER INDEX TABLE
2035 025362 112722 000006 MOVB #RMDA,(R2)+
2036 025366 112722 000032 MOVB #RMOF,(R2)+
2037 025372 112722 000034 MOVB #RMDC,(R2)+
2038 025376 112722 000004 MOVB #RMBA,(R2)+
2039 025402 112722 000002 MOVB #RMWC,(R2)+
2040 025406 112722 000000 MOVB #RMCS1,(R2)+
2041 025412 112712 000200 MOVB #200,(R2)
2042
2043 025416 004737 040360 JSR PC,PUT :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
025422 000404 BR 26$ :GO TO 26$ IF NO ERROR
025424 000240 NOP :RETURN HERE IF ERROR
025426 104000 EMT :ERROR # DEFINED BY PUT SUBROUTINE
025430 000137 025720 JMP 37$ :GO TO 37$ IF ERROR
2044 025434 26$:
2045
2046 :SETUP GET INDEX TABLE TO READ ALL REGISTERS
025434 004737 040024 JSR PC,GETSTS :GO TO GETSTS SUBROUTINE
2047
2048 :WAIT FOR COMMAND TO COMPLETE
025440 004737 040732 JSR PC,TIMOUT :GO TO TIMOUT SUBROUTINE
2049
2050 :GO READ STATUS FOR READ OPERATION
2051 025444 004737 040110 JSR PC,GET :GO READ REGISTER(S) WITH GET SUBROUTINE
025450 000404 BR 27$ :GO TO 27$ IF NO ERROR
025452 000240 NOP :RETURN HERE IF ERROR
025454 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
025456 000137 025720 JMP 37$ :GO TO 37$ IF ERROR
2052 025462 27$:
2053
2054 :CHECK FOR ERRORS DURING READ OPERATION
2055 025462 004737 041126 JSR PC,PRIERR :GO CHECK FOR PRIMARY ERRORS
025466 000405 BR 28$ :GO TO 28$ IF NO ERROR
025470 000240 NOP :RETURN HERE IF ERROR
025472 104000 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
025474 004736 JSR PC,@(SP)+ :GO BACK FOR MORE ERROR CHECKS
025476 000137 025720 JMP 37$ :GO TO 37$ IF ERROR
2056 025502 005037 001140 28$: CLR $GDDAT :EXPECTED STATUS
2057
2058 025506 032737 000220 001352 BIT #HCE!FER,RMER1I :ANY ERROR ?
2059 025514 001407 BEQ 29$ :NO!!
2060 025516 013737 001352 001142 MOV RMER1I,$BDDAT :RECEIVED STATUS
2061 025524 042737 177557 001142 BIC #^C<HCE!FER>,$BDDAT :CLEAR DONT CARES

```

```

2062 025532 000412          BR          30$
2063 025534                29$:
2064 025534 013737 001400 001142  MOV      RMER2I,$BDDAT      :RECEIVED STATUS
2065 025542 032737 100000 001400  BIT      #BSE,RMER2I      :ANY BAD SECTOR ERROR ?
2066 025550 001406                BEQ      31$              :NO !!
2067 025552 042737 077777 001142  BIC      #^CBSE,$BDDAT    :CLEAR DONT CARES
2068 025560                30$:
2069 025560 104346          EMT      346
2070 025562 000137 025720  JMP      37$
2071 025566                31$:
2072 025566 032737 000040 001400  BIT      #SSE,RMER2I      :ANY SKIP SECTOR ERROR ?
2073 025574 001406                BEQ      32$              :NO !!
2074 025576 043737 177737 001142  BIC      ^CSSE,$BDDAT    :SAVE SSE FOR ERROR
2075 025604 104227          EMT      227
2076 025606 000137 025720  JMP      37$
2077 025612                32$:
2078 025612 004737 054166  JSR      PC,DTASTS        :GO VERIFY RESULTS OF DATA TRANSFER
      025616 000405          BR      33$              :GO TO 33$ IF NO ERROR
      025620 000240          NOP
      025622 104000          EMT
      025624 004736          JSR      PC,@(SP)+        :ERROR # DEFINED BY DTASTS SUBROUTINE
      025626 000137 025720  JMP      37$              :GO BACK FOR MORE ERROR CHECKS
      :GO TO 37$ IF ERROR
2079 025632                33$:
2080 025632 004737 041760  JSR      PC,SECERR        :GO CHECK FOR SECONDARY ERRORS
      025636 000405          BR      34$              :GO TO 34$ IF NO ERROR
      025640 000240          NOP
      025642 104000          EMT
      025644 004736          JSR      PC,@(SP)+        :ERROR # DEFINED BY SECERR SUBROUTINE
      025646 000137 025720  JMP      37$              :GO BACK FOR MORE ERROR CHECKS
      :GO TO 37$ IF ERROR
2081 025652                34$:
2082 025652 004737 037462  JSR      PC,CMPBUF        :GO COMPARE WRITE, READ DATA BUFFERS
      025656 104260          .WORD  BUFOFF+4          :STARTING ADDRESS OF WRITE BUFFER
      025660 105264          .WORD  BUFTWO+4         :STARTING ADDRESS OF READ BUFFER
      025662 000404          BR      35$              :GO TO 35$ IF NO ERROR
      025664 000240          NOP
      025666 104000          EMT
      025670 000137 025720  JMP      37$              :ERROR # DEFINED BY CMPBUF SUBROUTINE
      :GO TO 37$ IF ERROR
2083 025674                35$:
2084
2085 025674 006303          ASL      R3                :SHIFT HCE BIT
2086 025676 001006          BNE     36$              :CONTINUE IF NOT DONE
2087 025700 005704          TST     R4                :SECOND HEADER DONE??
2088 025702 001006          BNE     37$              :YES!!
2089 025704 012703 000001  MOV      #1,R3            :START WITH BIT 0
2090 025710 012704 000002  MOV      #2,R4            :DO SECOND HEADER WORD
2091 025714                36$:
2092 025714 000137 024454  JMP      2$
2093 025720                37$:
2094
2095 025720                38$:
2096
2097
*****
:*TEST 16      WRITE, READ W/ IVC ERROR
*****
TST16:
025720          000004          SCOPE CALL
025720          000240          :START OF TEST
  
```



```

025724 012706 001100      MOV      #STACK,SP      ;INITIALIZE STACK POINTER
025730 013700 001276      MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
025734 013701 001466      MOV      TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
025740 012737 000016 001226  MOV      #16,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

2098
2099
2100      ;:*****
2101      ;:LOOP #1          WRITE,READ
2102      ;:LOAD PARAMETERS AND GENERATE DATA BUFFER.
2103 025746 1$:
2104 025746 012737 001057 001446      MOV      #559.,RMDCO     ;CYLINDER = 559.
2105 025754 012737 001000 001420      MOV      #TA2,RMDAO     ;TRACK = 2, SECTOR = 0
2106 025762 012737 104254 001416      MOV      #BUFONE,RMBAD  ;BUS ADDRESS
2107 025770 012737 177400 001414      MOV      #-256.,RMWCO   ;256. WORDS (2'S COMP)
2108 025776 012737 010000 001444      MOV      #FMT16,RMOFO   ;16 BIT FORMAT
2109 026004 012737 000060 001412      MOV      #WD,RMCS10     ;WRITE HEADER AND DATA COMMAND
2110
2111      ;:VERIFY THAT SECTOR IS NOT BAD
026012 004737 034762      JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
026016 000405      BR       2$             ;GO TO 2$ IF NO ERROR
026020 104401 066006      TYPE     .SCTMSG       ;TYPE BAD SECTOR MESSAGE
026024 104000      EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
026026 000137 026740      JMP      22$           ;GO TO 22$ IF ERROR
2112 026032 2$:
2113 026032 012737 001420 001174      MOV      #RMDAO,$TMP0   ;USE SECTOR FOR DATA PATTERN
2114 026040 012737 000001 001176      MOV      #1,$TMP1
2115 026046 004737 037214      JSR      PC,GENBUF     ;GO GENERATE DATA BUFFER
2116
2117      ;:PREPARE DEVICE FOR WRITE OPERATION
2118 026052 004737 034032      JSR      PC,TSTPRP     ;PREPARE DEVICE FOR TEST
026056 154130      .WORD   154130        ;TASK DESCRIPTOR AS FOLLOWS:
                                ;:SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;:CLEAR CONTROLLER & SELECT DEVICE
                                ;:VERIFY CONTROLLER CLEAR OPERATION
                                ;:PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;:VERIFY PACK ACKNOWLEDGE
                                ;:RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;:VERIFY RECALIBRATION
026060 000404      BR       3$             ;GO TO 3$ IF NO ERROR
026062 000240      NOP      ;RETURN HERE IF ERROR
026064 104000      EMT      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
026066 000137 026740      JMP      22$           ;GO TO 22$ IF ERROR
2119 026072 3$:
2120
2121      ;:RESET VOLUME VALID
2122 026072 112737 000024 001555      MOV      #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
026100 112737 000200 001556      MOV      #200,PUTINX+1 ;SET TERMINATOR BYTE
026106 012737 000001 001436      MOV      #DMD,RMMR1O   ;SET RMMR1 OUTPUT BUFFER = DMD
026114 004737 040360      JSR      PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
026120 000404      BR       4$             ;GO TO 4$ IF NO ERROR
026122 000240      NOP      ;RETURN HERE IF ERROR
026124 104000      EMT      ;ERROR DEFINED BY PUT SUBROUTINE
026126 000137 026740      JMP      22$           ;GO TO 22$ IF ERROR
2123 026132 4$:
2124
2125      ;:SETUP PARAMETERS AND EXECUTE WRITE DATA COMMAND
2126 026132 5$:

```



```

2127 026132 012737 000061 001412      MOV      #WD!GO,RMCS10      ;WRITE DATA COMMAND
2128 026140 012737 000000 001436      MOV      #0,RMMR10         ;RESET DIAGNOSTIC MODE
2129 026146 012702 001555              MOV      #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
2130 026152 112722 000024              MOV      #RMMR1,(R2)+
2131 026156 112722 000006              MOV      #RMDA,(R2)+
2132 026162 112722 000034              MOV      #RMDC,(R2)+
2133 026166 112722 000032              MOV      #RMOF,(R2)+
2134 026172 112722 000002              MOV      #RMWC,(R2)+
2135 026176 112722 000004              MOV      #RMB A,(R2)+
2136 026202 112722 000000              MOV      #RMCS1,(R2)+
2137 026206 112722 000200              MOV      #200,(R2)+
2138
2139 026212 004737 040360              JSR      PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026216 000404              BR       6$              ;GO TO 6$ IF NO ERROR
      026220 000240              NOP
      026222 104000              EMT
      026224 000137 026740              JMP      22$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 22$ IF ERROR
2140 026230      6$:
2141
2142      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2143 026230 004737 040024
2144      ;WAIT FOR COMMAND TO COMPLETE
      JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
2145 026234 004737 040732
2146      ;GO READ STATUS FOR WRITE COMMAND
2147 026240 004737 040110              JSR      PC,GET            ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026244 000404              BR       7$              ;GO TO 7$ IF NO ERROR
      026246 000240              NOP
      026250 104000              EMT
      026252 000137 026740              JMP      22$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 22$ IF ERROR
2148 026256      7$:
2149
2150      ;CHECK FOR ERRORS DURING WRITE OPERATION
2151 026256 004737 041126              JSR      PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      026262 000405              BR       8$              ;GO TO 8$ IF NO ERROR
      026264 000240              NOP
      026266 104000              EMT
      026270 004736              JSR      PC,@(SP)+        ;RETURN HERE IF ERROR
      026272 000137 026740              JMP      22$            ;ERROR # DEFINED BY PRIERR SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 22$ IF ERROR
2152 026276      8$:
2153 026276 032737 010000 001400              BIT      #IVC,RMER21      ;WAS "IVC" DETECTED??
2154 026304 001024              BNE      10$             ;YES!!
2155
2156 026306 004737 054166              JSR      PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      026312 000405              BR       9$              ;GO TO 9$ IF NO ERROR
      026314 000240              NOP
      026316 104000              EMT
      026320 004736              JSR      PC,@(SP)+        ;RETURN HERE IF ERROR
      026322 000137 026740              JMP      22$            ;ERROR # DEFINED BY DTASTS SUBROUTINE
      ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 22$ IF ERROR
2157 026326      9$:
2158 026326 013737 001400 001142              MOV      RMER21,$BDDAT    ;RECEIVED STATUS
2159 026334 013737 001400 001140              MOV      RMER21,$GDDAT    ;EXPECTED STATUS
2160 026342 052737 010000 001140              BIS      #IVC,$GDDAT
2161 026350 104342              EMT
2162 026352 000137 026740              JMP      22$
2163 026356      10$:

```

```

2164 026356 004737 041760      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      026362 000405          BR     11$           ;GO TO 11$ IF NO ERROR
      026364 000240          NOP                    ;RETURN HERE IF ERROR
      026366 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      026370 004736          JSR    PC,a(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      026372 000137 026740    JMP    22$           ;GO TO 22$ IF ERROR
2165 026376                    11$:
2166
2167
2168 026376 012737 026406 001124 ;CHANGE LOOP ADDRESSES
      026404 000410          MOV    #12$,$LPERR
      000000          BR     13$
2170
2171
2172
2173
2174 026406                    12$:
2175
2176
2177 026406 004737 034032    ;PREPARE DEVICE FOR READ OPERATION
      026412 154130          JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      000000          .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
      000000          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      000000          ;CLEAR CONTROLLER & SELECT DEVICE
      000000          ;VERIFY CONTROLLER CLEAR OPERATION
      000000          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      000000          ;VERIFY PACK ACKNOWLEDGE
      000000          ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      000000          ;VERIFY RECALIBRATION
      000000          ;GO TO 13$ IF NO ERROR
      000000          ;RETURN HERE IF ERROR
      000000          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      000000          ;GO TO 22$ IF ERROR
      000000          BR     13$
      000000          NOP
      000000          EMT
      000000          JMP    22$
2178 026426                    13$:
2179
2180
2181 026426 112737 000024 001555 ;RESET VOLUME VALID
      026434 112737 000200 001556 ;SETUP PUT INDEX TABLE
      026442 012737 000001 001436 ;SET TERMINATOR BYTE
      026450 004737 040360          MOV    #DMD,RMMR10 ;SET RMMR1 OUTPUT BUFFER = DMD
      026454 000404          JSR    PC,PUT        ;GO WRITE RMMR1 VIA PUT SUBROUTINE
      026456 000240          BR     14$           ;GO TO 14$ IF NO ERROR
      026460 104000          NOP                    ;RETURN HERE IF ERROR
      026462 000137 026740    EMT                    ;ERROR DEFINED BY PUT SUBROUTINE
      026466                    JMP    22$           ;GO TO 22$ IF ERROR
2182
2183
2184
2185 026466                    14$:
2186 026466 012737 000071 001412 ;READ DATA FROM DEVICE
      026474 012737 105260 001416 ;READ DATA COMMAND
      026502 012737 000000 001436 ;LOAD STARTING BUFFER ADDRESS
      026510 012702 001555          MOV    #0,RMMR10   ;RESET DIANOSTIC MODE
      026514 112722 000024          MOV    #PUTINX,R2  ;LOAD PUT REGISTER INDEX TABLE
      026520 112722 000006          MOVB  #RMMR1,(R2)+
      026524 112722 000032          MOVB  #RMDA,(R2)+
      026530 112722 000034          MOVB  #RMOF,(R2)+
      026534 112722 000004          MOVB  #RMDC,(R2)+
      026540 112722 000002          MOVB  #RMDA,(R2)+
      026544 112722 000000          MOVB  #RMWC,(R2)+
      000000          MOVB  #RMCS1,(R2)+

```


T16 WRITE, READ W/ IVC ERROR

```

2197 026550 112712 000200          MOVB   #200,(R2)
2198
2199 026554 004737 040360          JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026560 000404          BR     16$            ;GO TO 16$ IF NO ERROR
      026562 000240          NOP
      026564 104000          EMT
      026566 000137 026740          JMP    22$            ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 22$ IF ERROR
2200 026572          16$:
2201
2202          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
2203
2204          ;WAIT FOR COMMAND TO COMPLETE
      JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
2205
2206          ;GO READ STATUS FOR READ OPERATION
2207 026602 004737 040110          JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026606 000404          BR     17$            ;GO TO 17$ IF NO ERROR
      026610 000240          NOP
      026612 104000          EMT
      026614 000137 026740          JMP    22$            ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 22$ IF ERROR
2208 026620          17$:
2209
2210          ;CHECK FOR ERRORS DURING READ OPERATION
2211 026620 004737 041126          JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      026624 000405          BR     18$            ;GO TO 18$ IF NO ERROR
      026626 000240          NOP
      026630 104000          EMT
      026632 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      026634 000137 026740          JMP    22$            ;GO TO 22$ IF ERROR
2212 026640          18$:
2213 026640 032737 010000 001400      BIT    #IVC,RMER2I    ;WAS "IVC" DETECTED??
2214 026646 001024          BNE    20$            ;YES!!
2215
2216 026650 004737 054166          JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      026654 000405          BR     19$            ;GO TO 19$ IF NO ERROR
      026656 000240          NOP
      026660 104000          EMT
      026662 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      026664 000137 026740          JMP    22$            ;GO TO 22$ IF ERROR
2217 026670          19$:
2218 026670 013737 001400 001140      MOV    RMER2I,$GDDAT  ;EXPECTED STATUS
2219 026676 052737 010000 001140      BIS    #IVC,$GDDAT
2220 026704 013737 001400 001142      MOV    RMER2I,$BDDAT  ;RECEIVED STATUS
2221 026712 104342          EMT
2222 026714 000137 026740          JMP    22$
2223 026720          20$:
2224 026720 004737 041760          JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      026724 000405          BR     21$            ;GO TO 21$ IF NO ERROR
      026726 000240          NOP
      026730 104000          EMT
      026732 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      026734 000137 026740          JMP    22$            ;GO TO 22$ IF ERROR
2225 026740          21$:
2226
2227 026740          22$:
2228

```


2229

```

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

```

026740
026740 000004
026742 000240
026744 012706 001100
026750 013700 001276
026754 013701 001466
026760 012737 000017 001226
  
```

2230

```

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

```

2231
2232
2233
2234
2235 026766 012737 001057 001446
2236 026774 012737 001000 001420
2237 027002 012737 104254 001416
2238 027010 012737 177400 001414
2239 027016 012737 010000 001444
2240 027024 012737 000061 001412
2241 027032
2242
2243
  
```

```

:LOAD PARAMETERS AND GENERATE DATA BUFFER
      MOV #559.,RMDCO :CYLINDER = 559.
      MOV #TA2,RMDAO  :TRACK = 2, SECTOR = 0
      MOV #BUFONE,RMBAO :BUS ADDRESS
      MOV #-256.,RMWCO :256. WORDS (2'S COMP)
      MOV #FMT16,RMOFO :16 BIT FORMAT
      MOV #WD!GO,RMCS10 :WRITE DATA COMMAND
  
```

2242

```

1$:
:PREPARE DEVICE FOR WRITE OPERATION
  
```

```

2244 027032 004737 034032
      027036 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 2$ IF NO ERROR
                  :RETURN HERE IF ERROR
                  :ERROR # DEFINED BY TSTPRP SUBROUTINE
                  :GO TO 21$ IF ERROR
  
```

```

      027040 000404
      027042 000240
      027044 104000
      027046 000137 027656
2245 027052
2246
2247
  
```

```

2$:
:SET UNSAFE ERROR
      MOV #RMER1,PUTINX :SETUP PUT INDEX TABLE
      MOV #200,PUTINX+1 :SET TERMINATOR BYTE
      MOV #UNS,RMER10   :SET RMER1 OUTPUT BUFFER = UNS
      JSR PC,PUT        :GO WRITE RMER1 VIA PUT SUBROUTINE
      BR 3$             :GO TO 3$ IF NO ERROR
      NOP               :RETURN HERE IF ERROR
      EMT               :ERROR DEFINED BY PUT SUBROUTINE
      JMP 21$          :GO TO 21$ IF ERROR
  
```

```

2248 027052 112737 000014 001555
      027060 112737 000200 001556
      027066 012737 040000 001426
      027074 004737 040360
      027100 000404
      027102 000240
      027104 104000
      027106 000137 027656
2249 027112
2250
2251
  
```

```

3$:
:WRITE DATA TO THE DRIVE
4$:
  
```

```

2252 027112
2253 027112 012737 000061 001412
2254 027120 012702 001555
2255 027124 112722 000006
2256 027130 112722 000034
2257 027134 112722 000032
  
```

```

      MOV #WD!GO,RMCS10 :WRITE DATA COMMAND
      MOV #PUTINX,R2     :LOAD PUT REGISTER INDEX TABLE
      MOVB #RMDA,(R2)+
      MOVB #RMDC,(R2)+
      MOVB #RMOF,(R2)+
  
```

```

117 WRITE, READ W/ ABORT
2258 027140 112722 000004      MOVB  #RMB, (R2)+
2259 027144 112722 000002      MOVB  #RMWC, (R2)+
2260 027150 112722 000000      MOVB  #RMCS1, (R2)+
2261 027154 112722 000200      MOVB  #200, (R2)+
2262
2263 027160 004737 040360      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027164 000404      BR    5$          ;GO TO 5$ IF NO ERROR
      027166 000240      NOP                    ;RETURN HERE IF ERROR
      027170 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      027172 000137 027656      JMP   21$         ;GO TO 21$ IF ERROR
2264 027176
2265
2266      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR   PC,GETSTS ;GO TO GETSTS SUBROUTINE
2267
2268      ;GO GET REGISTER STATUS
2269 027202 004737 040110      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027206 000404      BR    6$          ;GO TO 6$ IF NO ERROR
      027210 000240      NOP                    ;RETURN HERE IF ERROR
      027212 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      027214 000137 027656      JMP   21$         ;GO TO 21$ IF ERROR
2270 027220
2271 027220 032737 020000 001350 6$: BIT   #PIP,RMDSI      ;WAS COMMAND EXECUTED?
2272 027226 001412      BEQ   7$          ;NO, GO WAIT
2273 027230 013737 001350 001140      MOV   RMDSI,$GDDAT ;EXPECTED STATUS
2274 027236 042737 020000 001140      BIC   #PIP,$GDDAT
2275 027244 013737 001350 001142      MOV   RMDSI,$BDDAT ;RECEIVED STATUS
2276 027252 104347
2277 027254
2278
2279      ;WAIT FOR COMMAND TO COMPLETE
      JSR   PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
2280
2281      ;GO GET REGISTER STATUS
2282 027260 004737 040110      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027264 000404      BR    8$          ;GO TO 8$ IF NO ERROR
      027266 000240      NOP                    ;RETURN HERE IF ERROR
      027270 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      027272 000137 027656      JMP   21$         ;GO TO 21$ IF ERROR
2283 027276
2284 027276 004737 041126      JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      027302 000405      BR    9$          ;GO TO 9$ IF NO ERROR
      027304 000240      NOP                    ;RETURN HERE IF ERROR
      027306 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      027310 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      027312 000137 027656      JMP   21$         ;GO TO 21$ IF ERROR
2285 027316
2286 027316 004737 041760      JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      027322 000405      BR    10$         ;GO TO 10$ IF NO ERROR
      027324 000240      NOP                    ;RETURN HERE IF ERROR
      027326 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      027330 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      027332 000137 027656      JMP   21$         ;GO TO 21$ IF ERROR
2287 027336
2288
2289
2290 027336 012737 027346 001124 ;CHANGE LOOP ADDRESSES
      MOV   #11$,$LPERR

```



```

2291 027344 000410          BR      12$
2292
2293          ;*****
2294          ;LOOP #2      READ
2295
2296 027346          11$:
2297
2298          ;PREPARE DEVICE FOR READ OPERATION
2299 027346 004737 034032      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
          027352 154130      .WORD    154130      ;TASK DESCRIPTOR AS FOLLOWS:
          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
          ;CLEAR CONTROLLER & SELECT DEVICE
          ;VERIFY CONTROLLER CLEAR OPERATION
          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
          ;VERIFY PACK ACKNOWLEDGE
          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
          ;VERIFY RECALIBRATION
          ;GO TO 12$ IF NO ERROR
          ;RETURN HERE IF ERROR
          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          ;GO TO 21$ IF ERROR

          027354 000404          BR      12$
          027356 000240          NOP
          027360 104000          EMT
          027362 000137 027656      JMP      21$
2300 027366          12$:
2301
2302          ;SET UNSAFE ERROR
2303 027366 112737 000014 001555      MOVB     #RMER1,PUTINX      ;SETUP PUT INDEX TABLE
          027374 112737 000200 001556      MOVB     #200,PUTINX+1      ;SET TERMINATOR BYTE
          027402 012737 040000 001426      MOV      #UNS,RMER10      ;SET RMER1 OUTPUT BUFFER = UNS
          027410 004737 040360          JSR      PC,PUT            ;GO WRITE RMER1 VIA PUT SUBROUTINE
          027414 000404          BR      13$              ;GO TO 13$ IF NO ERROR
          027416 000240          NOP
          027420 104000          EMT
          027422 000137 027656      JMP      21$              ;RETURN HERE IF ERROR
          ;ERROR DEFINED BY PUT SUBROUTINE
          ;GO TO 21$ IF ERROR
2304 027426          13$:
2305
2306          ;READ DATA FROM DEVICE
2307 027426          14$:
2308 027426 012737 000071 001412      MOV      #RD!GO,RMCS10      ;READ DATA COMMAND
          2309 027434 012702 001555          MOV      #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
          2310 027440 112722 000006          MOVB     #RMDA,(R2)+
          2311 027444 112722 000032          MOVB     #RMOF,(R2)+
          2312 027450 112722 000034          MOVB     #RMDC,(R2)+
          2313 027454 112722 000004          MOVB     #RMBA,(R2)+
          2314 027460 112722 000002          MOVB     #RMWC,(R2)+
          2315 027464 112722 000000          MOVB     #RMCS1,(R2)+
          2316 027470 112712 000200          MOVB     #200,(R2)
          2317
          2318 027474 004737 040360          JSR      PC,PUT            ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          027500 000404          BR      15$              ;GO TO 15$ IF NO ERROR
          027502 000240          NOP
          027504 104000          EMT
          027506 000137 027656      JMP      21$              ;RETURN HERE IF ERROR
          ;ERROR # DEFINED BY PUT SUBROUTINE
          ;GO TO 21$ IF ERROR
2319 027512          15$:
2320
2321          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
          027512 004737 040024      JSR      PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2322
2323          ;SEE IF DEVICE STARTED COMMAND

```



```

2324 027516 004737 040110      JSR    PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
      027522 000404      BR     16$         :GO TO 16$ IF NO ERROR
      027524 000240      NOP                    :RETURN HERE IF ERROR
      027526 104000      EMT                    :ERROR # DEFINED BY GET SUBROUTINE
      027530 000137 027656      JMP    21$         :GO TO 21$ IF ERROR
2325 027534      16$:
2326 027534 032737 020000 001350  BIT    #PIP,RMSI    :WAS COMMAND EXECUTED??
2327 027542 001414      BEQ    17$         :NO!!
2328 027544 013737 001350 001140  MOV    RMSI,$GDDAT :EXPECTED STATUS
2329 027552 042737 020000 001140  BIC    #PIP,$GDDAT
2330 027560 013737 001350 001142  MOV    RMSI,$BDDAT :RECEIVED STATUS
2331 027566 104347      EMT    347
2332 027570 000137 027656      JMP    21$
2333 027574      17$:
2334
2335      :WAIT FOR COMMAND TO COMPLETE
      027574 004737 040732      JSR    PC,TIMOUT   :GO TO TIMOUT SUBROUTINE
2336
2337      :GO READ STATUS FOR READ OPERATION
2338 027600 004737 040110      JSR    PC,GET      :GO READ REGISTER(S) WITH GET SUBROUTINE
      027604 000404      BR     18$         :GO TO 18$ IF NO ERROR
      027606 000240      NOP                    :RETURN HERE IF ERROR
      027610 104000      EMT                    :ERROR # DEFINED BY GET SUBROUTINE
      027612 000137 027656      JMP    21$         :GO TO 21$ IF ERROR
2339 027616      18$:
2340 027616 004737 041126      JSR    PC,PRIERR   :GO CHECK FOR PRIMARY ERRORS
      027622 000405      BR     19$         :GO TO 19$ IF NO ERROR
      027624 000240      NOP                    :RETURN HERE IF ERROR
      027626 104000      EMT                    :ERROR # DEFINED BY PRIERR SUBROUTINE
      027630 004736      JSR    PC,@(SP)+   :GO BACK FOR MORE ERROR CHECKS
      027632 000137 027656      JMP    21$         :GO TO 21$ IF ERROR
2341 027636      19$:
2342 027636 004737 041760      JSR    PC,SECERR   :GO CHECK FOR SECONDARY ERRORS
      027642 000405      BR     20$         :GO TO 20$ IF NO ERROR
      027644 000240      NOP                    :RETURN HERE IF ERROR
      027646 104000      EMT                    :ERROR # DEFINED BY SECERR SUBROUTINE
      027650 004736      JSR    PC,@(SP)+   :GO BACK FOR MORE ERROR CHECKS
      027652 000137 027656      JMP    21$         :GO TO 21$ IF ERROR
2343 027656      20$:
2344
2345 027656      21$:
2346
2347      :*****
      :*TEST 20      WRITE, READ EARLY PEAK SHIFT
      :*****
      TST20:
      027656      000004      SCOPE                :SCOPE CALL
      027660 000240      NOP                    :START OF TEST
      027662 012706 001100  MOV    #STACK,SP     :INITIALIZE STACK POINTER
      027666 013700 001276  MOV    $BASE,R0      :R0 = UNIBUS ADDRESS
      027672 013701 001466  MOV    TSTQUE,R1     :(R1) = DEVICE BEING TESTED
      027676 012737 000020 001226  MOV    #20,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
2348
2349
2350      :*****
2351      :LOOP #1      FORMAT,WRITE,READ
2352 027704      1$:
  
```

```

2353
2354
2355 027704 004737 034032      ;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 2$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 25$ IF ERROR
      BR     2$
      NOP
      EMT
      JMP    25$

      027712 000404      BR     2$
      027714 000240      NOP
      027716 104000      EMT
      027720 000137 030664  JMP    25$
2356 027724
2357
2358      ;LOAD PARAMETERS AND GENERATE DATA BUFFER
2359 027724 012737 001057 001446      MOV    #559.,RMDCO      ;CYLINDER = 559.
2360 027732 012737 001000 001420      MOV    #TA2,RMDAO      ;TRACK = 2, SECTOR = 0
2361 027740 012737 104254 001416      MOV    #BUFONE,RMBAO   ;BUS ADDRESS
2362 027746 012737 177376 001414      MOV    #-258.,RMWCO    ;2 + 256. WORDS (2'S COMP)
2363 027754 012737 010000 001444      MOV    #FMT16,RMOFO    ;16 BIT FORMAT
2364 027762 012737 000063 001412      MOV    #WH!GO,RMCS10   ;WRITE HEADER AND DATA COMMAND
2365
2366      ;VERIFY THAT SECTOR IS NOT BAD
      027770 004737 034762      JSR    PC,BADSCT      ;CALL BAD SECTOR MODULE
      027774 000405      BR     3$             ;GO TO 3$ IF NO ERROR
      027776 104401 066006      TYPE   .SCTMSG        ;TYPE BAD SECTOR MESSAGE
      030002 104000      EMT
      030004 000137 030664  JMP    25$           ;ERROR # DEFINED BY BADSCT SUBROUTINE
                                ;GO TO 25$ IF ERROR
2367 030010
2368 030010 012737 067670 001174      3$:   MOV    #EARLY,$TMP0   ;STARTING ADDRESS OF PATTERN
2369 030016 012737 000001 001176      MOV    #1,$TMP1       ;RANGE OF PATTERN
2370 030024 004737 037214      JSR    PC,GENBUF      ;GO GENERATE BUFFER FOR FORMAT
2371
2372      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
2373 030030 012702 001555      MOV    #PUTINX,R2     ;R2 = BYTE ENTRY POSITION
2374 030034 112722 000034      MOVB  #RMDC,(R2)+
2375 030040 112722 000006      MOVB  #RMDA,(R2)+
2376 030044 112722 000004      MOVB  #RMBA,(R2)+
2377 030050 112722 000002      MOVB  #RMWC,(R2)+
2378 030054 112722 000032      MOVB  #RMOF,(R2)+
2379 030060 112722 000000      MOVB  #RMCS1,(R2)+
2380 030064 112712 000200      MOVB  #200,(R2)      ;TERMINATE TABLE
2381 030070
2382 030070 004737 040360      4$:   JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      030074 000404      BR     5$             ;GO TO 5$ IF NO ERROR
      030076 000240      NOP
      030100 104000      EMT
      030102 000137 030664  JMP    25$           ;ERROR # DEFINED BY PUT SUBROUTINE
                                ;GO TO 25$ IF ERROR
2383 030106
2384
2385      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      030106 004737 040024      JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2386
2387      ;WAIT FOR COMMAND TO COMPLETE
  
```



```

2423
2424 030266 004737 040360      JSR    PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      030272 000404      BR     11$        ;GO TO 11$ IF NO ERROR
      030274 000240      NOP                    ;RETURN HERE IF ERROR
      030276 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      030300 000137 030664      JMP    25$        ;GO TO 25$ IF ERROR
2425 030304      11$:
2426
2427      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      030304 004737 040024      JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
2428
2429      ;WAIT FOR COMMAND TO COMPLETE
      030310 004737 040732      JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
2430
2431      ;GO READ STATUS FOR WRITE COMMAND
2432 030314 004737 040110      JSR    PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      030320 000404      BR     12$        ;GO TO 12$ IF NO ERROR
      030322 000240      NOP                    ;RETURN HERE IF ERROR
      030324 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      030326 000137 030664      JMP    25$        ;GO TO 25$ IF ERROR
2433 030332      12$:
2434
2435      ;CHECK FOR ERRORS DURING WRITE OPERATION
2436 030332 004737 041126      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      030336 000405      BR     13$        ;GO TO 13$ IF NO ERROR
      030340 000240      NOP                    ;RETURN HERE IF ERROR
      030342 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      030344 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      030346 000137 030664      JMP    25$        ;GO TO 25$ IF ERROR
2437 030352      13$:
2438 030352 004737 054166      JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      030356 000405      BR     14$        ;GO TO 14$ IF NO ERROR
      030360 000240      NOP                    ;RETURN HERE IF ERROR
      030362 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      030364 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      030366 000137 030664      JMP    25$        ;GO TO 25$ IF ERROR
2439 030372      14$:
2440 030372 004737 041760      JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      030376 000405      BR     15$        ;GO TO 15$ IF NO ERROR
      030400 000240      NOP                    ;RETURN HERE IF ERROR
      030402 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      030404 004736      JSR    PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      030406 000137 030664      JMP    25$        ;GO TO 25$ IF ERROR
2441 030412      15$:
2442
2443      ;CHANGE LOOP ADDRESSES
2444 030412 012737 030422 001124  MOV    #16$,$LPERR
2445 030420 000410      BR     17$
2446
2447      ;*****
2448      ;LOOP #3      READ
2449
2450 030422      16$:
2451
2452      ;PREPARE DEVICE FOR READ OPERATION
2453 030422 004737 034032      JSR    PC,TSTPRP   ;PREPARE DEVICE FOR TEST
      030426 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 17$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 25$ IF ERROR
030430 000404 BR 17$
030432 000240 NOP
030434 104000 EMT
030436 000137 030664 JMP 25$
2454 030442 17$:
2455
2456 :READ DATA FROM DEVICE
2457 030442 18$:
2458 030442 012737 000071 001412 MOV #RD!GO, RMCS10 :READ DATA COMMAND
2459 030450 012737 105264 001416 MOV #BUFTWO+4, RMBAD :CHANGE BUFFER
2460 030456 012702 001555 MOV #PUTINX, R2 :LOAD PUT REGISTER INDEX TABLE
2461 030462 112722 000006 MOVB #RMDA, (R2)+
2462 030466 112722 000032 MOVB #RMOF, (R2)+
2463 030472 112722 000034 MOVB #RMDC, (R2)+
2464 030476 112722 000004 MOVB #RMDA, (R2)+
2465 030502 112722 000002 MOVB #RMWC, (R2)+
2466 030506 112722 000000 MOVB #RMCS1, (R2)+
2467 030512 112712 000200 MOVB #200, (R2)
2468
2469 030516 004737 040360 JSR PC, PUT :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030522 000404 BR 19$ :GO TO 19$ IF NO ERROR
030524 000240 NOP :RETURN HERE IF ERROR
030526 104000 EMT :ERROR # DEFINED BY PUT SUBROUTINE
030530 000137 030664 JMP 25$ :GO TO 25$ IF ERROR
2470 030534 19$:
2471
2472 :SETUP GET INDEX TABLE TO READ ALL REGISTERS
030534 004737 040024 JSR PC, GETSTS :GO TO GETSTS SUBROUTINE
2473
2474 :WAIT FOR COMMAND TO COMPLETE
030540 004737 040732 JSR PC, TIMEOUT :GO TO TIMEOUT SUBROUTINE
2475
2476 :GO READ STATUS FOR READ OPERATION
2477 030544 004737 040110 JSR PC, GET :GO READ REGISTER(S) WITH GET SUBROUTINE
030550 000404 BR 20$ :GO TO 20$ IF NO ERROR
030552 000240 NOP :RETURN HERE IF ERROR
030554 104000 EMT :ERROR # DEFINED BY GET SUBROUTINE
030556 000137 030664 JMP 25$ :GO TO 25$ IF ERROR
2478 030562 20$:
2479
2480 :CHECK FOR ERRORS DURING READ OPERATION
2481 030562 004737 041126 JSR PC, PRIERR :GO CHECK FOR PRIMARY ERRORS
030566 000405 BR 21$ :GO TO 21$ IF NO ERROR
030570 000240 NOP :RETURN HERE IF ERROR
030572 104000 EMT :ERROR # DEFINED BY PRIERR SUBROUTINE
030574 004736 JSR PC, @ (SP)+ :GO BACK FOR MORE ERROR CHECKS
030576 000137 030664 JMP 25$ :GO TO 25$ IF ERROR
2482 030602 21$:
2483 030602 004737 054166 JSR PC, DTASTS :GO VERIFY RESULTS OF DATA TRANSFER
030606 000405 BR 22$ :GO TO 22$ IF NO ERROR
    
```



```

030610 000240      NOP      ;RETURN HERE IF ERROR
030612 104000      EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
030614 004736      JSR      PC,a(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030616 000137 030664 JMP      25$      ;GO TO 25$ IF ERROR
2484 030622      22$:      JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
2485 030622 004737 041760 BR      23$      ;GO TO 23$ IF NO ERROR
030626 000405      NOP      ;RETURN HERE IF ERROR
030630 000240      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
030632 104000      JSR      PC,a(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030634 004736      JMP      25$      ;GO TO 25$ IF ERROR
2486 030636 000137 030664      23$:      JSR      PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
2487 030642 004737 037462 .WORD  BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
030646 104260      .WORD  BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
030650 105264      BR      24$      ;GO TO 24$ IF NO ERROR
030652 000404      NOP      ;RETURN HERE IF ERROR
030654 000240      EMT      ;ERROR # DEFINED BY CMPBUF SUBROUTINE
030656 104000      JMP      25$      ;GO TO 25$ IF ERROR
2488 030664      24$:
2489
2490 030664      25$:
2491
2492
::*****
:*TEST 21      WRITE, READ EACH TRACK W/ MIXED PEAK SHIFT
::*****
TST21:
030664 000004      SCOPE    ;SCOPE CALL
030666 000240      NOP      ;START OF TEST
030670 012706 001100 MOV      #STACK,SP ;INITIALIZE STACK POINTER
030674 013700 001276 MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
030700 013701 001466 MOV      TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
030704 012737 000021 001226 MOV      #21,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
2493
2494
2495
2496
2497 030712      1$:
2498
2499
2500 030712 004737 034032 ;PREPARE THE DEVICE FOR FORMAT OPERATION
030716 154130      JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
                                .WORD  154130 ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 2$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 26$ IF ERROR
030720 000404      BR      2$
030722 000240      NOP
030724 104000      EMT
030726 000137 031712 JMP      26$
2501 030732      2$:
2502
2503
2504 030732 012737 001060 001446 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
                                MOV      #560.,RMDCO ;CYLINDER = 560.

```


T21

WRITE, READ EACH TRACK W/ MIXED PEAK SHIFT

```

2505 030740 012737 000000 001420      MOV      #0,RMDAO      ;TRACK = 0, SECTOR = 0
2506 030746      3$:      MOV      #BUFONE,RMBAO      ;BUS ADDRESS
2507 030746 012737 104254 001416      MOV      #-258.,RMWCO      ;2 + 256. WORDS (2'S COMP)
2508 030754 012737 177376 001414      MOV      #FMT16,RMOFO      ;16 BIT FORMAT
2509 030762 012737 010000 001444      MOV      #WH!GO,RMCS10      ;WRITE HEADER AND DATA COMMAND
2510 030770 012737 000063 001412
2511
2512      ;VERIFY THAT SECTOR IS NOT BAD
      JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
      BR      4$      ;GO TO 4$ IF NO ERROR
      TYPE      ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
      EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
      JMP      26$      ;GO TO 26$ IF ERROR
2513 031016      4$:      MOV      #MIXED,$TMP0      ;STARTING ADDRESS OF PATTERN
2514 031016 012737 067354 001174      MOV      #128.,$TMP1      ;RANGE OF PATTERN
2515 031024 012737 000200 001176      JSR      PC,GENBUF      ;GO GENERATE BUFFER FOR FORMAT
2516 031032 004737 037214
2517
2518      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
      MOV      #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
2519 031036 012702 001555      MOV      #RMDC,(R2)+
2520 031042 112722 000034      MOV      #RMDA,(R2)+
2521 031046 112722 000006      MOV      #RMBA,(R2)+
2522 031052 112722 000004      MOV      #RMWC,(R2)+
2523 031056 112722 000002      MOV      #RMOF,(R2)+
2524 031062 112722 000032      MOV      #RMCS1,(R2)+
2525 031066 112722 000000      MOV      #200,(R2)      ;TERMINATE TABLE
2526 031072 112712 000200
2527 031076      5$:
2528
2529      ;FORMAT THE DRIVE
      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR      6$      ;GO TO 6$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP      26$      ;GO TO 26$ IF ERROR
2530 031076 004737 040360
2531 031102 000404
2532 031104 000240
2533 031106 104000
2534 031110 000137 031712
2535 031114      6$:
2536
2537      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2538 031114 004737 040024
2539 031120 004737 040732
2540      ;WAIT FOR COMMAND TO COMPLETE
      JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
2541
2542      ;GO READ STATUS FOR FORMAT OPERATION
      JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR      7$      ;GO TO 7$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY GET SUBROUTINE
      JMP      26$      ;GO TO 26$ IF ERROR
2543 031124 004737 040110
2544 031130 000404
2545 031132 000240
2546 031134 104000
2547 031136 000137 031712
2548 031142      7$:
2549
2550      ;VERIFY NO ERRORS DURING FORMAT
      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      BR      8$      ;GO TO 8$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
2551 031142 004737 054166
2552 031146 000405
2553 031150 000240
2554 031152 104000
2555 031154 004736

```

```

2543 031156 000137 031712      JMP      26$      ;GO TO 26$ IF ERROR
2544 031162
2545      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
2546 031162 012737 031172 001124  MOV      #9$, $LPER
2547 031170 000410      BR       10$      ;SKIP TO WRITE OPERATION
2548
2549      ;*****
2550      ;LOOP #2      WRITE,READ
2551
2552 031172      9$:
2553
2554      ;PREPARE DEVICE FOR WRITE OPERATION
2555 031172 004737 034032  JSR      PC, TSTPRP ;PREPARE DEVICE FOR TEST
031176 154130      .WORD   154130    ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 10$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 26$ IF ERROR
                                ;
031200 000404      BR       10$
031202 000240      NOP
031204 104000      EMT
031206 000137 031712  JMP      26$
2556 031212      10$:
2557
2558      ;WRITE DATA TO THE DRIVE
2559 031212      11$:
2560 031212 012737 104260 001416  MOV      #BUFONE+4, RMBAO ;CHANGE BUS ADDRESS
2561 031220 012737 177400 001414  MOV      #-256., RMC0    ;CHANGE WORD COUNT
2562 031226 012737 000061 001412  MOV      #WD!GO, RMC$10 ;WRITE DATA COMMAND
2563 031234 012702 001555      MOV      #PUTINX, R2    ;LOAD PUT REGISTER INDEX TABLE
2564 031240 112722 000006      MOVB     #RMDA, (R2)+
2565 031244 112722 000034      MOVB     #RMDC, (R2)+
2566 031250 112722 000032      MOVB     #RMOF, (R2)+
2567 031254 112722 000004      MOVB     #RMB A, (R2)+
2568 031260 112722 000002      MOVB     #RMWC, (R2)+
2569 031264 112722 000000      MOVB     #RMCS1, (R2)+
2570 031270 112722 000200      MOVB     #200, (R2)+ ;TERMINATE TABLE
2571
2572 031274 004737 040360      JSR      PC, PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
031300 000404      BR       12$      ;GO TO 12$ IF NO ERROR
031302 000240      NOP
031304 104000      EMT
031306 000137 031712  JMP      26$      ;ERROR # DEFINED BY PUT SUBROUTINE
                                ;GO TO 26$ IF ERROR
2573 031312      12$:
2574
2575      ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
031312 004737 040024      JSR      PC, GETSTS ;GO TO GETSTS SUBROUTINE
2576
2577      ;WAIT FOR COMMAND TO COMPLETE
031316 004737 040732      JSR      PC, TIMOUT ;GO TO TIMOUT SUBROUTINE
2578
2579      ;GO READ STATUS FOR WRITE COMMAND
2580 031322 004737 040110      JSR      PC, GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
    
```



```

031326 000404 BR 13$ ;GO TO 13$ IF NO ERROR
031330 000240 NOP ;RETURN HERE IF ERROR
031332 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
031334 000137 031712 JMP 26$ ;GO TO 26$ IF ERROR
2581 031340 13$:
2582
2583 ;CHECK FOR ERRORS DURING WRITE OPERATION
2584 031340 004737 041126 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
031344 000405 BR 14$ ;GO TO 14$ IF NO ERROR
031346 000240 NOP ;RETURN HERE IF ERROR
031350 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
031352 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031354 000137 031712 JMP 26$ ;GO TO 26$ IF ERROR
2585 031360 14$:
2586 031360 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
031364 000405 BR 15$ ;GO TO 15$ IF NO ERROR
031366 000240 NOP ;RETURN HERE IF ERROR
031370 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
031372 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031374 000137 031712 JMP 26$ ;GO TO 26$ IF ERROR
2587 031400 15$:
2588 031400 004737 041760 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
031404 000405 BR 16$ ;GO TO 16$ IF NO ERROR
031406 000240 NOP ;RETURN HERE IF ERROR
031410 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
031412 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
031414 000137 031712 JMP 26$ ;GO TO 26$ IF ERROR
2589 031420 16$:
2590
2591 ;CHANGE LOOP ADDRESSES
2592 031420 012737 031430 001124 MOV #17$,SLPERR
2593 031426 000410 BR 18$ ;SKIP TO NEXT OPERATION
2594
2595 ;*****
2596 ;LOOP #3 READ
2597
2598 031430 17$:
2599
2600 ;PREPARE DEVICE FOR READ OPERATION
2601 031430 004737 034032 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
031434 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 18$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 26$ IF ERROR
031436 000404 BR 18$
031440 000240 NOP
031442 104000 EMT
031444 000137 031712 JMP 26$
2602 031450 18$:
2603
2604 ;READ DATA FROM DEVICE
2605 031450 19$:
2606 031450 012737 105264 001416 MOV #BUFTWO+4,RMBAO ;CHANGE BUS ADDRESS
    
```


2607	031456	012737	000071	001412	MOV	#RD!GO, RMCS10	:READ DATA COMMAND
2608	031464	012702	001555		MOV	#PUTINX, R2	:LOAD PUT REGISTER INDEX TABLE
2609	031470	112722	000006		MOVB	#RMDA, (R2)+	
2610	031474	112722	000032		MOVB	#RMOF, (R2)+	
2611	031500	112722	000034		MOVB	#RMDC, (R2)+	
2612	031504	112722	000004		MOVB	#RMBA, (R2)+	
2613	031510	112722	000002		MOVB	#RMWC, (R2)+	
2614	031514	112722	000000		MOVB	#RMCS1, (R2)+	
2615	031520	112712	000200		MOVB	#200, (R2)	
2616							
2617	031524	004737	040360		JSR	PC, PUT	:GO WRITE REGISTER(S) WITH PUT SUBROUTINE
	031530	000404			BR	20\$:GO TO 20\$ IF NO ERROR
	031532	000240			NOP		:RETURN HERE IF ERROR
	031534	104000			EMT		:ERROR # DEFINED BY PUT SUBROUTINE
	031536	000137	031712		JMP	26\$:GO TO 26\$ IF ERROR
2618	031542			20\$:			
2619							
2620							:SETUP GET INDEX TABLE TO READ ALL REGISTERS
	031542	004737	040024		JSR	PC, GETSTS	:GO TO GETSTS SUBROUTINE
2621							
2622							:WAIT FOR COMMAND TO COMPLETE
	031546	004737	040732		JSR	PC, TIMEOUT	:GO TO TIMEOUT SUBROUTINE
2623							
2624							:GO READ STATUS FOR READ OPERATION
2625	031552	004737	040110		JSR	PC, GET	:GO READ REGISTER(S) WITH GET SUBROUTINE
	031556	000404			BR	21\$:GO TO 21\$ IF NO ERROR
	031560	000240			NOP		:RETURN HERE IF ERROR
	031562	104000			EMT		:ERROR # DEFINED BY GET SUBROUTINE
	031564	000137	031712		JMP	26\$:GO TO 26\$ IF ERROR
2626	031570			21\$:			
2627							
2628							:CHECK FOR ERRORS DURING READ OPERATION
2629	031570	004737	041126		JSR	PC, PRIERR	:GO CHECK FOR PRIMARY ERRORS
	031574	000405			BR	22\$:GO TO 22\$ IF NO ERROR
	031576	000240			NOP		:RETURN HERE IF ERROR
	031600	104000			EMT		:ERROR # DEFINED BY PRIERR SUBROUTINE
	031602	004736			JSR	PC, @ (SP)+	:GO BACK FOR MORE ERROR CHECKS
	031604	000137	031712		JMP	26\$:GO TO 26\$ IF ERROR
2630	031610			22\$:			
2631	031610	004737	054166		JSR	PC, DTASTS	:GO VERIFY RESULTS OF DATA TRANSFER
	031614	000405			BR	23\$:GO TO 23\$ IF NO ERROR
	031616	000240			NOP		:RETURN HERE IF ERROR
	031620	104000			EMT		:ERROR # DEFINED BY DTASTS SUBROUTINE
	031622	004736			JSR	PC, @ (SP)+	:GO BACK FOR MORE ERROR CHECKS
	031624	000137	031712		JMP	26\$:GO TO 26\$ IF ERROR
2632	031630			23\$:			
2633	031630	004737	041760		JSR	PC, SECERR	:GO CHECK FOR SECONDARY ERRORS
	031634	000405			BR	24\$:GO TO 24\$ IF NO ERROR
	031636	000240			NOP		:RETURN HERE IF ERROR
	031640	104000			EMT		:ERROR # DEFINED BY SECERR SUBROUTINE
	031642	004736			JSR	PC, @ (SP)+	:GO BACK FOR MORE ERROR CHECKS
	031644	000137	031712		JMP	26\$:GO TO 26\$ IF ERROR
2634	031650			24\$:			
2635	031650	004737	037462		JSR	PC, CMPBUF	:GO COMPARE WRITE, READ DATA BUFFERS
	031654	104260			.WORD	BUFONE+4	:STARTING ADDRESS OF WRITE BUFFER
	031656	105264			.WORD	BUFTWO+4	:STARTING ADDRESS OF READ BUFFER
	031660	000404			BR	25\$:GO TO 25\$ IF NO ERROR

T21 WRITE, READ EACH TRACK W/ MIXED PEAK SHIFT

```

031662 000240      NOP      :RETURN HERE IF ERROR
031664 104000      EMT      :ERROR # DEFINED BY CMPBUF SUBROUTINE
031666 000137 031712  JMP      26$      :GO TO 26$ IF ERROR
2636 031672      25$:
2637 031672 105237 001421      INCB     RMDAO+1      :ADVANCE TO NEXT TRACK
2638 031676 123737 001421 001335  CMPB     RMDAO+1,LSTRK+1 :DONE ?
2639 031704 101002      BHI     26$      :YES!!
2640 031706 000137 030746  JMP      3$      :TEST NEXT TRACK
2641 031712      26$:
2642
2643
:*****
:*TEST 22      READ, WRITE CHECK MULTIPLE SECTORS IN OFFSET MODE
:*****
TST22:
031712      SCOPE     :SCOPE CALL
031712 000004      NOP     :START OF TEST
031714 000240      MOV     #STACK,SP :INITIALIZE STACK POINTER
031716 012706 001100  MOV     $BASE,R0  :R0 = UNIBUS ADDRESS
031722 013700 001276  MOV     TSTQUE,R1 : (R1) = DEVICE BEING TESTED
031726 013701 001466  MOV     #22,$TESTN :;SET TEST NUMBER IN APT MAIL BOX
031732 012737 000022 001226

:*****
:LOOP #1      FORMAT,READ OFFSET,WRITE CHECK OFFSET IN BOTH DIRECTIONS
1$:
:PREPARE DEVICE FOR FORMAT OPERATION
031740      JSR     PC,TSTPRP :PREPARE DEVICE FOR TEST
031744 154130      .WORD 154130 :TASK DESCRIPTOR AS FOLLOWS:
:SELECT DEVICE & VERIFY DEVICE AVAILABLE
:CLEAR CONTROLLER & SELECT DEVICE
:VERIFY CONTROLLER CLEAR OPERATION
:PACK ACKNOWLEDGE IF VOLUME NOT VALID
:VERIFY PACK ACKNOWLEDGE
:RECALIBRATE IF 'SKI' OR 'PIP' IS SET
:VERIFY RECALIBRATION
031746 000404      BR      2$      :GO TO 2$ IF NO ERROR
031750 000240      NOP     :RETURN HERE IF ERROR
031752 104000      EMT     :ERROR # DEFINED BY TSTPRP SUBROUTINE
031754 000137 033074  JMP     28$     :GO TO 28$ IF ERROR
2651
2652
:LOAD PARAMETERS AND GENERATE DATA BUFFER
2653 031760      2$:
2654 031760 012737 001057 001446  MOV     #559,RMDCO :CYLINDER = 559.
2655 031766 012737 001000 001420  MOV     #TA2,RMDAO :TRACK = 2, SECTOR = 0
2656 031774 012737 104254 001416  MOV     #BUFONE,RMBAO :BUFFER ADDRESS
2657 032002 012737 176774 001414  MOV     #-258,*2,RMWCO :WORD COUNT FOR 2 SECTORS (2'S COMP)
2658 032010 012737 010000 001444  MOV     #FMT16,RMOFO :16 BIT MODE
2659 032016 012737 000063 001412  MOV     #WH!GO,RMCS10 :WRITE HEADER
2660
2661
:VERIFY THAT SECTOR IS NOT BAD
032024 004737 034762      JSR     PC,BADSCT :CALL BAD SECTOR MODULE
032030 000405      BR      3$      :GO TO 3$ IF NO ERROR
032032 104401 066006      TYPE     ,SCTMSG :TYPE BAD SECTOR MESSAGE
032036 104000      EMT     :ERROR # DEFINED BY BADSCT SUBROUTINE
032040 000137 033074      JMP     28$     :GO TO 28$ IF ERROR
2662 032044      3$:
2663 032044 012737 067456 001174  MOV     #ZEROS,$TMP0 :DATA PATTERN

```



```

2664 032052 012737 000001 001176      MOV    #1,$TMP1
2665 032060 004737 037214                JSR    PC,GENBUF          ;TO GENERATE BUFFER FOR FORMAT
2666
2667                                ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
2668 032064 012702 001555      MOV    #PUTINX,R2        ;BYTE ENTRY TABLE
2669 032070 112722 000034      MOVB  #RMDC,(R2)+
2670 032074 112722 000006      MOVB  #RMDA,(R2)+
2671 032100 112722 000004      MOVB  #RMBA,(R2)+
2672 032104 112722 000002      MOVB  #RMWC,(R2)+
2673 032110 112722 000032      MOVB  #RMOF,(R2)+
2674 032114 112722 000000      MOVB  #RMCS1,(R2)+
2675 032120 112712 000200      MOVB  #200,(R2)         ;TABLE TERMINATOR
2676 032124
2677 032124 004737 040360      4$:   JSR    PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR    5$             ;GO TO 5$ IF NO ERROR
      NOP                    ;RETURN HERE IF ERROR
      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP   28$            ;GO TO 28$ IF ERROR
2678 032142
2679
2680                                ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
2681 032142 004737 040024      JSR    PC,GETSTS        ;GO TO GETSTS SUBROUTINE
2682
2683                                ;WAIT FOR COMMAND TO COMPLETE
2684 032146 004737 040732      JSR    PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
2685
2686                                ;GO READ STATUS FOR FORMAT OPERATION
2687 032152 004737 040110      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
2688 032156 000404      BR    6$             ;GO TO 6$ IF NO ERROR
2689 032160 000240      NOP                    ;RETURN HERE IF ERROR
2690 032162 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
2691 032164 000137 033074      JMP   28$            ;GO TO 28$ IF ERROR
2692
2693                                ;VERIFY NO ERROR DURING FORMAT
2694 032170
2695 032170 004737 054166      JSR    PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
2696 032174 000405      BR    7$             ;GO TO 7$ IF NO ERROR
2697 032176 000240      NOP                    ;RETURN HERE IF ERROR
2698 032200 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
2699 032202 004736      JSR    PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
2700 032204 000137 033074      JMP   28$            ;GO TO 28$ IF ERROR
2701
2702                                ;MOV LOOP ADDRESSES TO NEXT OPERATION
2703 032210
2704 032210 012737 032232 001124      7$:   MOV    #8$,$LPERR      ;ERROR LOOP ADDRESS
2705
2706                                ;LOCATE BUFFER ADDRESS AND WORD COUNT
2707 032216 012737 177000 001414      MOV    #-256.*2,RMWCO ;TWO SECTORS
2708 032224 012737 104254 001416      MOV    #BUFONE,RMBAO   ;BUFFER ADDRESS
2709
2710                                ;*****
2711                                ;LOOP 2 READ AND WRITE CHECK IN OFFSET MODE
2712 032232
2713 032232 004737 034032      8$:   ;PREPARE DEVICE FOR READ OFFSET OPEATION
      JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      .WORD 154130        ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
  
```



```

          032240 000404          BR      9$
          032242 000240          NOP
          032244 104000          EMT
          032246 000137 033074  JMP      28$
2703 032252          9$:
2704 032252          10$:
2705          :READ OFFSET
2706 032252 012702 001555      MOV      #PUTINX,R2      ;LOAD THE TABLE
2707 032256 112722 000006      MOVB     #RMDA,(R2)+
2708 032262 112722 000034      MOVB     #RMDC,(R2)+
2709 032266 112722 000032      MOVB     #RMOF,(R2)+
2710 032272 112722 000004      MOVB     #RMBA,(R2)+
2711 032276 112722 000002      MOVB     #RMWC,(R2)+
2712 032302 112722 000200      MOVB     #200,(R2)+      ;TABLE TERMINATOR
2713
2714 032306 004737 040360      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          032312 000404          BR      11$
          032314 000240          NOP
          032316 104000          EMT
          032320 000137 033074  JMP      28$
2715 032324 012737 000015 001412 11$: MOV      #OFFSET!GO,RMCS10 ;OFFSET COMMAND
2716 032332 012702 001555      MOV      #PUTINX,R2
2717 032336 112722 000000      MOVB     #RMCS1,(R2)+
2718 032342 112722 000200      MOVB     #200,(R2)+
2719
2720 032346 004737 040360      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          032352 000404          BR      12$
          032354 000240          NOP
          032356 104000          EMT
          032360 000137 033074  JMP      28$
2721 032364          12$:
2722
2723          :WAIT FOR COMMAND TO COMPLETE
          032364 004737 040732  JSR      PC,TIMOUT      ;GO TO TIMEOUT SUBROUTINE
2724
2725 032370 012737 000071 001412  MOV      #RD!GO,RMCS10      ;READ DATA COMMAND
2726 032376 012702 001555      MOV      #PUTINX,R2      ;TABLE INDEX
2727 032402 112722 000000      MOVB     #RMCS1,(R2)+
2728 032406 112712 000200      MOVB     #200,(R2)+      ;TABLE TERMINATOR
2729
2730 032412 004737 040360      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          032416 000404          BR      13$
          032420 000240          NOP
          032422 104000          EMT
          032424 000137 033074  JMP      28$
2731 032430          13$:
2732
2733          :SETUP GET INDEX TABLE TO READ ALL REGISTERS
          032430 004737 040024  JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2734
2735          :WAIT FOR COMMAND TO COMPLETE
          032434 004737 040732  JSR      PC,TIMOUT      ;GO TO TIMEOUT SUBROUTINE
  
```

```

;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
;VERIFY RECALIBRATION
;GO TO 9$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 28$ IF ERROR
  
```

```

;LOAD THE TABLE
  
```

```

;TABLE TERMINATOR
  
```

```

;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
;GO TO 11$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY PUT SUBROUTINE
;GO TO 28$ IF ERROR
;OFFSET COMMAND
  
```

```

;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
;GO TO 12$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY PUT SUBROUTINE
;GO TO 28$ IF ERROR
  
```

```

;GO TO TIMEOUT SUBROUTINE
  
```

```

;READ DATA COMMAND
;TABLE INDEX
;TABLE TERMINATOR
  
```

```

;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
;GO TO 13$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY PUT SUBROUTINE
;GO TO 28$ IF ERROR
  
```

```

;GO TO GETSTS SUBROUTINE
  
```

```

;GO TO TIMEOUT SUBROUTINE
  
```

```

2736
2737
2738 032440 004737 040110      ;READ STATUS FOR READ OFFSET OPERATION
      032444 000404          JSR   PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      032446 000240          BR    14$             ;GO TO 14$ IF NO ERROR
      032450 104000          NOP                    ;RETURN HERE IF ERROR
      032452 000137 033074    EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
                                      JMP   28$             ;GO TO 28$ IF ERROR

2739
2740
2741 032456
2742 032456 004737 041126      ;CHECK ERROR DURING READ OFFSET
      032462 000405          14$: JSR   PC,PRIERR       ;GO CHECK FOR PRIMARY ERRORS
      032464 000240          BR    15$             ;GO TO 15$ IF NO ERROR
      032466 104000          NOP                    ;RETURN HERE IF ERROR
      032470 004736          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      032472 000137 033074    JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
                                      JMP   28$             ;GO TO 28$ IF ERROR

2743 032476
2744 032476 004737 054166      15$: JSR   PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      032502 000405          BR    16$             ;GO TO 16$ IF NO ERROR
      032504 000240          NOP                    ;RETURN HERE IF ERROR
      032506 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      032510 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      032512 000137 033074    JMP   28$             ;GO TO 28$ IF ERROR

2745 032516
2746 032516 004737 041760      16$: JSR   PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      032522 000405          BR    17$             ;GO TO 17$ IF NO ERROR
      032524 000240          NOP                    ;RETURN HERE IF ERROR
      032526 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      032530 004736          JSR   PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
      032532 000137 033074    JMP   28$             ;GO TO 28$ IF ERROR

2747 032536
2748
2749
2750 032536 012737 032546 001124 ;CHANGE LOOP ADDRESS
      032544 000410          MOV   #18$, $LPERR
      2751 032544 000410          BR    19$             ;TO NEXT OPERATION
      2752
      2753
      2754
      2755 032546
      2756
      2757
      2758 032546 004737 034032      ;PREPARE DEVICE FOR READ OPERATION
      032552 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
      032554 000404          BR    19$             ;GO TO 19$ IF NO ERROR
      032556 000240          NOP                    ;RETURN HERE IF ERROR
      032560 104000          EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      032562 000137 033074    JMP   28$             ;GO TO 28$ IF ERROR

2759 032566
2760
2761
      19$:
      ;WRITE CHECK DATA IN OFFSET MODE
  
```



```

2762 032566          20$:
2763 032566 012702 001555      MOV    #PUTINX,R2
2764 032572 112722 000006      MOVB  #RMDA,(R2)+
2765 032576 112722 000032      MOVB  #RMOF,(R2)+
2766 032602 112722 000034      MOVB  #RMDC,(R2)+
2767 032606 112722 000004      MOVB  #RMBB,(R2)+
2768 032612 112722 000002      MOVB  #RMWC,(R2)+
2769 032616 112712 000200      MOVB  #200,(R2)      ;TABLE TERMINATOR
2770
2771 032622 004737 040360      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      032626 000404      BR    21$          ;GO TO 21$ IF NO ERROR
      032630 000240      NOP                    ;RETURN HERE IF ERROR
      032632 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      032634 000137 033074      JMP   28$          ;GO TO 28$ IF ERROR

2772
2773          ;SETUP OFFSET MODE FIRST
2774 032640          21$:
2775 032640 012737 000015 001412      MOV   #OFFSET!GO,RMCS10      ;OUTPUT BUFFER
2776 032646 012702 001555      MOV   #PUTINX,R2
2777 032652 112722 000000      MOVB  #RMCS1,(R2)+
2778 032656 112712 000200      MOVB  #200,(R2)
2779
2780 032662 004737 040360      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      032666 000404      BR    22$          ;GO TO 22$ IF NO ERROR
      032670 000240      NOP                    ;RETURN HERE IF ERROR
      032672 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      032674 000137 033074      JMP   28$          ;GO TO 28$ IF ERROR

2781 032700          22$:
2782
2783          ;WAIT FOR COMMAND TO COMPLETE
      032700 004737 040732      JSR   PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
2784
2785          ;EXECUTE WRITE CHECK DATA COMMAND
2786 032704 012737 000051 001412      MOV   #WCD!GO,RMCS10
2787 032712 012702 001555      MOV   #PUTINX,R2
2788 032716 112722 000000      MOVB  #RMCS1,(R2)+
2789 032722 112712 000200      MOVB  #200,(R2)
2790
2791 032726 004737 040360      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      032732 000404      BR    23$          ;GO TO 23$ IF NO ERROR
      032734 000240      NOP                    ;RETURN HERE IF ERROR
      032736 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      032740 000137 033074      JMP   28$          ;GO TO 28$ IF ERROR

2792 032744          23$:
2793
2794          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      032744 004737 040024      JSR   PC,GETSTS      ;GO TO GETSTS SUBROUTINE
2795
2796          ;WAIT FOR COMMAND TO COMPLETE
      032750 004737 040732      JSR   PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
2797
2798          ;READ STATUS FOR WRITE CHECK OFFSET OPERATION
2799 032754 004737 040110      JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      032760 000404      BR    24$          ;GO TO 24$ IF NO ERROR
      032762 000240      NOP                    ;RETURN HERE IF ERROR
      032764 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      032766 000137 033074      JMP   28$          ;GO TO 28$ IF ERROR
  
```



```

2800 032772          24$:
2801
2802
2803 032772 004737 041126 ;CHECK ERROR DURING WRITE CHECK OFFSET OPERATION
      032776 000405      JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
      033000 000240      BR 25$ ;GO TO 25$ IF NO ERROR
      033002 104000      NOP ;RETURN HERE IF ERROR
      033004 004736      EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
      033006 000137 033074 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 28$ IF ERROR
      JMP 28$

2804 033012          25$:
2805 033012 004737 054166 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
      033016 000405      BR 26$ ;GO TO 26$ IF NO ERROR
      033020 000240      NOP ;RETURN HERE IF ERROR
      033022 104000      EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
      033024 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      033026 000137 033074 JMP 28$ ;GO TO 28$ IF ERROR

2806 033032          26$:
2807 033032 004737 041760 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
      033036 000405      BR 27$ ;GO TO 27$ IF NO ERROR
      033040 000240      NOP ;RETURN HERE IF ERROR
      033042 104000      EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      033044 004736      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      033046 000137 033074 JMP 28$ ;GO TO 28$ IF ERROR

2808 033052          27$:
2809 033052 032737 000200 001444 BIT #OFD,RMOFO ;IN FORWARD DIRECTION OF OFFSET ?
2810 033060 001005      BNE 28$ ;BRANCH IF SO
2811 033062 052737 000200 001444 BIS #OFD,RMOFO ;SET TO FORWARD DIRECTION
2812 033070 000137 032210 JMP 7$ ;CHANGE OFFSET DIRECTION TEST AGAIN
2813 033074          28$:
2814
2815

```

```

*****
;*TEST 23          FORMAT FE CYLINDERS
*****
TST23:

```

```

033074 000004      SCOPE ;SCOPE CALL
033074 000240      NOP ;START OF TEST
033100 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
033104 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
033110 013701 001466 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
033114 012737 000001 001206 MOV #1,$TIMES ;DO 1 ITERATION
033122 012737 000023 001226 MOV #23,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

2816
2817 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
2818 033130 012737 001057 001446 MOV #559,RMDCO ;CYLINDER = 559.
2819 033136 012737 001000 001420 MOV #TA2,RMDAO ;TRACK = 2, SECTOR = 0
2820 033144          1$:
2821 033144 012737 011000 001444 MOV #FMT16!SSEI,RMOFO ;16 BIT FORMAT WITH SSEI SET
2822 033152 012737 177776 001414 MOV #-2,RMWCO ;2 WORDS (2'S COMP)
2823 033160 012737 104254 001416 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
2824 033166 012737 000062 001412 MOV #WH,RMCS10 ;WRITE HEADER AND DATA
2825
2826 ;VERIFY THAT SECTOR IS NOT BAD
033174 004737 034762 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
033200 000405      BR 2$ ;GO TO 2$ IF NO ERROR
033202 104401 066006 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
033206 104000      EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
033210 000137 033522 JMP 12$ ;GO TO 12$ IF ERROR

```

```

2827 033214          2$:
2828 033214 023727 001446 001060      CMP      RMDCO,#560.      ;IS DESIRED CYLINDER, OUTSIDE FE CYLINDERS ?
2829 033222 101402          BLOS     3$              ;NO !!
2830 033224 000137 033522          JMP      12$            ;YES, EXIT TEST
2831 033230          3$:
2832 033230 004737 037214          JSR      PC,GENBUF      ;GO GENERATE DATA BUFFER
2833 033234          4$:
2834 033234 005737 001514          TST      SSFENB         ;TEST SSF ENABLE FOR 1ST HEADER WORD
2835 033240 001403          BEQ      5$              ;NO, DO NOT SET 'SSF'
2836 033242 052737 020000 104254      BIS      #SSF,BUFONE    ;YES, SET SSF BIT
2837 033250          5$:
2838
2839          ;PREPARE DEVICE FOR DATA TRANSFER
2840 033250 004737 034032          JSR      PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      033254 154130          .WORD   154130        ;TASK DESCRIPTOR AS FOLLOWS:
      ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
      ;CLEAR CONTROLLER & SELECT DEVICE
      ;VERIFY CONTROLLER CLEAR OPERATION
      ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
      ;VERIFY PACK ACKNOWLEDGE
      ;RECALIBRATE IF 'SKI' OR 'PIP' IS SET
      ;VERIFY RECALIBRATION
      ;GO TO 6$ IF NO ERROR
      ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 12$ IF ERROR
      BR      6$
      NOP
      EMT
      JMP     12$
      033256 000404          BR      6$
      033260 000240          NOP
      033262 104000          EMT
      033264 000137 033522          JMP     12$
2841
2842          ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
2843          6$:
2844 033270 012737 000063 001412      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
2845 033276 012702 001555          MOV      #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
2846 033302 112722 000006          MOVB    #RMDA,(R2)+
2847 033306 112722 000034          MOVB    #RMDC,(R2)+
2848 033312 112722 000032          MOVB    #RMOF,(R2)+
2849 033316 112722 000002          MOVB    #RMWC,(R2)+
2850 033322 112722 000004          MOVB    #RMBA,(R2)+
2851 033326 112722 000000          MOVB    #RMCS1,(R2)+
2852 033332 112722 000200          MOVB    #200,(R2)+
2853
2854 033336 004737 040360          JSR      PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      033342 000404          BR      7$              ;GO TO 7$ IF NO ERROR
      033344 000240          NOP
      033346 104000          EMT
      033350 000137 033522          JMP     12$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 12$ IF ERROR
2855 033354          7$:
2856
2857          ;SETUP GET INDEX TABLE TO READ ALL REGISTERS
      JSR      PC,GETSTS   ;GO TO GETSTS SUBROUTINE
2858
2859          ;WAIT FOR COMMAND TO COMPLETE
      JSR      PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
2860
2861 033364 004737 040110          JSR      PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
      033370 000404          BR      8$              ;GO TO 8$ IF NO ERROR
      033372 000240          NOP
      033374 104000          EMT
      033376 000137 033522          JMP     12$            ;RETURN HERE IF ERROR
      ;ERROR # DEFINED BY GET SUBROUTINE
      ;GO TO 12$ IF ERROR
    
```



```

2862 033402      8$:
2863
2864      :VERIFY RESULTS OF WRITE COMMAND
2865 033402 004737 041126      JSR PC,PRIERR      :GO CHECK FOR PRIMARY ERRORS
      033406 000405          BR 9$              :GO TO 9$ IF NO ERROR
      033410 000240          NOP              :RETURN HERE IF ERROR
      033412 104000          EMT              :ERROR # DEFINED BY PRIERR SUBROUTINE
      033414 004736          JSR PC,a(SP)+    :GO BACK FOR MORE ERROR CHECKS
      033416 000137 033522      JMP 12$           :GO TO 12$ IF ERROR

2866 033422      9$:
2867 033422 004737 041760      JSR PC,SECERR     :GO CHECK FOR SECONDARY ERRORS
      033426 000405          BR 10$            :GO TO 10$ IF NO ERROR
      033430 000240          NOP              :RETURN HERE IF ERROR
      033432 104000          EMT              :ERROR # DEFINED BY SECERR SUBROUTINE
      033434 004736          JSR PC,a(SP)+    :GO BACK FOR MORE ERROR CHECKS
      033436 000137 033522      JMP 12$           :GO TO 12$ IF ERROR

2868 033442      10$:
2869
2870      :INCREMENT ADDRESS
2871 033442 005237 001420      INC RMDAO          :ADVANCE SECTOR COUNT
2872 033446 123727 001420 000037  CMPB RMDAO,#31.   :DONE ALL SECTORS ?
2873 033454 101420          BLOS 11$          :NO !!
2874
2875 033456 105037 001420      CLRB RMDAO        :START AT SECTOR 0 AND
2876 033462 105237 001421      INCB RMDAO+1     :ADVANCE TO NEXT TRACK.
2877 033466 123737 001421 001335  CMPB RMDAO+1,LSTRK+1 :DONE WITH LAST TRACK ?
2878 033474 101410          ELOS 11$          :NO !!
2879
2880 033476 105037 001421      CLRB RMDAO+1     :START WITH TRACK 0 AND
2881 033502 005237 001446      INC RMDCO         :ADVANCE TO NEXT CYLINDER.
2882 033506 023727 001446 001060  CMP RMDCO,#560.   :DONE WITH LAST CYLINDER ?
2883 033514 101002          BHI 12$          :YES !!
2884 033516 000137 033144      JMP 1$           :GO DO TEST AGAIN
2885
2886 033522      12$:
    
```


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

.SBTTL END OF SUB-PASS ROUTINE

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

033522	000004			\$EOSP:	SCOPE		
033524	000240				NOP		
033526	013700	001466			MOV	TSTQUE,R0	:GET POINTER TO TSTQUE
033532	062700	000002			ADD	#2,R0	:ADJUST POINTER TO NEXT DEVICE
033536	010037	001466			MOV	R0,TSTQUE	:SAVE POINTER TO TSTQUE
033542	005710				TST	(R0)	:ANY MORE DEVICES FOR TEST ?
033544	001402				BEQ	1\$:BR IF NO
033546	000137	033776			JMP	SHUT	:YES, JUMP TO 'SHUT' ROUTINE
033552	012737	001470	001466	1\$:	MOV	#TSTQUE+2,TSTQUE	:INITIALIZE POINTER TO FIRST DEVICE IN
							:TEST QUE TABLE

.SBTTL END OF PASS ROUTINE

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

033560	000240			\$EOP:	NOP		
033562	005037	001116			CLR	\$TSTNM	::ZERO THE TEST NUMBER
033566	005037	001206			CLR	\$TIMES	::ZERO THE NUMBER OF ITERATIONS
033572	005237	001230			INC	\$PASS	::INCREMENT THE PASS NUMBER
033576	042737	100000	001230		BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
033604	005327				DEC	(PC)+	::LOOP?
033606	000001			SEOPCT:	.WORD	1	
033610	003066				BGT	\$DOAGN	::YES
033612	012737				MOV	(PC)+,a(PC)+	::RESTORE COUNTER
033614	000001			SENDCT:	.WORD	1	
033616	033606				SEOPCT		
033620	104401	033626			TYPE	.65\$::TYPE ASCIZ STRING
033624	000407				BR	64\$::GET OVER THE ASCIZ
				::65\$:	.ASCIZ	<12><15>/END PASS #/	
033644	013746	001230		64\$:	MOV	\$PASS,-(SP)	::SAVE \$PASS FOR TYPEOUT
							::TYPE PASS NUMBER
033650	104405				TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
033652	005737	001126			TST	\$ERTTL	::SEE IF ANY ERRORS THIS PASS
033656	001431				BEQ	\$GT42P	::BR IF NO ERRORS TO REPORT
033660	104401	033666			TYPE	.67\$::TYPE ASCIZ STRING
033664	000421				BR	66\$::GET OVER THE ASCIZ
				::67\$:	.ASCIZ	/ TOTAL ERRORS SINCE LAST REPORT /	
033730	013746	001126		66\$:	MOV	\$ERTTL,-(SP)	::SAVE \$ERTTL FOR TYPEOUT
							::TOTAL NUMBER OF ERRORS
033734	104405				TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
033736	005037	001126			CLR	\$ERTTL	::CLEAR ERROR TOTAL

033742	104401	001217		\$GT42P: TYPE	,SCRLF	::TYPE CARRIAGE RETURN, LINE FEED
033746	013700	000042		\$GET42: MOV	@#42,R0	::GET MONITOR ADDRESS
033752	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
033754	000005			RESET		::CLEAR THE WORLD
033756	004710			\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
033760	000240			NOP		::SAVE ROOM
033762	000240			NOP		::FOR
033764	000240			NOP		::ACT11
033766				\$DOAGN:		
033766	000137			JMP	@(PC)+	::RETURN
033770	033776			\$RTNAD: .WORD	SHUT	
033772	377	377	000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
				.EVEN		
21						
22	033776	005737	001326	SHUT: TST	CTLFG	:WAS CONTROL C FLAGGED ?
23	034002	001002		BNE	1\$:BR IF YES
24	034004	000137	007524	JMP	READY	:CONTINUE
25	034010	005737	000042	1\$: TST	@#42	:ANY MONITOR PRESENT ?
26	034014	001002		BNE	2\$:BR IF YES
27	034016	000137	005434	JMP	START	:GO TO START
28	034022	005037	001300	2\$: CLR	\$DEVN	:FUDGE NO DRIVES IN MAP
29	034026	000137	033560	JMP	\$EOP	:RETURN TO \$EOP

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

.SBTTL TEST PREPARATION MODULE

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

:CALL:

```

    JSR    PC,TSTPRP          TASK/VERIFY DESCRIPTOR
    .WORD  NNNNNN            RETURN HERE IF NO ERROR
    BR     ??                 RETURN HERE IF ERROR
    NOP
    ERROR  ERROR DEFINED BY MODULE
    
```

:TASK/VERIFY DESCRIPTOR

```

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

TSTPRP:

:STORE TASK DESCRIPTOR AND CLEAR USER'S ERROR CALL

```

    034032 017637 000000 034756  MOV    @ (SP),39$      :STORE DESCRIPTOR
    034040 062716 000006          ADD    #6,(SP)        :MOVE SP TO USERS ERROR CALL
    034044 105076 000000          CLRB  @ (SP)         :CLEAR ERROR CALL
    034050 162716 000004          SUB    #4,(SP)       :MOVE SP TO NO ERROR RETURN
    034054 004737 040024          JSR    PC,GETSTS     :SETUP TO READ ALL REGISTERS
    034060 004737 040110          JSR    PC,GET        :GET RMER2
    034064 000411          BR     2$           :BR IF NO ERROR DETECTED
    034066 000401          BR     1$           :GET OVER ERROR NUMBER
    034070 000000          .WORD 0            :ERROR DEFINED BY GET SUBROUTINE
    034072 062716 000004 1$:    ADD    #4,(SP)       :XFER ERROR TO USER AND
    034076 113776 034070 000000  MOVB  1$-2,@(SP)    :GET ERROR NUMBER.
    034104 000137 034746          JMP    37$
    034110 013737 001400 034760 2$:    MOV    RMER2I,40$   :GET RMER2 AND SAVE FOR LATER
    
```

:*****


```

58                                     ;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
59 034116 005737 034756                TST      39$      ;SELECT DEVICE??
60 034122 100014                        BPL      4$      ;NO!!
61
62 034124 004737 046336                JSR      PC,DEVSEL ;GO SELECT DEVICE
63 034130 000411                        BR       4$      ;NO ERROR - CONTINUE
64 034132 000401                        BR       3$
65 034134 000000                        .WORD   0        ;ERROR NUMBER FROM DEVSEL
66 034136 062716 000004 000000 3$:   ADD     #4,(SP)   ;TRANSFER ERROR TO USER
67 034142 113776 034134 000000      MOVB    3$-2,a(SP)
68 034150 000137 034746                JMP     37$
69
70                                     ;*****
71                                     ;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
72 034154 032737 040000 034756 4$:   BIT     #BIT14,39$ ;CLEAR CONTROLLER??
73 034154 001451                        BEQ     13$      ;NO!!
74
75                                     ;*****
76 034164 004737 050030                JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
77 034170 000411                        BR       7$      ;CONTINUE - NO ERROR
78 034172 000401                        BR       6$
79 034174 000000                        .WORD   0        ;ERROR NUMBER FROM CNTCLR
80 034176 062716 000004 000000 5$:   ADD     #4,(SP)   ;TRANSFER ERROR TO USER
81 034202 113776 034174 000000 6$:   MOVB    5$,a(SP)
82 034210 000137 034746                JMP     37$
83
84                                     ;*****
85                                     ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
86 034214 032737 000100 034756 7$:   BIT     #BIT6,39$ ;VERIFY??
87 034214 001431                        BEQ     13$      ;NO!!
88
89                                     ;*****
90 034224 004737 040110                JSR      PC,GET    ;GO GET STATUS
91 034230 000411                        BR      10$      ;NO ERROR GETTING STATUS
92 034232 000401                        BR      9$
93 034234 000000                        .WORD   0        ;ERROR FROM GETTING STATUS
94 034236 062716 000004 000000 8$:   ADD     #4,(SP)   ;TRANSFER ERROR TO USER
95 034242 113776 034234 000000 9$:   MOVB    8$,a(SP)
96 034250 000137 034746                JMP     37$
97
98 034254 004737 050146 10$:   JSR     PC,CLRSTS ;GO VERIFY STATUS CLEAR
99 034260 000412                        BR      13$      ;NO ERROR IN CLEAR
100 034262 000401                        BR      12$
101 034264 000000                        .WORD   0        ;ERROR IN STATUS CLEAR
102 034266 005726                        TST     (SP)+    ;STRIP RETURN ADDRESS TO
103 034270 062716 000004 000000 11$:  ADD     #4,(SP)  ;SUBROUTINE AND TRANSFER
104 034274 113776 034264 000000 12$:  MOVB    11$,a(SP) ;ERROR TO USER
105 034302 000137 034746                JMP     37$
106
107                                     ;*****
108                                     ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
109                                     ;NOT VALID
110 034306 032737 010000 034756 13$:  BIT     #BIT12,39$ ;PACK ACKNOWLEDGE??
111 034306 001501                        BEQ     25$      ;NO!!
112
113
114 034316 004737 040110                JSR     PC,GET
    
```

TEST PREPARATION MODULE

```

115 034322 000411 BR 16$ ;NO ERROR GETTING RMDS
116 034324 000401 BR 15$
117 034326 000000 14$: .WORD 0
118 034330 062716 000004 15$: ADD #4,(SP) ;TRANSFER ERROR TO USER
119 034334 113776 034326 000000 MOVB 14$,a(SP)
120 034342 000137 034746 JMP 37$
121
122 034346 032737 000100 001350 16$: BIT #VV,RMDSI ;IS VOLUME VALID??
123 034354 001061 BNE 25$ ;YES!!
124
125 034356 012737 000023 001412 MOV #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
126 034364 112737 000000 001555 MOVB #RMCS1,PUTINX ;SETUP REGISTER INDEX TABLE
127 034372 112737 000200 001556 MOVB #200,PUTINX+1
128 034400 004737 040360 JSR PC,PUT ;GO WRITE COMMAND
129 034404 000410 BR 19$ ;NO ERROR LOADING REGISTER
130 034406 000401 BR 18$
131 034410 000000 17$: .WORD 0 ;ERROR FROM PUT SUB
132 034412 062716 000004 18$: ADD #4,(SP) ;TRANSFER ERROR TO USER
133 034416 113776 034410 000000 MOVB 17$,a(SP)
134 034424 000550 BR 37$
135
136 034426 004737 040732 19$: JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
137
138 :*****
139 :VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
140 034432 032737 000020 034756 BIT #BIT4,39$ ;VERIFY PACK ACKNOWLEDGE??
141 034440 001427 BEQ 25$ ;NO!!
142
143 034442 004737 040110 JSR PC,GET ;GO GET STATUS
144 034446 000410 BR 22$ ;NO ERROR GETTING STATUS
145 034450 000401 BR 21$
146 034452 000000 20$: .WORD 0 ;ERROR FROM GET SUB
147 034454 062716 000004 21$: ADD #4,(SP) ;TRANSFER ERROR TO USER
148 034460 113776 034452 000000 MOVB 20$,a(SP)
149 034466 000527 BR 37$
150
151 034470 004737 051026 22$: JSR PC,ACKSTS ;GO CHECK ACKNOWLEDGE
152 034474 000411 BR 25$ ;NO ERROR
153 034476 000401 BR 24$
154 034500 000000 23$: .WORD 0 ;PACK ACKNOWLEDGE ERROR
155 034502 005726 24$: TST (SP)+ ;STRIP RETURN TO SUB AND
156 034504 062716 000004 ADD #4,(SP) ;TRANSFER ERROR TO USER
157 034510 113776 034500 000000 MOVB 23$,a(SP)
158 034516 000513 BR 37$
159
160 :*****
161 :RECALIBRATE DRIVE IF BIT 11 IS SET IN TASK AND "SKI" IS SET
162 :OR "PIP" IS ACTIVE.
163 :RECALIBRATE DRIVE IF BIT 10 IS SET
164 034520 25$:
165 034520 032737 002000 034756 BIT #BIT10,39$ ;RECALIBRATE DRIVE ?
166 034526 001027 BNE 29$ ;YES!!
167 034530 032737 004000 034756 BIT #BIT11,39$ ;RECALIBRATE??
168 034536 001505 BEQ 38$ ;NO!!
169
170 034540 004737 040110 JSR PC,GET ;GO GET RMDS
171 034544 000410 BR 28$ ;NO ERROR GETTING RMDS

```



```

172 034546 000401          BR      27$
173 034550 000000          .WORD  0          :ERROR FROM GET SUB
174 034552 062716 000004 26$:      ADD    #4,(SP)   :TRANSFER ERROR TO USER
175 034556 113776 034550 000000 27$:      MOV    26$,a(SP)
176 034564 000470          BR      37$
177
178 034566 032737 040000 034760 28$:      BIT    #SKI,40$   :WAS SKI SET ?
179 034574 001004          BNE    29$        :YES, GO RECALIBRATE
180 034576 032737 020000 001350          BIT    #PIP,RMDSI :IS PIP ACTIVE??
181 034604 001462          BEQ    38$        :NO!!
182
183 034606 012737 000007 001412 29$:      MOV    #RECAL!GO,RMCS10 :LOAD RECALIBRATE COMMAND
184 034614 112737 000000 001555          MOV    #RMCS1,PUTINX  :AND REGISTER INDEX
185 034622 112737 000200 001556          MOV    #200,PUTINX+1 :SET TERMINATOR
186 034630 004737 040360          JSR    PC,PUT      :GO ISSUE RECALIBRATE
187 034634 000410          BR     31$        :NO ERROR
188 034636 000401          BR     30$
189 034640 000000          .WORD  0          :ERROR IN REGISTER TRANSFER
190 034642 062716 000004 30$:      ADD    #4,(SP)   :TRANSFER ERROR TO USER
191 034646 113776 034640 000000          MOV    30$,2,a(SP)
192 034654 000434          BR     37$
193
194 034656 004737 040732 31$:      JSR    PC,TIMOUT   :WAIT FOR COMPLETION
195
196          :*****
197          :VERIFY RECALIBRATE IF BIT 3 SET IN TASK
198 034662 032737 000010 034756          BIT    #BIT3,39$   :VERIFY RECALIBRATE??
199 034670 001430          BEQ    38$        :NO!!
200
201 034672 004737 040110          JSR    PC,GET      :GO GET STATUS
202 034676 000410          BR     34$        :NO ERROR GETTING STATUS
203 034700 000401          BR     33$
204 034702 000000          32$:      .WORD  0          :ERROR FROM GET
205 034704 062716 000004 33$:      ADD    #4,(SP)   :TRANSFER ERROR TO USER
206 034710 113776 034702 000000          MOV    32$,a(SP)
207 034716 000413          BR     37$
208
209 034720 004737 051622 34$:      JSR    PC,RCLSTS  :GO CHECK RECALIBRATE
210 034724 000412          BR     38$        :NO ERROR DURING RECALIBRATE
211 034726 000401          BR     36$
212 034730 000000          35$:      .WORD  0          :ERROR DURING RECALIBRATE
213 034732 005726          36$:      TST    (SP)+     :STRIP RETURN TO SUB AND
214 034734 062716 000004          ADD    #4,(SP)   :TRANSFER ERROR TO USER
215 034740 113776 034730 000000          MOV    35$,a(SP)
216 034746 162716 000002 37$:      SUB    #2,(SP)   :MOVE SP BACK BEFORE ERROR
217 034752 000240          38$:      NOP
218 034754 000207          RTS    PC        :RETURN TO USER
219
220 034756 000000          39$:      .WORD  0          :TASK/VERIFY DESCRIPTOR
221 034760 000000          40$:      .WORD  0          :CONTAINS RMER2
    
```


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

```
.SBTTL BAD SECTOR MODULE

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

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

;THE BAD SECTOR MODULE PERFORMS TWO MAJOR FUNCTIONS:
; (1) RECOVER THE BAD SECTOR FILES AND
; (2) APPROVE THE USAGE OF A SECTOR BASED ON INFORMATION IN
; THE BAD SECTOR FILES OR ASSIGN A NEW SECTOR IF THE ONE
; ELECTED IS NOT AVAILABLE FOR USE.

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

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

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

;TEST 'MEDIA ENABLE' TO DETERMINE WHETHER OR NOT THE BAD SECTOR FILES
;HAVE BEEN RECOVERED.
; TST MEDENB ;HAVE BAD SECTOR FILES BEEN RECOVERED ?
; BEQ 1$ ;BR IF NO
; JMP 56$ ;YES, BAD SECTOR FILE IS AVAILABLE

;RECOVER SKIP SECTOR FILE FROM CYLINDER = 559., TRACK = 0 AND
;RECOVER THE MANUFACTURES / USERS BAD SECTOR FILE FROM CYLINDER = 558.,
;TRACK = LAST TRACK (RMB0 = 13.). ALSO, SAVE THE USER'S PUT BUFFER
1$:
; MOV R0,-(SP) ;:PUSH R0 ON STACK
; CLR R0 ;:START WITH RMCS1
2$:
; MOV PUTBUF(R0),BUFFER(R0)
; ADD #2,R0 ;:ADVANCE TO NEXT BUFFER POSITION
; CMP #46,R0 ;:END OF BUFFER
; BHS 2$ ;NO !!

;SET RETRY COUNT AND LOAD PUT BUFFER AND REGISTER INDEX TABLE
;SETUP PARAMETERS TO READ SKIP SECTOR FILE FIRST (3 SECTORS)
; MOV #3,74$ ;:RETRY COUNT
; MOV #559.,RMDCO ;:DESIRED CYLINDER = 559.
```

```
034762 062716 000006
034766 105076 000000
034772 162716 000006

034776 005737 001512
035002 001402
035004 000137 036340

035010
035010 010046
035012 005000
035014 016060 001412 104254
035022 062700 000002
035026 022700 000046
035032 103370

035034 012737 000003 037054
035042 012737 001057 001446
```

BAD SECTOR MODULE

```

57 035050 012737 000000 001420      MOV      #0,RMDAO      ;STARTING TRACK = 0, SECTOR = 0
58 035056 012737 176372 001414      MOV      #-258,*3,RMWCO ;258. * 3 SECTORS (2'S COMP)
59 035064 012737 010000 001444      MOV      #FMT16,RMOFO  ;16 BIT FORMAT
60 035072 012737 105260 001416      MOV      #BUFTWO,RMBAO ;SETUP BUS ADDRESS TO READ SKIP SECTOR FILE
61
62 035100 012700 001555          MOV      #PUTINX,R0    ;R0 POINTS TO REGISTER INDEX TABLE
63 035104 112720 000006      MOV      #RMDA,(R0)+
64 035110 112720 000034      MOV      #RMDC,(R0)+
65 035114 112720 000002      MOV      #RMWC,(R0)+
66 035120 112720 000032      MOV      #RMOF,(R0)+
67 035124 112720 000004      MOV      #RMBA,(R0)+
68 035130 112720 000000      MOV      #RMCS1,(R0)+
69 035134 112720 000200      MOV      #200,(R0)+
70 035140 012600          MOV      (SP)+,R0      ;:POP STACK INTO R0
71
72          ;SET GET INDEX TABLE FOR READING STATUS
73 035142          3$:
74 035142 004737 040024      JSR      PC,GETSTS    ;SETUP GET INDEX REGISTER FOR STATUS
75 035146 004737 040110      JSR      PC,GET      ;GET REGISTERS
76 035152 000411          BR      5$           ;BR IF NO ERROR
77 035154 000401          BR      4$           ;JUMP OVER ERROR NUMBER
78 035156 000000          .WORD   0           ;ERROR DEFINED BY GET SUB
79 035160 062716 000006      4$:      ADD      #6,(SP)     ;XFER ERROR TO USER AND
80 035164 113776 035156 000000      MOV      4$,2,a(SP)  ;GET ERROR NUMBER.
81 035172 000137 035766      JMP      42$
82
83 035176 013737 001400 037066      5$:      MOV      RMER2I,79$   ;GET RMER2 AND SAVE FOR LATER
84
85          ;CLEAR THE DEVICE USING DRIVE CLEAR COMMAND
86 035204 012737 000011 001412      MOV      #DRVCLR!GO,RMCS10 ;LOAD COMMAND IN PUT BUFFER
87 035212 004737 040360      JSR      PC,PUT      ;OUTPUT COMMAND
88 035216 000411          BR      8$           ;RETURN HERE IF NO ERROR
89 035220 000401          BR      7$           ;GET AROUND ERROR #
90 035222 000000          6$:      .WORD   0           ;ERROR # GOES HERE
91 035224 062716 000006      7$:      ADD      #6,(SP)     ;MOVE SP TO USERS ERROR CALL
92 035230 113776 035222 000000      MOV      6$,a(SP)    ;MOVE ERROR NUMBER TO USER
93 035236 000137 035766      JMP      42$
94
95 035242 004737 040732      8$:      JSR      PC,TIMOUT   ;WAIT FOR COMPLETION
96 035246 004737 040110      JSR      PC,GET      ;GO GET STATUS
97 035252 000411          BR      11$          ;RETURN HERE IF NO ERROR
98 035254 000401          BR      10$          ;GET AROUND ERROR #
99 035256 000000          9$:      .WORD   0           ;ERROR # GOES HERE
100 035260 062716 000006      10$:     ADD      #6,(SP)     ;MOVE SP TO USERS ERROR CALL
101 035264 113776 035256 000000      MOV      9$,a(SP)    ;MOVE ERROR # TO USERS ERROR CALL
102 035272 000137 035766      JMP      42$
103
104 035276 004737 053364      11$:     JSR      PC,DRVSTS   ;GO VERIFY DRIVE CLEAR COMMAND
105 035302 000412          BR      14$          ;RETURN HERE IF NO ERROR
106 035304 000401          BR      13$          ;GET AROUND ERROR
107 035306 000000          12$:     .WORD   0           ;ERROR # GOES HERE
108 035310 005726          13$:     TST      (SP)+     ;STRIP RETURN TO SUBROUTINE
109 035312 062716 000006      ADD      #6,(SP)     ;MOVE SP TO USERS ERROR CALL
110 035316 113776 035306 000000      MOV      12$,a(SP)   ;MOVE ERROR # TO USER CALL
111 035324 000137 035766      JMP      42$
112
113          ;ISSUE A PACK ACKNOWLEDGE IF VOLUME VALID IS RESET

```


BAD SECTOR MODULE

```

114 035330
115 035330 032737 000100 001350 14$: BIT #VV,RMDSI ;IS VV RESET ??
116 035336 001052 BNE 23$ ;NO !!
117
118 035340 012737 000023 001412 MOV #PACACK!GO,RMCS10 ;LOAD COMMAND
119 035346 004737 040360 JSR PC,PUT ;GO PUT COMMAND TO DRIVE
120 035352 000411 BR 17$ ;RETURN HERE IF NO OUTPUT ERROR
121 035354 000401 BR 16$ ;GET AROUND ERROR #
122 035356 000000 .WORD 0 ;ERROR # GOES HERE
123 035360 062716 000006 15$: ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
124 035364 113776 035356 000000 16$: MOVB 15$,a(SP) ;MOVE ERROR # TO ERROR CALL
125 035372 000137 035766 JMP 42$
126
127 035376 004737 040732 17$: JSR PC,TIMOUT ;WAIT FOR COMPLETION
128 035402 004737 040110 JSR PC,GET ;GO GET STATUS FOR PACK ACK
129 035406 000411 BR 20$ ;RETURN HERE IF NO ERROR
130 035410 000401 BR 19$ ;GET AROUND ERROR #
131 035412 000000 .WORD 0 ;ERROR # GOES HERE
132 035414 062716 000006 18$: ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
133 035420 113776 035412 000000 19$: MOVB 18$,a(SP) ;MOVE ERROR # TO CALL
134 035426 000137 035766 JMP 42$
135
136 035432 004737 051026 20$: JSR PC,ACKSTS ;GO VERIFY ACKNOWLEDGE STATUS
137 035436 000412 BR 23$ ;RETURN HERE IF NO ERROR
138 035440 000401 BR 22$ ;GET AROUND ERROR #
139 035442 000000 .WORD 0 ;ERROR # GOES HERE
140 035444 005726 21$: TST (SP)+ ;STRIP RETURN TO SUBROUTINE
141 035446 062716 000006 22$: ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
142 035452 113776 035442 000000 MOVB 21$,a(SP) ;MOVE ERROR # TO USERS ERROR CALL
143 035460 000137 035766 JMP 42$
144
145 ;RECALIBRATE THE DRIVE IF "SKI" OR "PIP IS SET
23$:
146 035464 BIT #SKI,79$ ;WAS SKI SET ?
147 035464 032737 040000 037066 BNE 24$ ;YES, GO RECALIBRATE
148 035472 001004 BIT #PIP,RMDSI ;IS PIP SET ??
149 035474 032737 020000 001350 BEQ 32$ ;NO !!
150 035502 001452
151
152 035504 012737 000007 001412 24$: MOV #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
153 035512 004737 040360 JSR PC,PUT ;PUT THE RECAL COMMAND
154 035516 000411 BR 26$ ;RETURN HERE IF NO ERROR
155 035520 000401 BR 25$ ;GET AROUND ERROR #
156 035522 000000 .WORD 0 ;ERROR # GOES HERE
157 035524 062716 000006 25$: ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
158 035530 113776 035522 000000 MOVB 25$-2,a(SP) ;MOVE ERROR # TO USERS CALL
159 035536 000137 035766 JMP 42$
160
161 035542 004737 040732 26$: JSR PC,TIMOUT ;WAIT FOR RECALIBRATE TO COMPLETE
162 035546 004737 040110 JSR PC,GET ;GO GET RECAL STATUS
163 035552 000411 BR 29$ ;RETURN HERE IF NO ERROR
164 035554 000401 BR 28$ ;GET AROUND ERROR #
165 035556 000000 .WORD 0 ;ERROR # GOES HERE
166 035560 062716 000006 27$: ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
167 035564 113776 035556 000000 28$: MOVB 27$,a(SP) ;MOVE ERROR TO USERS CALL
168 035572 000137 035766 JMP 42$
169
170 035576 004737 051622 29$: JSR PC,RCLSTS ;GO VERIFY RECALIBRATE STATUS

```



```

171 035602 000412          BR      32$          ;RETURN HERE IF NO ERROR
172 035604 000401          BR      31$          ;GET AROUND ERROR #
173 035606 000000          30$: .WORD 0          ;ERROR # GOES HERE
174 035610 005726          31$: TST  (SP)+        ;STRIP RETURN TO SUBROUTINE
175 035612 062716 000006          ADD  #6,(SP)        ;MOVE SP TO USERS ERROR CALL
176 035616 113776 035606 000000          MOVB 30$,a(SP)      ;MOVE ERROR # TO USERS CALL
177 035624 000137 035766          JMP   42$
178
179          ;READ THE SECTOR IDENTIFIED BY RMDAO, INCLUDING HEADER AND DATA
180 035630          32$:
181 035630 012737 000073 001412          MOV  #RH!GO,RMCS10 ;LOAD READ HEADER AND DATA COMMAND
182 035636 004737 040360          JSR  PC,PUT         ;PUT COMMAND
183 035642 000411          BR   35$           ;RETURN HERE IF NO ERROR
184 035644 000401          BR   34$           ;GET AROUND ERROR #
185 035646 000000          33$: .WORD 0          ;ERROR # GOES HERE
186 035650 062716 000006          34$: ADD  #6,(SP)        ;MOVE SP TO USERS ERROR CALL
187 035654 113776 035646 000000          MOVB 33$,a(SP)      ;MOVE ERROR # TO USERS ERROR CALL
188 035662 000137 035766          JMP   42$
189
190 035666 004737 040732          35$: JSR  PC,TIMOUT      ;WAIT FOR READ OPERATION TO COMPLETE
191 035672 004737 040110          JSR  PC,GET         ;GO GET STATUS FOR READ OPERATION
192 035676 000411          BR   38$           ;RETURN HERE IF NO ERROR
193 035700 000401          BR   37$           ;GET AROUND ERROR #
194 035702 000000          36$: .WORD 0          ;ERROR # GOES HERE
195 035704 062716 000006          37$: ADD  #6,(SP)        ;MOVE SP TO USERS ERROR CALL
196 035710 113776 035702 000000          MOVB 36$,a(SP)      ;MOVE ERROR # TO CALL
197 035716 000137 035766          JMP   42$
198
199 035722 004737 054166          38$: JSR  PC,DTASTS      ;GO VERIFY RESULTS OF READ OPERATION
200 035726 000412          BR   41$           ;RETURN HERE IF NO ERROR
201 035730 000401          BR   40$           ;GET AROUND ERROR #
202 035732 000000          39$: .WORD 0          ;ERROR # GOES HERE
203 035734 005726          40$: TST  (SP)+        ;STRIP RETURN ADDRESS TO SUBROUTINE
204 035736 062716 000006          ADD  #6,(SP)        ;MOVE SP TO USERS ERROR CALL
205 035742 113776 035732 000000          MOVB 39$,a(SP)      ;MOVE ERROR # TO USERS CALL
206 035750 000137 035766          JMP   42$
207
208 035754 032737 040000 001336 41$: BIT  #TRE,RMCS1I    ;ANY CONTROLLER ERRORS ?
209 035762 001001          BNE  42$           ;BR IF YES
210 035764 000477          BR   51$           ;NO ERRORS DETECTED
211
212          ;*****
213          ;AN ERROR HAS BEEN DETECTED IN TRYING TO READ THE BAD SECTOR FILE.
214          ;THE SECTOR WILL BE RETRIED IF POSSIBLE.
215
216 035766 005337 037054          42$: DEC  74$          ;YES, DECREMENT RETRY COUNT AND
217 035772 001057          BNE  48$          ;RETRY IF COUNT IS POSITIVE
218
219          ;THE RETRY COUNT HAS EXPIRED - SEE IF THE ERROR IS MEDIA RELATED
220
221 035774 032737 100720 001352          BIT  #DCK!HCRC!HCE!FER!ECH,RMER1I ;ANY MEDIA RELATED ERRORS ?
222 036002 001004          BNE  43$          ;YES, GO TRY NEXT AVAILABLE SECTOR
223
224 036004 032737 100040 001400          BIT  #BSE!SSE,RMER2I ;ANY MEDIA RELATED ERRORS ?
225 036012 001453          BEQ  49$          ;NO, EXIT AND REPORT ERROR ON RETURN
226
227          ;THE ERRORS DETECTED WHILE TRYING TO RECOVER THE BAD SECTOR FILE ARE
    
```

```
228                                     ;DUE TO THE MEDIA. SEE IF THE BAD SECTOR FILE CAN BE RECOVERED FROM
229                                     ;ANOTHER AREA ON THE LAST TRACK
230
231 036014 032737 000040 001400 43$: BIT #SSE,RMER2I ;SSE ERROR ?
232 036022 001406 BEQ 44$ ;NO !!
233 036024 005237 001420 INC RMDAO ;DISPLACE SECTOR ADDRESS BY 1 TO
234 ;ACCOUNT FOR THE SKIP SECTOR ERROR
235 036030 052737 001000 001444 BIS #SSE1,RMOFO ;AND SET SSE1 IN OFFSET REGISTER.
236 036036 000432 BR 47$ ;TRY AGAIN
237
238 036040 062737 000002 001420 44$: ADD #2,RMDAO ;ADVANCE SECTOR ADDRESS BY 2
239 036046 022737 001056 001446 CMP #558.,RMDCO ;READING DEC 144 BAD SECTOR FILE ?
240 036054 001407 BEQ 45$ ;YES !!
241 036056 005237 001420 INC RMDAO ;IF READING SSF, THEN ADVANCE SECTOR ADDRESS BY 1 MORE
242 036062 122737 000017 001420 CMPB #15.,RMDAO ;QUIT IF ALL SSF SECTORS HAVE BEEN
243 036070 101424 BLOS 49$ ;TRIED.
244 036072 000414 BR 47$
245
246 036074 022737 106264 001416 45$: CMP #MFGFIL,RMBAO ;READING MANUFACTURES BAD FILE ?
247 036102 001004 BNE 46$ ;NO !!
248 036104 122737 000012 001420 CMPB #10.,RMDAO ;QUIT IF ALL MFG SECTORS HAVE BEEN
249 036112 101413 BLOS 49$ ;TRIED.
250 036114 122737 000040 001420 46$: CMPB #32.,RMDAO ;QUIT IF ALL USER SECTORS HAVE BEEN
251 036122 101407 BLOS 49$ ;TRIED.
252
253 036124 012737 000003 037054 47$: MOV #3,74$ ;REINSTATE RETRY COUNT FOR THIS SECTOR
254 036132 162716 000006 48$: SUB #6,(SP) ;MOVE SP BACK TO NO ERROR
255 036136 000137 035142 JMP 3$ ;RETRY THE READ OPERATION
256
257                                     ;THE BAD SECTOR FILE CANNOT BE READ
258 036142 49$: NOP
259 036142 000240
260 036144 032777 020000 143002 BIT #SW13,@SWR ;INHIBIT MESSAGE ?
261 036152 001002 BNE 50$ ;YES
262 036154 162716 000004 SUB #4,(SP) ;MOVE SP TO ERROR RETURN
263 036160 000137 037050 50$: JMP 73$ ;GO TO MODULE EXIT
264
265                                     ;THE SECTOR WAS RECOVERED WITHOUT ERROR - READ THE MFG/USER FILE IF
266                                     ;THIS IS THE SSF FILE OR ELSE DONE.
267 036164 51$:
268 036164 022737 001056 001446 CMP #558.,RMDCO ;READING DEC 144 FILES ?
269 036172 001422 BEQ 52$ ;YES !!
270 036174 004737 037070 JSR PC,GETSSF ;GET SKIP SECTOR FILE FOR USE
271
272 036200 012737 001056 001446 MOV #558.,RMDCO ;READ DEC 144 FILE, CYLINDER 558.
273 036206 013737 001334 001420 MOV LSTRK,RMDAO ;MFG FILE LAST TRACK, SECTOR = 0
274 036214 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
275 036222 012737 106264 001416 MOV #MFGFIL,RMBAO ;POINT TO MANUFACTURES FILE BUFFER
276 036230 012737 177376 001414 MOV #-258.,RMWCO ;2 + 256. WORDS (2'S COMP)
277 036236 000415 BR 53$ ;GO READ MFG FILE
278 036240 52$:
279 036240 022737 107272 001416 CMP #USRFIL,RMBAO ;WAS THE USER FILE READ ??
280 036246 001416 BEQ 54$ ;YES - READ IS COMPLETE
281 036250 112737 000012 001420 MOVB #10.,RMDAO ;READ THE USER FILE LAST TRACK, SECTOR = 10.
282 036256 012737 107272 001416 MOV #USRFIL,RMBAO ;POINT TO USERS FILE BUFFER
283 036264 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
284
```


BAD SECTOR MODULE

```

285 036272 012737 000003 037054 53$:  MOV    #3,74$      ;RELOAD THE RETRY COUNT FOR THIS SECTOR
286 036300 000137 035142          JMP    3$          ;GO READ THE USER FILE
287
288          ;SET MEDIA ENABLE AND RESTORE THE USERS PUT BUFFER
289          54$:
036304          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
036304 010046          CLR    R0          ;;R0 IS REGISTER INDEX
290 036306 005000          MOV    #-1,MEDENB
291 036310 012737 177777 001512          MOV    BUFFER(R0),PUTBUF(R0)
292 036316 016060 104254 001412 55$:  ADD    #2,R0      ;ADVANCE R0
293 036324 062700 000002          CMP    #46,R0     ;DONE ??
294 036330 022700 000046          BHS   55$
295 036334 103370          MOV    (SP)+,R0   ;;POP STACK INTO R0
296 036336 012600
297
298          ;*****
299          ;VERIFY THAT THE DESIRED SECTOR IS NOT IN THE MFG BAD SECTOR FILE,
300          ;NOT IN THE USERS BAD SECTOR FILE AND NOT IN THE SSF BAD SECTOR FILE.
301          ;ASSIGN A NEW SECTOR IF THE SECTOR IS IN ANY OF THE FILES.
302          ;*****
303
304          ;LOAD INITIAL VARIABLES AND COMPUTE THE NUMBER OF SECTORS
305          56$:
036340          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
036340 010046          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
036342 010146          MOV    R2,-(SP)      ;;PUSH R2 ON STACK
036344 010246          MOV    RMDCO,ASNDC   ;LOAD REQUESTED CYLINDER, TRACK,
306 036346 013737 001446 001516          MOV    RMDAO,ASNDA   ;AND SECTOR ADDRESS IN ASSIGNED STORAGE
307 036354 013737 001420 001520          CLR    R2          ;R2 = NUMBER OF SECTORS
308 036362 005002          MOV    RMWCO,R0     ;R0 = WORD COUNT
309 036364 013700 001414          NEG   R0          ;MAKE NUMBER POSITIVE
310 036370 005400          MOV    #256,R1     ;R1 = NUMBER OF WORDS PER SECTOR
311 036372 012701 000400          BIT    #BIT1,RMCS10 ;IS THIS A HEADER AND DATA COMMAND ??
312 036376 032737 000002 001412          BEQ   57$         ;NO !!
313 036404 001402          MOV    #258,R1     ;CHANGE WORDS PER SECTOR
314 036406 012701 000402          CMP   R1,R0       ;IS THERE A FULL SECTOR ??
315 036412 020100 57$:  BLOS  58$         ;YES !!
316 036414 101404          TST   R0          ;IS R0 ZERO ??
317 036416 005700          BEQ   59$         ;YES !!
318 036420 001405          INC   R2          ;INCREMENT FOR PARTIAL SECTOR
319 036422 005202          BR    59$
320 036424 000403          SUB   R1,R0       ;SUBTRACT ONE SECTOR FROM WORD COUNT
321 036426 160100 58$:  INC   R2          ;INCREMENT SECTOR COUNT
322 036430 005202          BR    57$
323 036432 000767          MOV   R2,74$     ;SAVE SECTOR COUNT
324 036434 010237 037054 59$:
325
326          ;LOAD PARAMETERS AND SEARCH THE MFG, USER AND THE MAPPED SSF BAD
327          ;SECTOR FILES FOR THE ASSIGNED SECTOR. ALSO, SEARCH THE ADJACENT
328          ;SECTORS IF THE SECTOR COUNT IS MORE THAN ONE.
329
330 036440 012737 106300 037064          MOV   #MFGFIL+14,78$ ;THE STARTING ADDRESS OF MFG FILE
331          ;TO BASE ADDRESS STORAGE.
332 036446 004737 036506          JSR   PC,60$      ;GO SEARCH FILE
333 036452 012737 107306 037064          MOV   #USRFIL+14,78$ ;LOAD STARTING ADDRESS OF USR FILE
334          ;TO BASE ADDRESS STORAGE.
335 036460 004737 036506          JSR   PC,60$      ;GO SEARCH FILE
336 036464 012737 110304 037064          MOV   #SSFIL+4,78$ ;LOAD STARTING ADDRESS OF MAPPED SSF
337          ;FILE TO BASE ADDRESS STORAGE.

```



```

338 036472 005037 001514          CLR    SSFENB          :ASSUME 'SSF' BIT IS NOT TO BE WRITTEN
339                                     :IN THE 1ST HEADER WORD.
340 036476 004737 036506          JSR    PC,60$         :SEARCH SKIP SECTOR MAPPED FILE
341 036502 000137 037026          JMP    71$            :DONE WITH ALL FILE SEARCHES !!
342
343 036506 013737 001516 037060 60$:  MOV    ASNDC,76$       :LOAD COMPARING CYLINDER ADDRESS
344 036514 013737 001520 037062    MOV    ASNDA,77$       :LOAD COMPARING TRACK, SECTOR ADDRESS
345 036522 013737 037054 037056    MOV    74$,75$        :LOAD NUMBER OF SECTORS TO CONFIRM
346
347                                     :SETUP FOR A BINARY SEARCH OF THE CURRENT FILE FOR THE COMPARING
348                                     :CYLINDER, TRACK AND SECTOR ADDRESS
349
350 036530 013700 037064          61$:  MOV    78$,RO      :LOAD THE BASE ADDRESS IN RO
351 036534 022710 177777          62$:  CMP    #-1,(RO)    :IS THIS FILE TERMINATOR ?
352 036540 001462                                     BEQ    67$            :BR IF YES
353 036542 023710 037060          CMP    76$, (RO)     :IS THIS CYLINDER IN BAD TABLE ?
354 036546 001014                                     BNE    63$            :BR IF NO
355
356                                     :FILE ENTRY EQUALS COMPARING CYLINDER. SEE IF THE NEXT ENTRY EQUALS
357                                     :THE COMPARING TRACK, AND SECTOR.
358
359 036550 123760 037063 000003    CMPB   77$+1,3(RO)   :IS THIS TRACK IN BAD TABLE ?
360 036556 001010                                     BNE    63$            :BR IF NO
361
362 036560 123760 037062 000002    CMPB   77$,2(RO)    :IS THIS SECTOR IN BAD TABLE ?
363 036566 001406                                     BEQ    64$            :BR IF YES
364 036570 002403                                     BLT    63$            :BR IF SECTOR IS LESS THAN TABLE ENTRY, ELSE
365 036572 012737 177777 001514    MOV    #-1,SSFENB   :ENABLE 'SSF' BIT TO BE WRITTEN IN THE
366                                     :1ST HEADER WORD.
367
368 036600 022020          63$:  CMP    (RO)+,(RO)+ :NO, ADJUST CYLINDER POINTER IN BAD FILE
369 036602 000754          BR     62$           :AND CONTINUE SEARCH.
370
371                                     :THE COMPARING CYLINDER, TRACK AND SECTOR IS IN THE BAD SECTOR FILE.
372                                     :ADVANCE THE ASSIGNED SECTOR AND START THE SEARCH ALL OVER.
373 036604          64$:
374 036604 105237 001520          INCB   ASNDA         :INCREMENT SECTOR
375 036610 032737 001000 001444    BIT    #SSEI,RMOFO  :IS SSEI SET ?
376 036616 001405                                     BEQ    65$            :NO !!
377 036620 122737 000037 001520    CMPB   #31.,ASNDA   :SECTOR OK ?
378 036626 103327                                     BHIS   60$           :YES !!
379 036630 000404                                     BR     66$
380 036632 122737 000036 001520 65$:  CMPB   #30.,ASNDA   :SECTOR OK ??
381 036640 103322                                     BHIS   60$           :YES !!
382 036642 105037 001520          66$:  CLRB   ASNDA         :CLEAR SECTOR AND ADVANCE TRACK
383 036646 105237 001521          INCB   ASNDA+1
384 036652 123737 001335 001521    CMPB   LSTRK+1,ASNDA+1 :TRACK OK ?
385 036660 103312                                     BHIS   60$           :YES !!
386 036662 005037 001520          CLR    ASNDA         :CLEAR TRACK AND SECTOR
387 036666 005237 001516          INC    ASNDC         :INCREMENT CYLINDER
388 036672 022737 001060 001516    CMP    #560.,ASNDC  :CYLINDER OK ??
389 036700 103302                                     BHIS   60$           :YES !!
390 036702 005726          TST    (SP)+        :RESTORE STACK AND
391 036704 000450          BR     71$           :GET OUT !!
392
393                                     :THE COMPARING SECTOR IS NOT IN THE BAD SECTOR FILES. DECREMENT THE
394                                     :NUMBER OF SECTORS TO COMPARE AND SEARCH THE NEXT SECTOR IF THE NUMBER
    
```

```

395                                     :IS NOT ZERO.
396 036706                             67$:
397 036706 005337 037056                DEC 75$      :DECREMENT NUMBER OF SECTORS TO COMPARE
398 036712 001456                       BEQ 73$      :DONE IF ZERO
399
400 036714 105237 037062                INCB 77$     :INCREMENT THE COMPARING SECTOR
401 036720 032737 001000 001444        BIT #SSEI,RMOFO :IS SSEI SET ?
402 036726 001405                       BEQ 68$     :NO !!
403 036730 122737 000037 037062        CMPB #31,,77$  :SECTOR OK ?
404 036736 103032                       BHIS 70$    :YES !!
405 036740 000404                       BR 69$
406 036742 122737 000036 037062 68$:  CMPB #30,,77$  :SECTOR OK ??
407 036750 103025                       BHIS 70$    :YES !!
408 036752 105037 037062 69$:  CLRB 77$     :CLEAR SECTOR
409 036756 105237 037063                INCB 77$+1  :INCREMENT TRACK
410 036762 123737 001335 037063        CMPB LSTRK+1,77$+1 :TRACK OK ??
411 036770 103015                       BHIS 70$    :YES !!
412 036772 005037 037062                CLR 77$     :CLEAR SECTOR TRACK
413 036776 005237 037060                INC 76$     :INCREMENT CYLINDER
414 037002 022737 001060 037060        CMP #560,,76$ :CYLINDER OK ??
415 037010 103005                       BHIS 70$    :YES !!
416 037012 013737 037060 001446        MOV 76$,RMDCO :LOAD CYLINDER
417 037020 005726                       TST (SP)+   :RESTORE STACK AND
418 037022 000407                       BR 72$     :GET OUT !!
419
420 037024 000641                       70$: BR 61$ :SEARCH NEXT SECTOR
421
422                                     :ASSIGN THE SECTOR AND RETURN TO USER
423 037026                             71$:
424 037026 013737 001516 001446        MOV ASNDC,RMDCO :LOAD CYLINDER
425 037034 013737 001520 001420        MOV ASNDA,RMDAO :LOAD TRACK AND SECTOR
426 037042                             72$:
427 037042 012602                       MOV (SP)+,R2  ::POP STACK INTO R2
428 037044 012601                       MOV (SP)+,R1  ::POP STACK INTO R1
429 037046 012600                       MOV (SP)+,R0  ::POP STACK INTO R0
430
431                                     :THE FOLLOWING ARE STORAGE LOCATIONS FOR THE MODULE
432
433 037054 000000                       74$: .WORD 0 :RETRY COUNT/ NUMBER OF SECTORS REQUIRED
434 037056 000000                       75$: .WORD 0 :NUMBER OF SECTORS TO COMPARE
435 037060 000000                       76$: .WORD 0 :COMPARING CYLINDER
436 037062 000000                       77$: .WORD 0 :COMPARING TRACK AND SECTOR
437 037064 000000                       78$: .WORD 0 :BASE ADDRESS OF BAD SECTOR FILE BEING SEARCHED
438 037066 000000                       79$: .WORD 0 :CONTAINS RMER2
439
440                                     :THIS ROUTINE SEARCHES THE SKIP SECTOR FILE FOR ANY BAD SECTORS
441                                     :WHICH OCCUR ON THE FE CYLINDERS. THE BAD SECTORS ARE THEN MAPPED
442                                     :INTO A 'SSFIL' TABLE TO BE USED BY THE 'BADSC' ROUTINE.
443
444 037070                             GETSSF:
445 037070 010046                       MOV R0,-(SP)  ::PUSH R0 ON STACK
446 037072 010146                       MOV R1,-(SP)  ::PUSH R1 ON STACK
447 037074 012700 105260                 MOV #BUFTWO,R0 :R0 = POINTER TO BUFFER
448 037100 012701 110300                 MOV #SSFIL,R1 :R1 = POINTER TO SKIP SECTOR TABLE
    
```


BAD SECTOR MODULE

```

447 037104 012021      MOV      (R0)+,(R1)+      ;SAVE HEADER WORD 0(CYL ADDRESS)
448 037106 012021      MOV      (R0)+,(R1)+      ;SAVE HEADER WORD 1(TRK/SEC ADDRESS)
449
450 037110 012721 177777      1$:      MOV      #-1,(R1)+      ;INITIALIZE 'SSFIL' TABLE
451 037114 022701 110374      CMP      #SSFIL+60.,R1    ;DONE ?
452 037120 001373      BNE      1$              ;BR IF NO
453
454 037122 012701 110304      MOV      #SSFIL+4,R1     ;SETUP TO MAP SSFIL TABLE
455 037126 062700 000026      ADD      #22.,R0        ;GET OVER S/N(2), CREATE DATE, REV DATE, REV#,
456                                     ;FORMAT TYPE, UNUSED WORDS(3), ENTRY# AND
457                                     ;CHECKSUM (11. WORDS)
458 037132 021027 001057      2$:      CMP      (R0),#559.     ;IS CYLINDER IN FE AREA ?
459 037136 103410      BLO      4$              ;NO !!
460 037140 101004      BHI      3$              ;YES, ON CYLINDER 560.
461                                     ;YES, WHEN ON CYLINDER 559. DO NOT LOG
462                                     ;ANY BAD SECTORS THAT OCCUR ON THE SKIP
463                                     ;SECTOR TRACK(0) OR ALTERNATE DEC144 TRACK(1)
464 037142 126027 000003 000001      CMPB     3(R0),#1        ;IS IT ON TRACK 0 OR 1 ?
465 037150 003403      BLE      4$              ;YES !!
466 037152 011021      3$:      MOV      (R0),(R1)+      ;LOG THIS CYLINDER AND
467 037154 016021 000002      MOV      2(R0),(R1)+    ;TRACK/SECTOR IN 'SSFIL' TABLE.
468
469 037160 022020      4$:      CMP      (R0)+,(R0)+    ;ADJUST BUFFER TO NEXT CYLINDER ENTRY
470 037162 020127 110374      CMP      R1,#SSFIL+60.  ;ANY MORE ENTRIES AVAILABLE IN TABLE ?
471 037166 103003      BHIS     5$              ;NO, EXIT !!
472 037170 020027 110274      CMP      R0,#BUFTWO+1548. ;DONE WITH ALL 3 SECTORS ?
473 037174 103756      BLO      2$              ;NO, DO NEXT ENTRY
474
475 037176 012737 177777 107270      5$:      MOV      #-1,MFGFIL+516. ;RESTORE TERMINATOR TO MFGFIL
476 037204 000240      NOP
477 037206 012601      MOV      (SP)+,R1       ;;POP STACK INTO R1
478 037210 012600      MOV      (SP)+,R0       ;;POP STACK INTO R0
478 037212 000207      RTS      PC

```


.SBTTL BUFFER GENERATOR SUBROUTINE

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

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

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

GENBUF:

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

```

037214      010046
037214      010146
037220      010246
037222      010346
037224      010446
21 037226      013700      001416
22 037232      013701      001414
23 037236      013737      001446      037456
24 037244      013737      001420      037460
25
26 037252      032737      000002      001412      1$:
27 037260      001451
28 037262      013710      037456
29 037266      052710      140000
30 037272      012702      000035
31
32 037276      032737      010000      001444
33 037304      001410
34 037306      052710      010000
35 037312      005202
36
37 037314      032737      001000      001444
38 037322      001401
39 037324      005202
40
41 037326      005201      2$:
42 037330      001443
43
44 037332      005720
45 037334      013720      037460
46 037340      005201
47 037342      001436
48 037344      012703      037460
49 037350      105213
50 037352      120213
51 037354      103013
52 037356      105013
    
```

```

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 RMBAD,R0      ;;LOAD DATA BUFFER ADDRESS
MOV RMWCO,R1      ;;LOAD WORD COUNT
MOV RMDCO,7$     ;;LOAD STARTING CYLINDER ADDRESS
MOV RMDAO,8$     ;;LOAD STARTING TRACK,SECTOR ADDRESS
26: BIT #BIT1,RMCS10 ;;WRITE HEADER & DATA??
    BEQ 3$         ;;NO!!
    MOV 7$(R0);WRITE HEADER WORD #1
    BIS #MSE!USE,(R0) ;;SET BAD SECTOR FLAGS FOR GOOD SECTOR
    MOV #29.,R2    ;;R2 = MAXIMUM SECTOR ADDRESS (29.)
32: BIT #FMT16,RMOFO ;;18 BIT FORMAT??
    BEQ 2$         ;;YES !!
    BIS #FMT16,(R0) ;;SET 16 FORMAT BIT IN HEADER
    INC R2         ;;CHANGE MAXIMUM SECTOR ADDRESS (30.)
37: BIT #SSEI,RMOFO ;;IS SSEI SET TO ACCESS ALL SECTORS ?
    BEQ 2$         ;;NO !!
    INC R2         ;;CHANGE MAXIMUM SECTOR ADDRESS (31.)
41: INC R1         ;;INCREMENT WORD COUNT
    BEQ 6$         ;;EXIT IF DONE
    TST (R0)+     ;;MOVE R0 TO HEADER WORD #2
    MOV 8$(R0)+   ;;WRITE HEADER WORD #2
    INC R1        ;;INCREMENT WORD COUNT AND
    BEQ 6$        ;;EXIT IF DONE
    MOV #8$,R3    ;;ADVANCE SECTOR ADDRESS
    INCB (R3)
    CMPB R2,(R3) ;;SECTOR OVERFLOW ??
    BHIS 3$      ;;NO !!
    CLRB (R3)    ;;YES - CLEAR SECTOR ADDRESS
    
```

53	037360	105263	000001		INCB	1(R3)	:ADVANCE TRACK ADDRESS
54	037364	123763	001335	000001	CMPB	LSTRK+1,1(R3)	:TRACK OVERFLOW ??
55	037372	103004			BHIS	3\$:NO !!
56	037374	105063	000001		CLRB	1(R3)	:YES - CLEAR TRACK ADDRESS
57	037400	105237	037456		INCB	7\$:ADVANCE CYLINDER ADDRESS
58	037404	012704	000400	3\$:	MOV	#256.,R4	:LOAD SECTOR DATA COUNT
59	037410	013702	001174	4\$:	MOV	\$TMP0,R2	:LOAD PATTERN ADDRESS
60	037414	013703	001176		MOV	\$TMP1,R3	:LOAD PATTERN COUNT
61	037420	012220		5\$:	MOV	(R2)+,(R0)+	:WRITE DATA PATTERN
62	037422	005201			INC	R1	:INCREMENT WORD COUNT AND
63	037424	001405			BEQ	6\$:EXIT IF DONE
64	037426	005304			DEC	R4	:DECREMENT SECTOR COUNT
65	037430	001710			BEQ	1\$:START NEXT SECTOR IF 0
66	037432	005303			DEC	R3	:DECREMENT PATTERN COUNT
67	037434	001765			BEQ	4\$:RESTART PATTERN IF 0
68	037436	000770			BR	5\$:CONTINUE DATA PATTERN
69	037440			6\$:			
	037440	012604			MOV	(SP)+,R4	::POP STACK INTO R4
	037442	012603			MOV	(SP)+,R3	::POP STACK INTO R3
	037444	012602			MOV	(SP)+,R2	::POP STACK INTO R2
	037446	012601			MOV	(SP)+,R1	::POP STACK INTO R1
	037450	012600			MOV	(SP)+,R0	::POP STACK INTO R0
70	037452	000240			NOP		
71	037454	000207			RTS	PC	
72							
73	037456	000000		7\$:	.WORD		:CYLINDER ADDRESS STORAGE
74	037460	000000		8\$:	.WORD		:TRACK, SECTOR ADDRESS STORAGE

COMPARE BUFFER SUBROUTINE

```

1      .SBTTL COMPARE BUFFER SUBROUTINE
2
3      :THIS SUBROUTINE COMPARES THE CONTENTS OF BUFONE AND BUFTWO,
4      :ASSUMING THAT BUFONE IS THE BUFFER FROM WHICH DATA WAS WRITTEN
5      :AND BUFTWO IS THE BUFFER TO WHICH DATA WAS READ. ERRORS IN BUFFER
6      :COMPARISON ARE REPORTED TO THE USER VIA THE USER'S ERROR CALL.
7
8      :CALL:
9      : (1) JSR PC,CMPBUF
10     : (2) .WORD WRITE BUFFER ADDRESS
11     : (3) .WORD READ BUFFER ADDRESS
12     : (4) BR ?? RETURN HERE IF NO ERROR
13     : (5) NOP RETURN HERE IF ERROR
14     : (6) ERROR ERROR DEFINED BY SUBROUTINE
15
16
17     037462 CMPBUF: MOV R0,-(SP) ;;PUSH R0 ON STACK
18     037462 010046 MOV R1,-(SP) ;;PUSH R1 ON STACK
19     037464 010146 MOV R2,-(SP) ;;PUSH R2 ON STACK
20     037466 010246 MOV R3,-(SP) ;;PUSH R3 ON STACK
21     037470 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
22     037472 005037 040022 CLR 14$ ;CLEAR CORRECTION FLAG
23
24     :DETERMINE IF DATA SHOULD BE CORRECTED
25     037476 033737 004000 001370 BIT ECI,RMOFI ;WAS ECC CORRECTION ALLOWED ??
26     037504 001063 BNE 9$ ;NO !!
27     037506 032737 100000 001352 BIT #DCK,RMER1I ;WAS THERE A DATA CHECK ??
28     037514 001457 BEQ 9$ ;NO !!
29     037516 032737 000100 001352 BIT #ECH,RMER1I ;IS ERROR CORRECTION HARD SET ?
30     037524 001053 BNE 9$ ;YES !!
31     037526 032737 010000 001370 BIT #FMT16,RMOFI ;IS THIS 16 BIT FORMAT ??
32     037534 001447 BEQ 9$ ;NO !!
33
34     :CORRECT DATA USING ECC INFORMATION
35     037536 013700 001416 MOV RMBAD,R0 ;R0 = STARTING BUFFER ADDRESS
36     037542 013701 001402 MOV RMEC1I,R1 ;R1 = ECC POSITION
37     037546 052737 100000 040022 BIS #BIT15,14$ ;SET CORRECTION FLAG
38
39     :MOVE R0 TO WORD BOUNDARY OF ERROR BURST
40     037554 022701 000020 1$: CMP #16.,R1 ;IS BIT POSITION > 1 WORD
41     037560 103004 BHIS 2$ ;NO !!
42     037562 162701 000020 SUB #16.,R1 ;SUBTRACT 1 WORDS WORTH
43     037566 005720 TST (R0)+ ;ADVANCE BUFFER ADDRESS 1 WORD
44     037570 000771 BR 1$
45     037572 012702 000001 2$: MOV #1,R2 ;R2 = BIT POINTER
46     037576 010203 MOV R2,R3 ;R3 = BIT NUMBER
47
48     :MOVE R2 TO STARTING BIT OF ERROR BURST
49     037600 020301 3$: CMP R3,R1 ;IS R3 SAME AS R1 ??
50     037602 001403 BEQ 4$ ;YES !!
51     037604 006302 ASL R2 ;SHIFT BIT POINTER
52     037606 005203 INC R3 ;INCREMENT BIT NUMBER
53     037610 000773 BR 3$
54     037612 012703 000013 4$: MOV #11.,R3 ;R3 = LENGTH OF ERROR BURST
55
56     :CORRECT THE ERROR BURST
57     037616 030237 001404 5$: BIT R2,RMEC2I ;IS THIS BIT SET IN ECC PATTERN ??

```


COMPARE BUFFER SUBROUTINE

```

54 037622 001405          BEQ      7$          ;NO - DO NOT CORRECT THIS BIT
55 037624 030210          BIT      R2,(R0)     ;IS THE BIT PRESENTLY SET ??
56 037626 001402          BEQ      6$          ;NO
57 037630 040210          BIC      R2,(R0)     ;RESET THE BIT
58 037632 000401          BR       7$
59 037634 050210          6$:  BIS      R2,(R0)     ;SET THE BIT
60 037636 006302          7$:  ASL      R2          ;SHIFT TO NEXT BIT
61 037640 001003          BNE
62 037642 012702 000001  MOV      #1,R2      ;CONTINUE WITH FIRST BIT OF NEXT WORD
63 037646 005720          TST     (R0)+
64 037650 005303          8$:  DEC      R3          ;END OF BURST ??
65 037652 001361          BNE     5$          ;NO !!
66
67          ;COMPARE WRITE BUFFER TO READ BUFFER
68 037654 017600 000010  9$:  MOV      @10(SP),R0 ;R0 = WRITE BUFFER
69 037660 062766 000002 000010  ADD      #2,10(SP)   ;MOVE SP TO READ ADDRESS
70 037666 017601 000010          MOV      @10(SP),R1 ;R1 = READ BUFFER
71 037672 062766 000002 000010  ADD      #2,10(SP)   ;MOVE SP TO RETURN ADDRESS
72 037700 013702 001340          MOV      RMWCI,R2    ;R2 = NUMBER OF WORDS TRANSFER
73 037704 163702 001414          SUB      RMWCO,R2
74 037710 022021          10$:  CMP      (R0)+,(R1)+ ;COMPARE DATA WORDS
75 037712 001003          BNE     11$         ;EXIT IF NOT EQUAL
76 037714 005302          DEC      R2          ;DECREMENT WORD COUNT
77 037716 001374          BNE     10$         ;CONTINUE IF NOT DONE
78 037720 000433          BR       13$        ;DONE COMPARE - NO ERROR
79
80          ;DATA COMPARE FAILED
81 037722 014037 001140  11$:  MOV      -(R0),SGDDAT ;STORE GOOD DATA FOR TYPEOUT
82 037726 014137 001142          MOV      -(R1),SBDDAT ;STORE BAD DATA FOR TYPEOUT
83 037732 010037 001134          MOV      R0,SGDADR    ;STORE ADDRESS OF GOOD DATA
84 037736 010137 001136          MOV      R1,SBADR     ;STORE ADDRESS OF BAD DATA
85 037742 010237 001174          MOV      R2,$TMP0     ;STORE WORD COUNT OF ERROR
86 037746 062766 000004 000010  ADD      #4,10(SP)    ;MOVE SP TO USER'S ERROR CALL
87 037754 112776 000336 000010  MOVB    #336,@10(SP) ;WRITE ERROR NUMBER IN CALL
88
89          ;CHANGE ERROR NUMBER IF ECC CORRECTION FAILED
90 037762 032737 100000 040022  BIT      #BIT15,14$  ;WAS ECC CORRECTION USED ??
91 037770 001403          BEQ     12$         ;NO !!
92 037772 112776 000163 000010  MOVB    #163,@10(SP) ;ECC CORRECTION FAILED
93 040000 162766 000002 000010  12$:  SUB      #2,10(SP)   ;MOVE SP TO RETURN IF ERROR
94 040006 000240          NOP
95 040010          13$:
96 040010 012603          MOV     (SP)+,R3     ;;POP STACK INTO R3
97 040012 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
98 040014 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
99 040016 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
100 040020 000207          RTS      PC          ;RETURN TO USER
101 040022 000000          14$:  .WORD          ;ECC CORRECTION FLAG

```

GET STATUS SUBROUTINE

```

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     1$:     MOVB     R2,(R0)+ ;WRITE REGISTER INDEX IN TABLE
18     CLR      -(R1)        ;CLEAR CORRESPONDING LOCATION
19     2$:     SUB      #2,R2  ;DECREMENT TO NEXT INDEX
20     BMI      3$          ;BRANCH OUT IF DONE
21     CMP      #RMDB,R2    ;DONT WRITE RMDB INDEX
22     BNE     1$
23     CLR      -(R1)
24     BR      2$
25     3$:     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 04 IN THE TABLE WILL CAUSE THE SUBROUTINE TO
 : READ "RMA" AND STORE ITS CONTENTS AT THE LOCATION IN
 : THE BUFFER ASSIGNED TO THAT REGISTER. THE NUMBER OF
 : REGISTERS TO BE READ IS VARIABLE FROM 1 TO 22; THE INDEX
 : TABLE MUST BE TERMINATED WITH A CONTROL BYTE (200)
 : WHICH SHOULD FOLLOW THE LAST ENTRY.

:SUBROUTINE CALL:

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

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

```

GET:  NOP
      ADD #4,(SP) ;CLEAR ERROR NUMBER IN USER'S
      CLRB @ (SP) ;ERROR CALL
      SUB #4,(SP)
      MOV R0,-(SP) ;:PUSH R0 ON STACK
      MOV R1,-(SP) ;:PUSH R1 ON STACK
      MOV R2,-(SP) ;:PUSH R2 ON STACK
      MOV R3,-(SP) ;:PUSH R3 ON STACK
      MOV R4,-(SP) ;:PUSH R4 ON STACK
      MOV ERRVEC,-(SP) ;:PUSH ERRVEC ON STACK
      MOV ERRVEC+2,-(SP) ;:PUSH ERRVEC+2 ON STACK
      MOV $BASE,R0
      MOV #GETBUF,R2
      MOV #GETINX,R4
      MOV #3$,ERRVEC ;:SETUP FOR TIMEOUT
      MOV #PR6,ERRVEC+2
      1$: MOV RMCS2(R0),STMP0 ;GET 'NED' STATUS
      MOV RMCS1(R0),STMP1 ;GET 'DVA' STATUS
      BIT #DVA,STMP1 ;:DEVICE AVAILABLE??
      BNE 2$ ;:YES!!
      ADD #4,16(SP) ;:WRITE ERROR NUMBER IN USER'S
      MOVB #112,@16(SP) ;:ERROR CALL
      BR 4$
      2$: TSTB (R4) ;:DONE??
      BMI 6$ ;:YES!!
      MOVB (R4),R1 ;:R1 = REGISTER ADDRESS
      BIC #^CIDXMSK,R1 ;:CLEAR ANY SIGN EXTENSION
    
```

040110	000240		
040112	062716	000004	
040116	105076	000000	
040122	162716	000004	
040126	010046		
040130	010146		
040132	010246		
040134	010346		
040136	010446		
040140	013746	000004	
040144	013746	000006	
040150	013700	001276	
040154	012702	001336	
040160	012704	001526	
040164	012737	040272	000004
040172	012737	000300	000006
040200	016037	000010	001174
040206	016037	000000	001176
040214	032737	004000	001176
040222	001007		
040224	062766	000004	000016
040232	112776	000112	000016
040240	000423		
040242	105714		
040244	100433		
040246	111401		
040250	042701	177700	

GET SUBROUTINE

```

52 040254 060001          ADD      R0,R1
53 040256 112403          MOVb    (R4)+,R3      ;R3 = STORAGE ADDRESS FOR REGISTER
54 040260 042703 177700  BIC     #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
55 040264 060203          ADD     R2,R3
56 040266 011113          MOV     (R1),(R3)    ;READ REGISTER
57 040270 000764          BR     2$
58
59 040272 022626          3$:    CMP     (SP)+,(SP)+ ;RESTORE STACK
60 040274 062766 000004 000016  ADD     #4,16(SP)    ;WRITE ERROR NUMBER IN
61 040302 112776 000007 000016  MOVb    #7,@16(SP)  ;USER'S ERROR CALL
62 040310 162766 000002 000016  4$:    SUB     #2,16(SP)
63 040316 105714          5$:    TSTB   (R4)        ;DONE CLEARING??
64 040320 100405          BMI     6$          ;YES!!
65 040322 005003          CLR     R3          ;CLEAR REMAINING STORAGE
66 040324 112403          MOVb    (R4)+,R3    ;LOCATIONS
67 040326 060203          ADD     R2,R3
68 040330 005013          CLR     (R3)
69 040332 000771          BR     5$
70 040334          6$:
   040334 012637 000006          MOV     (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
   040340 012637 000004          MOV     (SP)+,ERRVEC  ;:POP STACK INTO ERRVEC
   040344 012604          MOV     (SP)+,R4      ;:POP STACK INTO R4
   040346 012603          MOV     (SP)+,R3      ;:POP STACK INTO R3
   040350 012602          MOV     (SP)+,R2      ;:POP STACK INTO R2
   040352 012601          MOV     (SP)+,R1      ;:POP STACK INTO R1
   040354 012600          MOV     (SP)+,R0      ;:POP STACK INTO R0
71 040356 000207          RTS     PC          ;RETURN

```

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

.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

040360 000240
040362 010046
040364 010146
040366 010246
040370 010346
040372 010446
040374 013746 000004
040400 013746 000006
040404 013700 001276
040410 012702 001412
040414 012704 001555
040420 012737 040540 000004
040426 012737 000300 000006
040434 016037 000010 001174
040442 016037 000000 001176
040450 032737 004000 001176
040456 001007
040460 062766 000004 000016
040466 112776 000112 000016
040474 000430
040476 105714
040500 100431
040502 111401
040504 042701 177700
040510 060001
040512 111403
040514 042703 177700
040520 060203
040522 011311
040524 122714 000032

PUT: NOP
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV ERRVEC,-(SP) ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
MOV \$BASE,R0
MOV #PUTBUF,R2
MOV #PUTINX,R4
MOV #4\$,ERRVEC ;SETUP FOR TIMEOUT
MOV #PR6,ERRVEC+2
1\$: MOV RMCS2(R0),\$TMP0 ;GET 'NED' STATUS
MOV RMCS1(R0),\$TMP1 ;GET 'DVA' STATUS
BIT #DVA,\$TMP1 ;DEVICE AVAILABLE??
BNE 2\$;YES!!
ADD #4,16(SP) ;WRITE ERROR NUMBER IN
MOVB #112,@16(SP) ;USER'S ERROR CALL
BR 5\$
2\$: TSTB (R4) ;DONE??
BMI 6\$;YES!!
MOVB (R4),R1 ;R1 = REGISTER ADDRESS
BIC #^CIDXMSK,R1 ;CLEAR ANY SIGN EXTENSION
ADD R0,R1
MOVB (R4),R3 ;R3 = STORAGE ADDRESS
BIC #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
ADD R2,R3
MOV (R3),(R1) ;WRITE REGISTER
CMPB #RMOF,(R4) ;WAS RMOF JUST LOADED ?

PUT SUBROUTINE

```

52 040530 001001      BNE      3$
53 040532 011311      MOV      (R3), (R1)      ;BR IF NO
54                                     ;WRITE RMOF AGAIN, TO ENSURE THAT 16 BIT MODE
55 040534 105724      3$:    TSTB      (R4)+      ;IS SET BEFORE "SSEI" BIT IS SET.
56 040536 000757      BR       2$           ;ADJUST REGISTER POINTER
57
58 040540 022626      4$:    CMP      (SP)+, (SP)+      ;ADJUST STACK
59 040542 062766 000004 000016  ADD      #4, 16(SP)      ;WRITE ERROR NUMBER IN
60 040550 112776 000007 000016  MOVB     #7, @16(SP)     ;USER'S ERROR CALL
61 040556 162766 000002 000016  5$:    SUB      #2, 16(SP)
62
63 040564 012637 000006      6$:    MOV      (SP)+, ERRVEC+2      ;:POP STACK INTO ERRVEC+2
64 040564 012637 000004      MOV      (SP)+, ERRVEC      ;:POP STACK INTO ERRVEC
65 040574 012604      MOV      (SP)+, R4      ;:POP STACK INTO R4
66 040576 012603      MOV      (SP)+, R3      ;:POP STACK INTO R3
67 040600 012602      MOV      (SP)+, R2      ;:POP STACK INTO R2
68 040602 012601      MOV      (SP)+, R1      ;:POP STACK INTO R1
69 040604 012600      MOV      (SP)+, R0      ;:POP STACK INTO R0
70 040606 000207      RTS      PC           ;RETURN

```



```

1          .SETTL  SIZE CLOCK SUBROUTINE
2
3 040610    013746  000004    SIZCLK:
040610    013746  000006    MOV     ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
040614    013746  000006    MOV     ERRVEC+2,-(SP)  ;;PUSH ERRVEC+2 ON STACK
4 040620    012737  040656  000004    MOV     #1$,ERRVEC      ;SET UP FOR BUS TIMEOUT
5 040626    012737  000300  000006    MOV     #PR6,ERRVEC+2
6 040634    012737  177546  001522    MOV     #177546,CLKADR  ;LOAD ADDRESSES FOR KW11-L
7 040642    012737  000100  001524    MOV     #100,CLKVCT
8 040650    005777  140646    TST     @CLKADR         ;TEST FOR KW11-L PRESENT
9 040654    000421    BR      3$             ;YES - KW11-L IS PRESENT
10 040656   022626    1$:  CMP     (SP)+,(SP)+   ;RESTORE SP
11 040660   012737  040710  000004    MOV     #2$,ERRVEC      ;SET UP FOR BUS TIMEOUT
12 040666   012737  172540  001522    MOV     #172540,CLKADR  ;LOAD ADDRESSES FOR KW11-P CLOCK
13 040674   012737  000104  001524    MOV     #104,CLKVCT
14 040702   005777  140614    TST     @CLKADR         ;TEST FOR KW11-P PRESENT
15 040706   000404    BR      3$             ;YES - KW11-P IS PRESENT
16 040710   022626    2$:  CMP     (SP)+,(SP)+   ;RESTORE SP
17 040712   062766  000002  000004    ADD     #2,4(SP)        ;MOVE RETURN TO ERROR
18 040720    040720   012637  000006    MOV     (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
    040720   012637  000004    MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
19 040730   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     040732 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
12     040734 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
13     040736 013746 000004  MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
14     040742 013746 000006  MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
15     040746 012737 041032 000004  MOV      #3$,ERRVEC    ;SETUP FOR BUS TIMEOUT - 04 TRAP
16     040754 012737 000300 000006  MOV      #PR6,ERRVEC+2
17     040762 013700 001276      MOV      $BASE,R0      ;R0=BASE ADDRESS
18     040766 012702 000036      MOV      #30.,R2      ;R2=NUMBER OF CLOCK CYCLES
19     040772 004737 041074 1$:      JSR      PC,STIMER    ;START CLOCK TIMER
20
21     2$:      MOV      RMCS1(R0),-(SP) ;GET STATUS
22     040776 016046 000000      BIC      #^C<RDY!GO>,(SP)
23     041002 042716 177576      CMP      #RDY,(SP)+    ;RDY=1,GO=0??
24     041006 022726 000200      BEQ      4$           ;YES!!
25     041012 001421      BIT      #BIT7,@CLKADR ;TIMER DONE??
26     041014 032777 000200 140500  BEQ      2$           ;NO!!
27     041022 001765      DEC      R2           ;DEC NUMBER OF CYCLES
28     041024 005302      BNE      1$           ;CONTINUE IF NOT DONE
29     041026 001361      BR       4$           ;'RDY' DID NOT SET OR 'GO' DID NOT RESET
30     041030 000412      ;WITHIN 500 MSEC AFTER THE COMMAND WAS ISSUED.
31     3$:      CMP      (SP)+,(SP)+    ;ADJUST STACK
32     041032 022626 000004 000012  ADD      #4,12(SP)    ;MOVE SP TO USER'S CALL
33     041034 062766 000007 000012  MOV      #7,@12(SP)  ;WRITE ERROR NUMBER
34     041042 112776 000002 000012  SUB      #2,12(SP)
35
36     4$:      MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
37     041056 012637 000006      MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
38     041056 012637 000004      MOV      (SP)+,R2      ;;POP STACK INTO R2
39     041062 012637 000004      MOV      (SP)+,R0      ;;POP STACK INTO R0
40     041066 012602      RTS      PC           ;RETURN TO USER
41     041070 012600
42     041072 000207
43
44     ;THIS ROUTINE IS USED START THE KW11-L OR THE KW11-P CLOCK. THE CLOCK
45     ;IS STARTED TO RUN IN FLAG MODE. A CLOCK WILL BE 16.667MS FOR A 60 HZ CPU
46     ;AND 20MS FOR A 50 HZ CPU.
47
48     STIMER:
49     041074 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
50     041074 010146      MOV      CLKADR,R1    ;R1=CLOCK ADDRESS
51     041076 013701 001522      CMP      R1,#172540   ;KW11-P CLOCK??
52     041102 020127 172540      BNE      1$           ;NO!!
53     041106 001003      MOV      #1,2(R1)    ;SET COUNTER
54     041110 012761 000001 000002 1$:      MOV      #BIT2!BIT0,(R1) ;START COUNTER
55     041116 012711 000005      MOV      (SP)+,R1    ;;POP STACK INTO R1
56     041122 012601      RTS      PC           ;RETURN
57     041124 000207
    
```

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

041126

```

041126 062716 000004
041132 105076 000000
041136 162716 000004
041142 013737 001346 001142
041150 042737 177770 001142
041156 013737 001234 001140
041164 042737 177770 001140
041172 123737 001140 001142
041200 001415
041202 062716 000004
041206 112776 000001 000000
041214 162716 000902
041220 004736
    
```



```

58 041222 162716 000010          SUB      #10,(SP)          ;RESTORE (SP)
59 041226 000240          NOP
60 041230 000137 041750          JMP      10$              ;SKIP OTHER CHECKS
61 041234
62
63                               ;REPORT AN ERROR IF THE DEVICE IS NOT AVAILABLE OR IF
64                               ;THE DEVICE IS NONEXISTANT
65 041234 032737 004000 001336    BIT      #DVA,RMCS1I      ;DEVICE AVAILABLE??
66 041242 001045          BNE      3$              ;YES!!
67 041244 013737 001336 001140    MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
68 041252 052737 004000 001140    BIS      #DVA,$GDDAT
69 041260 013737 001336 001142    MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
70 041266 062716 000004          ADD      #4,(SP)
71 041272 112776 000002 000000    MOVB     #2,@(SP)         ;ERROR #2
72 041300 032737 010000 001346    BIT      #NED,RMCS2I      ;WAS NED SET??
73 041306 001414          BEQ      2$              ;NO!!
74 041310 013737 001346 001140    MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
75 041316 013737 001346 001142    MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
76 041324 042737 010000 001140    BIC      #NED,$GDDAT
77 041332 112776 000003 000000    MOVB     #3,@(SP)         ;YES - CHANGE ERROR NUMBER
78 041340 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
79 041344 004736          JSR      PC,@(SP)+        ;REPORT DEVICE NOT AVAILABLE
80 041346 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
81 041352 000240          NOP
82 041354 000575          BR       10$              ;SKIP OTHER CHECKS
83 041356
84
85                               ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
86 041356 032737 000200 001336    BIT      #RDY,RMCS1I      ;CONTROLLER READY??
87 041364 001030          BNE      4$              ;YES!!
88 041366 013737 001336 001140    MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
89 041374 052737 000200 001140    BIS      #RDY,$GDDAT
90 041402 042737 160001 001140    BIC      #SC!TR!MCPE!GO,$GDDAT
91 041410 013737 001336 001142    MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
92 041416 062716 000004          ADD      #4,(SP)
93 041422 112776 000004 000000    MOVB     #4,@(SP)         ;ERROR #4
94 041430 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
95 041434 004736          JSR      PC,@(SP)+        ;REPORT CONTROLLER NOT READY
96 041436 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
97 041442 000240          NOP
98 041444 000541          BR       10$              ;SKIP OTHER CHECKS
99 041446
100
101                               ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
102 041446 032737 000001 001336    BIT      #GO,RMCS1I       ;GO RESET??
103 041454 001431          BEQ      5$              ;YES!!
104 041456 032737 000200 001350    BIT      #DRY,RMDSI       ;DRIVE READY??
105 041464 001025          BNE      5$              ;YES!!
106 041466 013737 001336 001140    MOV      RMCS1I,$GDDAT    ;EXPECTED STATUS
107 041474 042737 160001 001140    BIC      #SC!TR!MCPE!GO,$GDDAT
108 041502 013737 001336 001142    MOV      RMCS1I,$BDDAT    ;RECEIVED STATUS
109 041510 062716 000004          ADD      #4,(SP)
110 041514 112776 000005 000000    MOVB     #5,@(SP)         ;ERROR #5
111 041522 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
112 041526 004736          JSR      PC,@(SP)+        ;REPORT DRIVE NOT READY
113 041530 162716 000010          SUB      #10,(SP)        ;RESTORE (SP)
114 041534 000240          NOP
    
```

```

115 041536 000504          BR      10$
116 041540          5$:
117
118          :REPORT AN ERROR IF THE RH CONTROLLER DETECTED BAD
119          :PARITY ON THE MASSBUS CONTROL BUS
120 041540 032737 020000 001336  BIT      #MCPE,RMCS1I  :PARITY ERROR ??
121 041546 001425          BEQ      6$          :NO!!
122 041550 013737 001336 001140  MOV      RMCS1I,$GDDAT :EXPECTED STATUS
123 041556 042737 160001 001140  BIC      #SC!TRE!MCPE!GO,$GDDAT
124 041564 013737 001336 001142  MOV      RMCS1I,$BDDAT :RECEIVED STATUS
125 041572 062716 000004          ADD      #4,(SP)      :MOVE STACK TO USER'S ERROR
126 041576 112776 000013 000000  MOVB    #13,@(SP)    :ERROR #13
127 041604 162716 000002          SUB      #2,(SP)      :MOVE SP TO RETURN FOR ERROR
128 041610 004736          JSR     PC,@(SP)+    :REPORT ERROR VIA USER
129 041612 162716 000010          SUB      #10,(SP)    :RESTORE STACK
130 041616 000240          NOP
131 041620 000453          BR      10$
132 041622          6$:
133
134          :REPORT AN ERROR IF THE RM80 DETECTED A CONTROL BUS PARITY ERROR
135 041622 032737 000010 001352  BIT      #PAR,RMER1I  :WAS THERE A PARITY ERROR??
136 041630 001451          BEQ      11$         :NO!!
137 041632 032737 000010 001400  BIT      #DPE,RMER2I  :WAS IT THE CONTROL BUS??
138 041640 001045          BNE     11$         :NOT SURE!!
139 041642 032737 000010 001426  BIT      #PAR,RMER10  :DID TEST SET PAR ??
140 041650 001413          BEQ      9$          :NO!!
141 041652 010046          MOV     R0,-(SP)     :PUSH R0 ON STACK
142 041654 012700 001555          MOV     #PUTINX,R0   :R0 POINTS TO INDEX TABLE
143 041660 122710 000014          7$: CMPB    #RMER1,(R0)  :SEARCH TABLE FOR RMER1
144 041664 001002          BNE     8$          :
145 041666 012600          MOV     (SP)+,R0    :POP STACK INTO R0
146 041670 000431          BR     11$         :PAR WAS SET BY TEST
147 041672 105720          8$: TSTB   (R0)+        :END OF TABLE??
148 041674 100371          BPL    7$          :NO!!
149 041676 012600          MOV     (SP)+,R0    :POP STACK INTO R0
150 041700 013737 001352 001140  9$: MOV     RMER1I,$GDDAT :EXPECTED STATUS
151 041706 042737 000010 001140  BIC     #PAR,$GDDAT
152 041714 013737 001352 001142  MOV     RMER1I,$BDDAT :RECEIVED STATUS
153 041722 062716 000004          ADD     #4,(SP)     :MOVE SP TO USER'S ERROR CALL
154 041726 112776 000050 000000  MOVB   #50,@(SP)    :WRITE THE ERROR NUMBER
155 041734 162716 000002          SUB     #2,(SP)     :MOVE SP TO RETURN FOR ERROR
156 041740 004736          JSR    PC,@(SP)+   :REPORT THE ERROR
157 041742 162716 000010          SUB     #10,(SP)    :MOVE SP TO NO ERROR RETURN
158 041746 000240          NOP
159 041750 062716 000010          10$: ADD     #10,(SP)    :RETURN TO ERROR
160 041754 000240          11$: NOP
161 041756 000207          RTS     PC
    
```


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

.SBTTL SECONDARY ERROR CHECK SUBROUTINE

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

```

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

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

SECERR:

```

:STORE FUNCTION CODE AND CLEAR USER'S ERROR NUMBER
MOV    RMCS10,81$      :STORE FUNCTION CODE
BIC    #^C<F0!F1!F2!F3!F4>,81$
ADD    #4,(SP)        :MOVE (SP) TO ERROR CALL
CLRB   @ (SP)         :CLEAR ERROR NUMBER
SUB    #4,(SP)        :MOVE (SP) TO NO ERROR RETURN
    
```

```

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

```

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

```

24 041760
27 041760 013737 001412 046120
28 041766 042737 177701 046120
29 041774 062716 000004
30 042000 105076 000000
31 042004 162716 000004
34 042010 032737 000200 001350
35 042016 001024
36 042020 013737 001350 001142
37 042026 042737 177577 001142
38 042034 012737 000200 001140
39 042042 062716 000004
40 042046 112776 000010 000000
41 042054 162716 000002
42 042060 004736
43 042062 162716 000010
44 042066 000240
47 042070 032737 000001 001336
48 042076 001423
49 042100 013737 001336 001142
50 042106 042737 177776 001142
51 042114 005037 001140
52 042120 062716 000004
53 042124 112776 000011 000000
54 042132 162716 000002
55 042136 004736
56 042140 162716 000010
57 042144 000240
    
```



```

58
59
60 042146 013737 001336 001142 :REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
61 042154 042737 177701 001142 2$: MOV RMCS1I,$BDDAT :IS FUNCTION CODE CORRECT??
62 042162 013737 046120 001140 BIC #^C76,$BDDAT
63 042170 023737 001142 001140 MOV 81,$GDDAT :EXPECTED FUNCTION CODE
64 042176 001413 BEQ 3$ :YES!!
65 042200 062716 000004 ADD #4,(SP)
66 042204 112776 000012 000000 MOV#B #12,@(SP) :ERROR NUMBER
67 042212 162716 000002 SUB #2,(SP) :MOVE SP TO RETURN FOR ERROR
68 042216 004736 JSR PC,@(SP)+ :REPORT WRONG FUNCTION CODE
69 042220 162716 000010 SUB #10,(SP) :RESTORE (SP)
70 042224 000240 NOP
71 042226
72
73
74
75 042226 005037 001140 :REPORT AN ERROR IF COMPOSITE ERROR IS SET AND NO OTHER
76 042232 005737 001352 :ERRORS ARE SET, OR IF COMPOSITE ERROR IS NOT SET AND
77 042236 001003 :OTHER ERRORS ARE SET
78 042240 005737 001400 CLR $GDDAT :EXPECT 'ERR' = 0
79 042244 001403 TST RMER1I :IS RMER1 = 0??
80 042246 052737 040000 001140 4$: BNE 4$ :NO!!
81 042254 013737 001350 001142 5$: TST RMER2I :IS RMERZ = 0??
82 042262 042737 137777 001142 BEQ 5$ :YES!!
83 042270 023737 001140 001142 4$: BIS #ERR,$GDDAT :'ERR' SOULD BE SET
84 042276 001412 5$: MOV RMDSI,$BDDAT
85 042300 062716 000004 000000 BIC #^CERR,$BDDAT
86 042304 112776 000047 000000 CMP $GDDAT,$BDDAT :IS 'ERR' OK??
87 042312 162716 000002 BEQ 6$ :YES!!
88 042316 004736 ADD #4,(SP) :MOVE SP TO USER'S ERROR
89 042320 162716 000010 MOV#B #47,@(SP) :WRITE ERROR NUMBER
90
91 :REPORT AN ERROR IF 'TRE' IS SET AND NONE OF THE BITS WHICH SET
92 :TRE IS SET, OR IF TRE IS NOT SET AND ONE OR MORE BITS WHICH
93 :SET TRE IS SET
94 042324 005037 001140 6$: CLR $GDDAT :EXPECT 'TRE' = 0
95 042330 013746 001346 MOV RMCS2I,-(SP) :WAS DLT, WCE, UPE, NED, NEM
96 042334 042726 000377 BIC #377,(SP)+ :PGE, MXF OR MDPE SET
97 042340 001010 BNE 7$ :YES!!
98 042342 032737 040000 001350 BIT #ERR,RMDSI :WAS EXCEPTION RECEIVED??
99 042350 001407 BEQ 8$ :NO!!
100 042352 022737 000030 046120 CMP #SEARCH,81$ :WAS DATA TRANSFERRED??
101 042360 103003 BHIS 8$ :NO!!
102 042362 052737 040000 001140 7$: BIS #TRE,$GDDAT :'TRE' SHOULD BE SET
103 042370 013737 001336 001142 8$: MOV RMCS1I,$BDDAT :BAD DATA FOR TYPEOUT
104 042376 042737 137777 001142 BIC #^CTRE,$BDDAT
105 042404 023737 001140 001142 CMP $GDDAT,$BDDAT :IS 'TRE' OK??
106 042412 001413 BEQ 9$ :YES!!
107 042414 062716 000004 000000 ADD #4,(SP) :MOVE SP TO USER'S ERROR CALL
108 042420 112776 000014 000000 MOV#B #14,@(SP) :WRITE ERROR NUMBER
109 042426 162716 000002 SUB #2,(SP) :MOVE SP TO RETURN FOR ERROR
110 042432 004736 JSR PC,@(SP)+ :REPORT TRE ERROR
111 042434 162716 000010 SUB #10,(SP) :RESTORE (SP)
112 042440 000240 NOP
113 042442
114
9$:
    
```

```

115
116
117
118
119
120
121
122
123
124 042442 010046
125 042444 013700 046120
126 042450 016037 067244 046112
127 042456 012600
128
129
130
131 042460 013737 046112 001140
132 042466 032737 040000 001350
133 042471 001403
134 042476 052737 100000 001140
135 042504 042737 077777 001140
136 042512 013737 001350 001142
137 042520 042737 077777 001142
138 042526 023737 001140 001142
139 042534 001413
140 042536 062716 000004
141 042542 112776 000006 000000
142 042550 162716 000002
143 042554 004736
144 042556 162716 000010
145 042562 000240
146 042564
147
148
149
150 042564 013737 046112 001140
151 042572 042737 177776 001140
152 042600 013737 001352 001142
153 042606 042737 177776 001142
154 042614 023737 001140 001142
155 042622 001412
156 042624 062716 000004
157 042630 112776 000254 000000
158 042636 162716 000002
159 042642 004736
160 042644 162716 000010
161 042650 005037 001140
162
163
164 042654 013746 046112
165 042660 052716 137777
166 042664 013737 001346 001142
167 042672 042637 001142
168 042676 001412
169 042700 062716 000004
170 042704 112776 000026 000000
171 042712 162716 000002
    
```

 :USING THE FUNCTION CODE TABLE, CHECK FOR THE FOLLOWING ERRORS:
 : .STATUS BITS NOT SET THAT SHOULD BE SET, E.G., ATA AND ILF
 : .STATUS BITS SET THAT SHOULD NOT BE SET, E.G., WCE AND ECH
 :NOTE THAT SOME ERROR BITS ARE CONDITIONAL ON THE COMMAND AND OTHER
 :STATUS CONDITIONS, E.G., WRITE LOCK ERROR SHOULD ONLY BE SET IF
 :WRITE LOCK IS ON AND THE COMMAND IS A WRITE.
 :GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE
 :MOV R0,-(SP) :PUSH R0 ON STACK
 :MOV 81\$,R0 :GET FUNCTION CODE
 :MOV FNCDTB(R0),78\$:STORE ENTRY
 :MOV (SP)+,R0 :POP STACK INTO R0
 :REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
 :ATA IS NOT SET AND SHOULD BE SET.
 :MOV 78\$,SGDDAT :GET EXPECTED ATA STATUS
 :BIT #ERR,RMSI :IS COMPOSITE ERROR SET ??
 :BEQ 10\$:NO !!
 :BIS #ATA,SGDDAT :EXPECT AN ATTENTION
 :BIC #^ATA,SGDDAT :STRIP DONT CARES
 :MOV RMSI,\$BDDAT :GET RECEIVED ATA
 :BIC #^ATA,\$BDDAT :STRIP DONT CARES
 :CMP SGDDAT,\$BDDAT :IS ATA OK ??
 :BEQ 11\$:YES !!
 :ADD #4,(SP) :MOVE SP TO USERS ERROR CALL
 :MOVB #6,@(SP) :LOAD ERROR # IN CALL
 :SUB #2,(SP) :MOVE SP TO ERROR RETURN
 :JSR PC,@(SP)+ :REPORT ERROR
 :SUB #10,(SP) :RESTORE SP
 10\$:
 11\$:
 :REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
 :WITH FUNCTION CODE TABLE
 :MOV 78\$,SGDDAT :GET EXPECTED ILF
 :BIC #^CILF,SGDDAT :CLEAR ALL OTHER BITS
 :MOV RMER1I,\$BDDAT :GET RECEIVED ILF
 :BIC #^CILF,\$BDDAT :CLEAR ALL OTHER BITS
 :CMP SGDDAT,\$BDDAT :IS ILF OK ??
 :BEQ 12\$:YES !!
 :ADD #4,(SP) :MOVE SP TO USERS ERROR CALL
 :MOVB #254,@(SP) :WRITE ERROR NUMBER IN CALL
 :SUB #2,(SP) :MOVE SP TO ERROR RETURN
 :JSR PC,@(SP)+ :REPORT ERROR AND RETURN
 :SUB #10,(SP) :MOVE SP TO NO ERROR
 12\$: CLR \$GDDAT :CLEAR EXPECTED STATUS
 :REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
 :MOV 78\$,-(SP) :GET WCE STATUS ENABLE
 :BIS #^CWCE,(SP) :SET ALL OTHER BITS
 :MOV RMCS2I,\$BDDAT :RECEIVED STATUS
 :BIC (SP)+,\$BDDAT :CLEAR WCE IF ENABLED
 :BEQ 13\$:BRANCH IF WCE OK
 :ADD #4,(SP) :MOVE SP TO USER'S ERROR CALL
 :MOVB #26,@(SP) :WRITE ERROR NUMBER
 :SUB #2,(SP) :MOVE SP TO ERROR RETURN


```

172 042716 004736          JSR    PC,@(SP)+      ;REPORT ERROR
173 042720 162716 000010  SUB    #10,(SP)      ;RESTORE ERROR
174 042724          13$:
175
176          ;REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
177 042724 013746 046112  MOV    78$,-(SP)    ;GET OPI STATUS ENABLE
178 042730 052716 157777  BIS    #^COPI,(SP) ;SET ALL OTHER BITS
179 042734 013737 001352 001142 MOV    RMER1I,$BDDAT ;GET RECEIVED STATUS
180 042742 042637 001142  BIC    (SP)+,$BDDAT ;CLEAR OPI IF ENABLED
181 042746 001412          BEQ    14$          ;BRANCH IF OPI OK
182 042750 062716 000004  ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
183 042754 112776 000164 000000 MOV    #164,@(SP)   ;WRITE ERROR NUMBER IN CALL
184 042762 162716 000002  SUB    #2,(SP)      ;MOVE SP TO ERROR RETURN
185 042766 004736          JSR    PC,@(SP)+      ;REPORT ERROR
186 042770 162716 000010  SUB    #10,(SP)      ;RESTORE SP
187 042774          14$:
188
189          ;REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
190          ;SET AND VV IS NOT RESET
191 042774 013746 046112  MOV    78$,-(SP)    ;GET IVC STATUS ENABLE
192 043000 032737 000100 001350 BIT    #VV,RMDSI    ;IS VV SET
193 043006 001402          BEQ    15$          ;NO !!
194 043010 042716 010000  BIC    #IVC,(SP)    ;YES - IVC SHOULD BE 0
195 043014 052716 167777 001142 15$: BIS    #^CIVC,(SP) ;SET ALL OTHER BITS
196 043020 013737 001400  MOV    RMER2I,$BDDAT ;GET RECEIVED STATUS
197 043026 042637 001142  BIC    (SP)+,$BDDAT ;CLEAR IVC IF ENABLED
198 043032 001412          BEQ    16$          ;BRANCH IF IVC OK
199 043034 062716 000004  ADD    #4,(SP)      ;MOVE SP TO USERS ERROR CALL
200 043040 112776 000165 000000 MOV    #165,@(SP)   ;WRITE ERROR NUMBER IN CALL
201 043046 162716 000002  SUB    #2,(SP)      ;MOVE SP TO ERROR RETURN
202 043052 004736          JSR    PC,@(SP)+      ;REPORT ERROR
203 043054 162716 000010  SUB    #10,(SP)      ;RESTORE SP TO NO ERROR
204 043060          16$:
205
206          ;BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
207          ; ALL WRITE ERRORS, I.E.,
208          ; RMER1 - WLE, WCF
209          ; RMER2 - DPE
210          ; RMCS2 - UPE.
211          ;EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
212          ;WRITE ERROR ENABLE BIT IS RESET.
213
214          ;REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
215          ;THE DRIVE IS NOT WRITE PROTECTED
216 043060 012746 177777  MOV    #-1,-(SP)    ;ASSUME WRITE ERRORS ENABLED
217 043064 032737 004000 046112 BIT    #WLE,78$    ;ARE WRITE ERRORS ENABLED ??
218 043072 001404          BEQ    17$          ;NO !!
219 043074 032737 004000 001350 BIT    #WRL,RMDSI    ;IS THE DRIVE WRITE PROTECTED ??
220 043102 001002          BNE    18$          ;YES !!
221 043104 042716 004000 17$: BIC    #WLE,(SP)    ;RESET WLE ENABLE
222 043110 013737 001352 001142 18$: MOV    RMER1I,$BDDAT ;GET RECEIVED STATUS
223 043116 042637 001142  BIC    (SP)+,$BDDAT ;CLEAR WLE IF ENABLED
224 043122 001412          BEQ    19$          ;BRANCH IF WLE OK
225 043124 062716 000004  ADD    #4,(SP)      ;MOVE SP TO USERS ERROR CALL
226 043130 112776 000023 000000 MOV    #23,@(SP)    ;WRITE ERROR NUMBER IN CALL
227 043136 162716 000002  SUB    #2,(SP)      ;MOVE SP TO ERROR RETURN
228 043142 004736          JSR    PC,@(SP)+      ;REPORT ERROR AND RETURN
    
```



```

229 043144 162716 000010          SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
230 043150          19$:
231
232          ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
233 043150 012746 177777          MOV      #-1,-(SP)          ;ASSUME WRITE ERRORS ENABLED
234 043154 032737 004000 046112          BIT      #WLE,78$          ;ARE WRITE ERRORS ENABLED ??
235 043162 001002          BNE      20$              ;YES !!
236 043164 042716 000040          BIC      #WCF,(SP)          ;DISABLE WCF ERROR
237 043170 013737 001352 001142 20$:          MOV      RMER1I,$BDDAT      ;GET RECEIVED STATUS
238 043176 042637 001142          BIC      (SP)+,$BDDAT      ;RESET WCF IF ENABLED
239 043202 001412          BEQ      21$              ;BRANCH IF WCF OK
240 043204 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
241 043210 112776 000025 000000          MOVSB   #25,a(SP)          ;WRITE ERROR NUMBER IN CALL
242 043216 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
243 043222 004736          JSR      PC,a(SP)+         ;REPORT ERROR
244 043224 162716 000010          SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
245 043230          21$:
246
247          ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
248 043230 012746 177777          MOV      #-1,-(SP)          ;ASSUME WRITE ERRORS ARE ENABLED
249 043234 032737 004000 046112          BIT      #WLE,78$          ;ARE WRITE ERRORS ENABLED ??
250 043242 001002          BNE      22$              ;YES !!
251 043244 042716 000010          BIC      #DPE,(SP)          ;RESET DPE ENABLE
252 043250 013737 001400 001142 22$:          MOV      RMER2I,$BDDAT      ;GET RECEIVED STATUS
253 043256 042637 001142          BIC      (SP)+,$BDDAT      ;RESET DPE IF ENABLED
254 043262 001412          BEQ      23$              ;BRANCH IF DPE OK
255 043264 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
256 043270 112776 000040 000000          MOVSB   #40,a(SP)         ;WRITE ERROR NUMBER IN CALL
257 043276 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
258 043302 004736          JSR      PC,a(SP)+         ;REPORT ERROR
259 043304 162716 000010          SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
260 043310          23$:
261
262          ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
263 043310 012746 177777          MOV      #-1,-(SP)          ;ASSUME WRITE ERRORS ARE ENABLED
264 043314 032737 004000 046112          BIT      #WLE,78$          ;ARE WRITE ERRORS ENABLED ??
265 043322 001002          BNE      24$              ;YES !!
266 043324 042716 020000          BIC      #UPE,(SP)          ;DISABLE UPE ERROR
267 043330 013737 001346 001142 24$:          MOV      RMCS2I,$BDDAT      ;GET RECEIVED STATUS
268 043336 042637 001142          BIC      (SP)+,$BDDAT      ;RESET UPE IF ENABLED
269 043342 001412          BEQ      25$              ;BRANCH IF UPE OK
270 043344 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
271 043350 112776 000024 000000          MOVSB   #24,a(SP)         ;WRITE ERROR NUMBER IN CALL
272 043356 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
273 043362 004736          JSR      PC,a(SP)+         ;REPORT ERROR AND RETURN
274 043364 162716 000010          SUB      #10,(SP)         ;MOVE SP TO NO ERROR
275 043370          25$:
276
277          ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
278 043370 013746 046112          MOV      78$,-(SP)          ;GET IAE ENABLE
279 043374 052716 175777          BIS      #CIAE,(SP)        ;SET ALL OTHER BITS
280 043400 013737 001352 001142          MOV      RMER1I,$BDDAT      ;GET RECEIVED STATUS
281 043406 042637 001142          BIC      (SP)+,$BDDAT      ;CLEAR IAE IF ENABLED
282 043412 001412          BEQ      26$              ;BRANCH IF IAE IS OK
283 043414 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
284 043420 112776 000166 000000          MOVSB   #166,a(SP)        ;WRITE ERROR NUMBER
285 043426 162716 000002          SUB      #2,(SP)          ;MOVE SP TO ERROR RETURN
    
```

```

286 043432 004736          JSR      PC,@(SP)+      :REPORT ERROR AND RETURN
287 043434 162716 000010  SUB      #10,(SP)      :MOVE SP TO NO ERROR
288 043440          26$:
289
290          ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
291          ; ALL READ/WRITE ERRORS, I.E.,
292          ;
293          ; RMCS1 - TRE
294          ; RMCS2 - DLT,NEM,MXF
295          ; RMDS  - LBT
296          ; RMER1 - AOE
297          ;NOTE:
298          ; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
299          ; CYLINDER REGISTER IS WRITTEN
300          ;NOTE:
301          ; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
302          ;NOTE:
303          ; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
304
305          ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
306 043440 012746 177777      MOV      #-1,-(SP)      :ASSUME ERRORS ARE ENABLED
307 043444 032737 001000 046112 BIT      #AOE,78$      :ARE ERRORS ENABLED ??
308 043452 001002          BNE      27$          :YES !!
309 043454 042716 100000      BIC      #DLT,(SP)      :RESET DLT ENABLE
310 043460 013737 001346 001142 27$: MOV      RMCS2I,$BDDAT :GET RECEIVED STATUS
311 043466 042637 001142      BIC      (SP)+,$BDDAT  :CLEAR DLT IF ENABLED
312 043472 001412          BEQ      28$          :BRANCH IF DLT IS OK
313 043474 062716 000004      ADD      #4,(SP)       :MOVE SP TO USERS ERROR CALL
314 043500 112776 000032 000000 MOVB     #32,@(SP)      :WRITE ERROR NUMBER IN CALL
315 043506 162716 000002      SUB      #2,(SP)       :MOVE SP TO ERROR RETURN
316 043512 004736          JSR      PC,@(SP)+      :REPORT ERROR AND RETURN
317 043514 162716 000010  SUB      #10,(SP)      :MOVE SP TO NO ERROR
318 043520          28$:
319
320          ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
321 043520 012746 177777      MOV      #-1,-(SP)      :ASSUME ERRORS ARE ENABLED
322 043524 032737 001000 046112 BIT      #AOE,78$      :ARE ERRORS ENABLED ??
323 043532 001002          BNE      29$          :YES !!
324 043534 042716 004000      BIC      #NEM,(SP)     :DISABLE NEM
325 043540 013737 001346 001142 29$: MOV      RMCS2I,$BDDAT :GET RECEIVED STATUS
326 043546 042637 001142      BIC      (SP)+,$BDDAT  :CLEAR NEM IF ENABLED
327 043552 001412          BEQ      30$          :BRANCH IF NEM IS OK
328 043554 062716 000004      ADD      #4,(SP)       :MOVE SP TO USERS ERROR CALL
329 043560 112776 000167 000000 MOVB     #167,@(SP)     :WRITE ERROR NUMBER IN CALL
330 043566 162716 000002      SUB      #2,(SP)       :MOVE SP TO ERROR RETURN
331 043572 004736          JSR      PC,@(SP)+      :REPORT ERROR AND RETURN
332 043574 162716 000010  SUB      #10,(SP)      :MOVE SP TO NO ERROR
333 043600          30$:
334
335          ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
336 043600 012746 177777      MOV      #-1,-(SP)      :ASSUME ERRORS ARE ENABLED
337 043604 032737 001000 046112 BIT      #AOE,78$      :ARE DATA ERRORS ENABLED ??
338 043612 001002          BNE      31$          :YES !!
339 043614 042716 001000      BIC      #MXF,(SP)     :DISABLE MXF ERROR
340 043620 013737 001346 001142 31$: MOV      RMCS2I,$BDDAT :GET RECEIVED STATUS
341 043626 042637 001142      BIC      (SP)+,$BDDAT  :CLEAR MXF IF ENABLED
342 043632 001412          BEQ      32$          :BRANCH IF MXF IS OK
    
```



```

343 043634 062716 000004          ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
344 043640 112776 000033 000000    MOV      #33,a(SP)       ;WRITE ERROR NUMBER IN CALL
345 043646 162716 000002          SUB      #2,(SP)         ;MOVE SP TO ERROR RETURN
346 043652 004736          JSR      PC,a(SP)+       ;REPORT ERROR AND RETURN
347 043654 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
348 043660          32$:
349
350          ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
351 043660 012746 177777          MOV      #-1,-(SP)       ;ASSUME DATA ERRORS ARE ENABLED
352 043664 032737 001000 046112    BIT      #AOE,78$        ;ARE DATA ERRORS EANBLED ??
353 043672 001404          BEQ      33$             ;NO !!
354 043674 032737 002000 001350    BIT      #LBT,RMDSI      ;IS LBT ALSO SET ??
355 043702 001002          BNE      34$             ;YES !!
356 043704 042716 001000 33$:      BIC      #AOE,(SP)        ;DISABLE AOE
357 043710 013737 001352 001142 34$:      MOV      RMER1I,$BDDAT    ;GET RECEIVED STATUS
358 043716 042637 001142          BIC      (SP)+,$BDDAT    ;CLEAR AOE IF ENABLED
359 043722 001412          BEQ      35$             ;BRANCH IF AOE IS OK
360 043724 062716 000004          ADD      #4,(SP)         ;MOVE SP TO USERS ERROR CALL
361 043730 112776 000020 000000    MOV      #20,a(SP)       ;WRITE ERROR NUMBER
362 043736 162716 000002          SUB      #2,(SP)         ;MOVE SP TO ERROR RETURN
363 043742 004736          JSR      PC,a(SP)+       ;REPORT ERROR AND RETURN
364 043744 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
365 043750          35$:
366
367          ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
368          ;HEADER ERRORS, I.E.,
369          ;
370          ;
371          ;
372          ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
373 043750 032737 002000 001370    BIT      #HCI,RMOFI      ;IS HCI SET ??
374 043756 001403          BEQ      36$             ;NO !!
375 043760 042737 000200 046112    BIC      #HCE,78$        ;YES - DISABLE ALL HEADER ERRORS
376 043766          36$:
377
378          ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
379 043766 012746 177777          MOV      #-1,-(SP)       ;ASSUME ERRORS ENABLED
380 043772 032737 000200 046112    BIT      #HCE,78$        ;ARE HEADER ERRORS ENABLED ??
381 044000 001002          BNE      37$             ;YES !!
382 044002 042716 000400          BIC      #HCRC,(SP)      ;DISABLE HCRC
383 044006 013737 001352 001142 37$:      MOV      RMER1I,$BDDAT    ;GET RECEIVED STATUS
384 044014 042637 001142          BIC      (SP)+,$BDDAT    ;RESET HCRC IF ENABLED
385 044020 001412          BEQ      38$             ;BRANCH IF HCRC IS OK
386 044022 062716 000004          ADD      #4,(SP)         ;MOVE SP TO USERS ERROR CALL
387 044026 112776 000035 000000    MOV      #35,a(SP)       ;WRITE ERROR NUMBER IN CALL
388 044034 162716 000002          SUB      #2,(SP)         ;MOVE SP TO ERROR RETURN
389 044040 004736          JSR      PC,a(SP)+       ;REPORT ERROR AND RETURN
390 044042 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
391 044046          38$:
392
393          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
394 044046 012746 177777          MOV      #-1,-(SP)       ;ASSUME ERRORS ENABLED
395 044052 032737 000200 046112    BIT      #HCE,78$        ;ARE ERRORS ENABLED ??
396 044060 001002          BNE      39$             ;YES !!
397 044062 042716 000200          BIC      #HCE,(SP)      ;DISABLE HCE
398 044066 013737 001352 001142 39$:      MOV      RMER1I,$BDDAT    ;GET RECEIVED STATUS
399 044074 042637 001142          BIC      (SP)+,$BDDAT    ;CLEAR HCE IF ENABLED
    
```



```

400 044100 001412          BEQ      40$          :BRANCH IF HCE IS OK
401 044102 062716 000004  ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
402 044106 112776 000036 000000  MOVVB   #36,@(SP)    :WRITE ERROR NUMBER IN CALL
403 044114 162716 000002  SUB      #2,(SP)      :MOVE SP TO ERROR RETURN
404 044120 004736  JSR     PC,@(SP)+    :REPORT ERROR AND RETURN
405 044122 162716 000010  SUB      #10,(SP)     :MOVE SP TO NO ERROR
406 044126          40$:
407
408          ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
409 044126 012746 177777  MOV     #-1,-(SP)    :ASSUME FER IS ENABLED
410 044132 032737 000200 046112  BIT     #HCE,78$    :ARE HEADER ERRORS ENABLED ??
411 044140 001002  BNE     41$         :YES !!
412 044142 042716 000020  BIC     #FER,(SP)    :DISABLE FER
413 044146 013737 001352 001142 41$:  MOV     RMER11,$BDDAT :GET RECEIVED STATUS
414 044154 042637 001142  BIC     (SP)+,$BDDAT :RESET FER IF ENABLED
415 044160 001412  BEQ     42$         :BRANCH IF FER OK
416 044162 062716 000004  ADD     #4,(SP)      :MOVE SP TO USERS ERROR CALL
417 044166 112776 000037 000000  MOVVB   #37,@(SP)    :WRITE ERROR NUMBER IN CALL
418 044174 162716 000002  SUB     #2,(SP)      :MOVE SP TO ERROR RETURN
419 044200 004736  JSR     PC,@(SP)+    :REPORT ERROR AND RETURN
420 044202 162716 000010  SUB     #10,(SP)     :MOVE SP TO NO ERROR
421 044206          42$:
422
423          ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
424 044206 012746 177777  MOV     #-1,-(SP)    :ASSUME ERRORS ENABLED
425 044212 032737 000200 046112  BIT     #HCE,78$    :ARE THEY ENABLED ??
426 044220 001002  BNE     43$         :YES !!
427 044222 042716 100000  BIC     #BSE,(SP)    :DISABLE BSE
428 044226 013737 001400 001142 43$:  MOV     RMER21,$BDDAT :GET RECEIVED STATUS
429 044234 042637 001142  BIC     (SP)+,$BDDAT :CLEAR BSE IF ENABLED
430 044240 001412  BEQ     44$         :BRANCH IF BSE OK
431 044242 062716 000004  ADD     #4,(SP)      :MOVE SP TO USERS ERROR CALL
432 044246 112776 000354 000000  MOVVB   #354,@(SP)   :WRITE ERROR NUMBER
433 044254 162716 000002  SUB     #2,(SP)      :MOVE SP TO ERROR RETURN
434 044260 004736  JSR     PC,@(SP)+    :REPORT ERROR AND RETURN
435 044262 162716 000010  SUB     #10,(SP)     :MOVE SP TO NO ERROR
436 044266          44$:
437
438          ;HCE IS USED AS THE ENABLING BIT FOR DETECTING SSE, SO REESTABLISH ENTRY
439          ;FROM THE FUNCTION CODE TABLE, TO CHECK THE "SSE" BIT
440 044266 010046  MOV     R0,-(SP)     ;;PUSH R0 ON STACK
441 044270 013700 046120  MOV     81$,R0       :GET FUNCTION CODE
442 044274 016037 067244 046112  MOV     FNCDTB(R0),78$ :STORE FUNCTION TABLE ENTRY
443 044302 012600  MOV     (SP)+,R0     ;;POP STACK INTO R0
444
445          ;IF SKIP SECTOR ERROR INHIBIT (SSEI) IS SET OR FORMAT 16 (FMT) IS CLEAR,
446          ;RESET THE ENABLING BIT (HCE), OTHERWISE LEAVE ENABLING BIT (HCE) SET.
447 044304 032737 010000 001370  BIT     #FMT16,RMOFI :18 BIT FORMAT MODE ?
448 044312 001404  BEQ     45$         :BR IF YES
449 044314 032737 001000 001370  BIT     #SSEI,RMOFI  :IS SSEI SET ?
450 044322 001403  BEQ     46$         :BR IF NO
451 044324 042737 000200 046112 45$:  BIC     #HCE,78$    :DISABLE HEADER ERROR
452 044332          46$:
453
454          ;REPORT ERROR IF SSE IS SET AND HEADER ERROR IS NOT ENABLED
455 044332 012746 177777  MOV     #-1,-(SP)    :ASSUME ERRORS ENABLED
456 044336 032737 000200 046112  BIT     #HCE,78$    :IS HCE ENABLED ?
    
```

```

457 044344 001002          BNE      47$      :YES !!
458 044346 042716 000040    BIC      #SSE,(SP) :DISABLE SSE
459 044352 013737 001400 001142 47$:  MOV      RMER2I,$BDDAT :GET RECIEVED STATUS
460 044360 042637 001142    BIC      (SP)+,$BDDAT :CLEAR SSE IF NOT VALID FOR FUNCTION
461 044364 001412          BEQ      48$      :BRANCH IF SSE OK
462 044366 062716 000004    ADD      #4,(SP)   :MOVE SP TO USERS ERROR CALL
463 044372 112776 000063 000000    MOVVB   #63,@(SP) :WRITE ERROR NUMBER
464 044400 162716 000002    SUB      #2,(SP)   :MOVE SP TO ERROR RETURN
465 044404 004736          JSR      PC,@(SP)+ :REPORT ERROR AND RETURN
466 044406 162716 000010    SUB      #10,(SP)  :MOVE SP TO NO ERROR
467 044412          48$:
468
469          :BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
470          :FIELD ERRORS, I.E.,
471          :
472          :   RMCS2 - MDPE
473          :   RMER1 - DCK,ECH
474          :NOTE:
475          :   ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
476          :   DCK IS SET.
477          :REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
478 044412 012746 177777    MOV      #-1,-(SP) :ASSUME ENABLED
479 044416 032737 000100 046112    BIT      #ECH,78$  :ARE DATA FIELD ERRORS ENABLED ??
480 044424 001002          BNE      49$      :YES !!
481 044426 042716 000400    BIC      #MDPE,(SP) :DISBALE MDPE
482 044432 013737 001346 001142 49$:  MOV      RMCS2I,$BDDAT :GET RECEIVED STATUS
483 044440 042637 001142    BIC      (SP)+,$BDDAT :CLEAR MDPE IF ENABLED
484 044444 001412          BEQ      50$      :BRANCH IF MDPE OK
485 044446 062716 000004    ADD      #4,(SP)   :MOVE SP TO USERS ERROR CALL
486 044452 112776 000027 000000    MOVVB   #27,@(SP) :WRITE ERROR NUMBER IN CALL
487 044460 162716 000002    SUB      #2,(SP)   :MOVE SP TO ERROR RETURN
488 044464 004736          JSR      PC,@(SP)+ :REPORT ERROR AND RETURN
489 044466 162716 000010    SUB      #10,(SP)  :MOVE SP TO NO ERROR
490 044472          50$:
491
492          :REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
493 044472 012746 177777    MOV      #-1,-(SP) :ASSUME ENABLED
494 044476 032737 000100 046112    BIT      #ECH,78$  :ARE THEY ENABLED ??
495 044504 001002          BNE      51$      :YES !!
496 044506 042716 100000    BIC      #DCK,(SP)  :DISABLE DCK
497 044512 013737 001352 001142 51$:  MOV      RMER1I,$BDDAT :GET RECEIVED STATUS
498 044520 042637 001142    BIC      (SP)+,$BDDAT :CLEAR DCK IF ENABLED
499 044524 001412          BEQ      52$      :BRANCH IF DCK IS OK
500 044526 062716 000004    ADD      #4,(SP)   :MOVE SP TO USERS ERROR CALL
501 044532 112776 000030 000000    MOVVB   #30,@(SP) :WRITE ERROR NUMBER IN CALL
502 044540 162716 000002    SUB      #2,(SP)   :MOVE SP TO ERROR RETURN
503 044544 004736          JSR      PC,@(SP)+ :REPORT ERROR AND RETURN
504 044546 162716 000010    SUB      #10,(SP)  :MOVE SP TO NO ERROR
505 044552          52$:
506
507          :REPORT ERROR IF ECH IS SET AND
508          :   DATA FIELD ERRORS ARE NOT ENABLED, OR
509          :   ECI IS SET, OR
510          :   DCK IS NOT SET.
511 044552 012746 177777    MOV      #-1,-(SP) :ASSUME ENABLED
512 044556 032737 000100 046112    BIT      #ECH,78$  :ARE ERRORS ENABLED ??
513 044564 001410          BEQ      53$      :NO !!
    
```



```

514 044566 032737 004000 001370      BIT      #ECI,RMOFI      :IS ECI SET ??
515 044574 001004                      BNE      53$           :YES !!
516 044576 032737 100000 001352      BIT      #DCK,RMER1I   :IS DCK ALSO SET ??
517 044604 001002                      BNE      54$           :YES !!
518 044606 042716 000100          53$:    BIC      #ECH,(SP)    :DISABLE ECH
519 044612 013737 001352 001142  54$:    MOV      RMER1I,$BDDAT :GET RECEIVED STATUS
520 044620 042637 001142          BIC      (SP)+,$BDDAT  :CLEAR ECH IF ENABLED
521 044624 001412                      BEQ      55$           :BRANCH IF ECH IS OK
522 044626 062716 000004          ADD      #4,(SP)       :MOVE SP TO USERS ERROR CALL
523 044632 112776 000031 000000      MOVVB   #31,a(SP)     :WRITE ERROR NUMBER IN CALL
524 044640 162716 000002          SUB      #2,(SP)       :MOVE SP TO ERROR RETURN
525 044644 004736                      JSR      PC,a(SP)+     :REPORT ERROR AND RETURN
526 044646 162716 000010          SUB      #10,(SP)     :MOVE SP TO NO ERROR
527 044652          55$:
528
529
530          :*****
531          :*PERFORM THE REMAINING ERROR CHECKS FOR DATA TRANSFER COMMANDS
532          :*****
533 044652 022737 000030 046120      CMP      #SEARCH,81$   :WAS DATA TRANSFERRED ?
534 044660 103402                      BLO      56$           :BR IF YES
535 044662 000137 046064          JMP      75$           :NO - EXIT
536
537          :REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
538 044666 013737 001340 001142  56$:    MOV      RMWCI,$BDDAT  :WORD COUNT ZERO??
539 044674 001421                      BEQ      57$           :YES
540 044676 032737 040000 001336      BIT      #TRE,RMCS1I   :TRANSFER ERROR DETECTED??
541 044704 001015                      BNE      57$           :YES!!
542 044706 062716 000004          ADD      #4,(SP)
543 044712 112776 000015 000000      MOVVB   #15,a(SP)     :ERROR NUMBER
544 044720 005037 001140          CLR      $GDDAT       :GOOD DATA FOR TYPEOUT
545 044724 162716 000002          SUB      #2,(SP)       :MOVE SP TO RETURN FOR ERROR
546 044730 004736                      JSR      PC,a(SP)+     :REPORT WORD COUNT NOT ZERO
547 044732 162716 000010          SUB      #10,(SP)     :RESTORE (SP)
548 044736 000240          NOP
549
550          :REPORT ERROR IF RMBA IS NOT CORRECT
551 044740 013737 001340 001140  57$:    MOV      RMWCI,$GDDAT  :GET WORD COUNT AT END OF TRANSFER AND
552 044746 163737 001414 001140      SUB      RMWCO,$GDDAT  :SUBTRACT STARTING WORD COUNT.
553 044754 006337 001140          ASL      $GDDAT       :* 2
554 044760 063737 001416 001140      ADD      RMBAO,$GDDAT  :ADD STARTING BUS ADDRESS
555
556 044766 032737 000010 001346      BIT      #BAI,RMCS2I   :WAS BUS ADDRESS INHIBIT (BAI) SET ??
557 044774 001403                      BEQ      58$           :NO !!
558 044776 013737 001416 001140      MOV      RMBAO,$GDDAT  :ADDRESS SHOULD NOT HAVE CHANGED
559
560 045004 023737 001140 001342  58$:    CMP      $GDDAT,RMBAI   :BUS ADDRESS OK??
561 045012 001416                      BEQ      59$           :YES!!
562 045014 013737 001342 001142      MOV      RMBAI,$BDDAT  :BAD DATA FOR TYPEOUT
563 045022 062716 000004          ADD      #4,(SP)
564 045026 112776 000016 000000      MOVVB   #16,a(SP)     :ERROR NUMBER
565 045034 162716 000002          SUB      #2,(SP)       :MOVE SP TO RETURN FOR ERROR
566 045040 004736                      JSR      PC,a(SP)+     :REPORT UNEXPECTED ADDRESS
567 045042 162716 000010          SUB      #10,(SP)     :RESTORE (SP)
568 045046 000240          NOP
569
570          :COMPUTE NUMBER OF SECTORS TRANSFERRED FROM WORD COUNT
    
```

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


```

571 045050 005046          59$: CLR      -(SP)      ;NUMBER OF SECTORS TRANSFERRED
572 045052 013746 001340   MOV      RMWCI,-(SP)  ;GET WORD COUNT AT END OF TRANSFER AND
573 045056 163716 001414   SUB      RMWCO,(SP)  ;SUBTRACT STARTING WORD COUNT.
574
575 045062 012746 000400   MOV      #256,-(SP) ;ASSUME 256. WORDS PER SECTOR
576 045066 032737 000002 001412 BIT      #BIT1,RMCS10 ;HEADER & DATA COMMAND ??
577 045074 001402          BEQ      60$        ;NO !!
578 045076 062716 000002   ADD      #2,(SP)    ;CHANGE TO 258. WORDS PER SECTOR
579
580 045102 005266 000004   60$: INC      4(SP)    ;INCREMENT SECTOR COUNT
581 045106 161666 000002   SUB      (SP),2(SP) ;SUBTRACT ONE SECTOR'S WORTH
582 045112 003373          BGT      60$        ;CONTINUE IF NOT DONE
583 045114 022626          CMP      (SP)+,(SP)+;RESTORE STACK
584
585
586
587 045116 013737 001446 046112 ;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM
588 045124 013737 001420 046114 ;NUMBER OF SECTORS
589 045132 013737 001420 046116   MOV      RMDCO,78$  ;STORE ORIGINAL CYLINDER
590 045140 013737 001334 046122   MOV      RMDAO,79$  ;STORE ORIGINAL TRACK
591 045146 000337 046122          MOV      RMDAO,80$  ;STORE ORIGINAL SECTOR
592 045152 005237 046122          MOV      LSTRK,82$  ;STORE LAST TRACK,
593
594 045156 042737 000377 046114   SWAB     82$        ;GET TRACK ADDRESS TO LO BYTE AND
595 045164 000337 046114          INC      82$        ;INCREMENT TO GET TOTL # OF TRACKS.
596 045170 042737 177400 046116   BIC      #^CTADMSK,79$ ;SAVE TRACK ADDRESS BITS AND
597 045176 062637 046116   SWAB     79$        ;GET TRACK ADDRESS TO LO BYTE.
598
599 045202 032737 001000 001444   BIC      #^CSADMSK,80$ ;SAVE SECTOR ADDRESS BITS
600 045210 001411          ADD      (SP)+,80$
601
602 045212 023727 046116 000040   BIT      #SSEI,RMOFO ;WAS SSEI SET ?
603 045220 103417          BEQ      61$        ;NO !!
604 045222 005237 046114          CMP      80$,#32.   ;SECTOR OVEFLOWED??
605 045226 162737 000040 046116   BLO     63$        ;NO!!
606
607 045234 023727 046116 000037 61$: INC      79$        ;INCREMENT TRACK
608 045242 103406          SUB     #32.,80$    ;ADJUST SECTOR
609 045244 005237 046114          62$: CMP      80$,#31. ;SECTOR OVEFLOWED ?
610 045250 162737 000037 046116   BLO     63$        ;NO !!
611 045256 000766          INC      79$        ;INCREMENT TRACK
612
613 045260 023737 046114 046122 63$: SUB     #31.,80$    ;ADJUST SECTOR
614 045266 103407          BR      61$        ;TRY AGAIN
615 045270 005237 046112          64$: CMP      79$,82$ ;TRACK OVEFLOWED??
616 045274 163737 046122 046114   BLO     64$        ;NO!!
617 045302 000766          INC      78$        ;INCREMENT CYLINDER
618 045304 000240          SUB     82$,79$    ;ADJUST TRACK
619
620
621
622 045306 005037 001140          65$: BR      63$        ;TRY AGAIN
623 045312 013737 001370 001142 ;REPORT ERROR IF "SSEI" IS SET AFTER TRACK SWITCHING OCCURED
624 045320 042737 176777 001142 ;THRU DATA TRANSFER
625
626 045326 032737 001000 001444   64$: CLR      $GDDAT  ;SETUP EXPECTED SSEI = 0
627 045334 001420          MOV      RMOFI,$BDDAT ;GET RECIEVED DATA AND
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
    
```

```

628 045336 123737 001421 001345      CMPB   RMDAO+1,RMDAI+1 ;DID TRACK SWITCH OCCUR ?
629 045344 001411                      BEQ    65$              ;NO
630 045346 005737 001142                      TST   $BDDAT          ;IS SSEI STILL SET ?
631 045352 001430                      BEQ    68$              ;NO
632 045354 062716 000004                      ADD   #4,(SP)         ;POINT TO ERROR
633 045360 112776 000225 000000          MOVB  #225,a(SP)     ;DEFINE ERROR NUMBER
634 045366 000414                      BR    67$              ;
635
636                                     ;REPORT ERROR IF "SSEI" IS INCORRECT
637 045370 012737 001000 001140          65$:  MOV   #SSEI,$GDDAT ;SETUP EXPECTED SSEI = 1
638 045376 023737 001140 001142          66$:  CMP   $GDDAT,$BDDAT ;IS SSEI CORRECT ?
639 045404 001413                      BEQ    68$              ;YES
640 045406 062716 000004                      ADD   #4,(SP)         ;POINT TO ERROR
641 045412 112776 000226 000000          MOVB  #226,a(SP)     ;DEFINE ERROR NUMBER
642 045420 162716 000002                      67$:  SUB   #2,(SP)     ;ADJUST RETURN TO ERROR
643 045424 004736                      JSR   PC,a(SP)+      ;REPORT ERROR
644 045426 162716 000010                      SUB   #10,(SP)       ;RESTORE (SP)
645 045432 000240                      NOP
646
647                                     ;REPORT ERROR IF "LBT" IS NOT CORRECT
648 045434 005037 001140                      68$:  CLR   $GDDAT      ;SET GOOD DATA FOR LBT = 0
649 045440 023727 046112 001060          CMP   78$,#560.     ;SHOULD LBT BE SET??
650 045446 101407                      BLOS  69$              ;NO!!
651 045450 032737 002000 001352          BIT   #IAE,RMER1I   ;WAS IAE SET ??
652 045456 001003                      BNE   69$              ;YES - LBT SHOULD NOT BE SET
653 045460 012737 002000 001140          MOV   #LBT,$GDDAT  ;SET GOOD DATA FOR LBT = 1
654 045466 013737 001350 001142          69$:  MOV   RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
655 045474 042737 175777 001142          BIC   #^CLBT,$BDDAT
656 045502 023737 001140 001142          CMP   $GDDAT,$BDDAT ;IS LBT CORRECT??
657 045510 001413                      BEQ    70$              ;YES!!
658 045512 062716 000004                      ADD   #4,(SP)
659 045516 112776 000017 000000          MOVB  #17,a(SP)     ;ERROR NUMBER
660 045524 162716 000002                      SUB   #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
661 045530 004736                      JSR   PC,a(SP)+      ;REPORT LBT IS WRONG
662 045532 162716 000010                      SUB   #10,(SP)       ;RESTORE (SP)
663 045536 000240                      NOP
664
665                                     ;REPORT ERROR IF "AOE" IS INCORRECT
666 045540 005037 001140                      70$:  CLR   $GDDAT      ;SET FOR AOE = 0
667 045544 032737 002000 001352          BIT   #IAE,RMER1I   ;WAS "IAE" DETECTED??
668 045552 001031                      BNE   72$              ;YES-"AOE" SHOULD BE ZERO
669 045554 023727 046112 001060          CMP   78$,#560.     ;SHOULD AOE BE SET??
670 045562 101425                      BLOS  72$              ;NO!!
671 045564 005737 046114                      TST   79$              ;MAYBE
672 045570 001012                      BNE   71$              ;YES
673 045572 005737 046116                      TST   80$
674 045576 001007                      BNE   71$              ;YES !!
675 045600 032737 000010 046120          BIT   #F2,81$ ;WAS THIS READ OR WRITE CHECK ??
676 045606 001413                      BEQ    72$              ;NO !!
677 045610 005737 001340                      TST   RMWCI          ;WAS ALL DATA TRANSFERRED ??
678 045614 001410                      BEQ    72$              ;YES !!
679 045616 012737 001000 001140          71$:  MOV   #AOE,$GDDAT ;SET FOR AOE = 1
680 045624 005037 046114                      CLR   79$              ;CLEAR EXPECTED TRACK
681 045630 012737 000001 046116          MOV   #1,80$         ;EXPECT SECTOR = 1
682 045636 013737 001352 001142          72$:  MOV   RMER1I,$BDDAT ;BAD DATA FOR TYPEOUT
683 045644 042737 176777 001142          BIC   #^CAOE,$BDDAT
684 045652 023737 001140 001142          CMP   $GDDAT,$BDDAT ;IS AOE CORRECTY??
    
```



```

685 045660 001413          BEQ      73$          :YES!!
686 045662 062716 000004  ADD      #4,(SP)
687 045666 112776 000020 000000  MOVB    #20,a(SP)   :ERROR NUMBER
688 045674 162716 000002          SUB      #2,(SP)   :MOVE SP TO RETURN FOR ERROR
689 045700 004736          JSR     PC,a(SP)+  :REPORT AOE IS WRONG
690 045702 162716 000010  SUB     #10,(SP)   :RESTORE (SP)
691 045706 000240          NOP
692
693
694 045710 032737 002000 001352 :REPORT ERROR IF RMDA IS NOT CORRECT
73$: BIT   #IAE,RMER1I :WAS THERE AN IAE ERROR ??
695 045716 001062          BNE     75$        :YES - DONT CHECK RMDA,RMDC
696 045720 013737 046114 001140  MOV     79$,SGDDAT :SETUP EXPECTED DISK ADDRESS
697 045726 000337 001140          SWAB   SGDDAT
698 045732 113737 046116 001140  MOVB   80$,SGDDAT
699 045740 013737 001344 001142  MOV     RMDAI,$BDDAT :SETUP RECEIVED DISK ADDRESS
700 045746 023737 001140 001142  CMP     SGDDAT,$BDDAT :COMPARE EXPECTED & RECEIVED
701 045754 001413          BEQ     74$        :BRANCH IF EQUAL
702 045756 062716 000004  ADD     #4,(SP)
703 045762 112776 000021 000000  MOVB   #21,a(SP)   :ERROR NUMBER
704 045770 162716 000002          SUB     #2,(SP)   :MOVE SP TO RETURN FOR ERROR
705 045774 004736          JSR     PC,a(SP)+  :REPORT BAD DISK ADDRESS
706 045776 162716 000010  SUB     #10,(SP)   :RESTORE (SP)
707 046002 000240          NOP
708
709
710 046004 013737 046112 001140 :REPORT ERROR IF RMDC IS INCORRECT
74$: MOV     78$,SGDDAT :SETUP EXPECTED CYLINDER
711 046012 042737 176000 001140  BIC    #C1777,SGDDAT
712 046020 013737 001372 001142  MOV     RMDCI,$BDDAT :SETUP RECEIVED CYLINDER
713 046026 023737 001140 001142  CMP     SGDDAT,$BDDAT :COMPARE CYLINDERS
714 046034 001413          BEQ     75$        :BRANCH IF EQUAL
715 046036 062716 000004  ADD     #4,(SP)
716 046042 112776 000022 000000  MOVB   #22,a(SP)   :ERROR NUMBER
717 046050 162716 000002          SUB     #2,(SP)   :MOVE SP TO RETURN FOR ERROR
718 046054 004736          JSR     PC,a(SP)+  :REPORT BAD CYLINDER
719 046056 162716 000010  SUB     #10,(SP)   :RESTORE (SP)
720 046062 000240          NOP
721
722 046064 062716 000004          75$:  ADD     #4,(SP)   :MOVE (SP) TO ERROR CALL
723 046070 105776 000000          TSTB   a(SP)      :WAS ERROR FOUND??
724 046074 001403          BEQ     76$
725 046076 062716 000004          ADD     #4,(SP)   :MOVE (SP) TO ERROR RETURN
726 046102 000402          BR      77$
727 046104 162716 000004          76$:  SUB     #4,(SP)   :MOVE (SP) TO NO ERROR RETURN
728 046110 000207          77$:  RTS      PC
729
730 046112 000000          78$:  .WORD  0       :CYLINDER
731 046114 000000          79$:  .WORD  0       :TRACK
732 046116 000000          80$:  .WORD  0       :SECTOR
733 046120 000000          81$:  .WORD  0       :FUNCTION CODE
734 046122 000000          82$:  .WORD  0       :TOTAL # OF TRACKS = LAST TRACK +1
    
```


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

.SBTTL COMPOSITE ERROR CHECK SUBROUTINE

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

```

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

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

CMPEERRSTS:

```

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

```

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

```

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

1\$:

```

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

```

046124
046124 013737 001352 001176
046132 047637 000000 001176
046140 062716 000002
046144 013737 001400 001200
046152 047637 000000 001200
046160 062716 000006
046164 105076 000000
046170 162716 000004
046174 005737 001176
046200 001420
046202 013737 001176 001142
046210 005037 001140
046214 062716 000004
046220 112776 000066 000000
046226 162716 000002
046232 004736
046234 162716 000010
046240 000240
046242
046242 005737 001200
046246 001420
046250 013737 001200 001142
046256 005037 001140
046262 062716 000004
046266 112776 000067 000000
046274 162716 000002
    
```



```

1      .SBTTL  DEVICE SELECT SUBROUTINE
2
3      ; THIS SUBROUTINE SELECTS THE DEVICE, GETTING THE DEVICE NUMBER FROM THE
4      ; TEST QUEUE.
5
6      ;CALL:
7      ;(1)   JSR      PC,DEVSEL
8      ;(2)   BR       ??          RETURN IF NO ERROR
9      ;(3)   NOP
10      ;(4)   ERROR          RETURN IF ERROR
11                                     ERROR DEFINED BY SUBROUTINE
12 046336  DEVSEL:
13
14      ;CLEAR USER'S ERROR CALL
15 046336 062716 000004      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR
16 046342 105076 000000      CLRB   @ (SP)          ;CLEAR LOW ORDER BYTE OF CALL
17 046346 162716 000004      SUB      #4,(SP)          ;MOVE SP BACK
18
19      ;SAVE USER'S INFORMATION AND SETUP REGISTERS
20 046352 013746 000004      MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
21 046356 013746 000006      MOV      ERRVEC+2,-(SP)  ;;PUSH ERRVEC+2 ON STACK
22 046362 010046             MOV      R0,-(SP)          ;;PUSH R0 ON STACK
23 046364 010146             MOV      R1,-(SP)          ;;PUSH R1 ON STACK
24 046366 012737 046506 000004  MOV      #2$,ERRVEC      ;SETUP FOR BUS TIMEOUT
25 046374 012737 000300 000006  MOV      #PR6,ERRVEC+2
26 046402 013700 001276             MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
27 046406 013701 001466             MOV      TSTQUE,R1        ;R1 POINTS TO DEVICE NUMBER
28
29      ;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
30
31 046412 111160 000010             MOV      (R1),RMCS2(R0)   ;WRITE UNIT SELECT BITS
32 046416 016037 000000 001176     MOV      RMCS1(R0),STMP1 ;GET 'DVA' STATUS
33 046424 016037 000010 001174     MOV      RMCS2(R0),STMP0 ;GET 'NED' STATUS
34
35 046432 032737 010000 001174     BIT      #NED,STMP0      ;IS DEVICE NONEXISTENT ?
36 046440 001407             BEQ      1$              ;NO!!
37 046442 062766 000004 000010     ADD      #4,10(SP)       ;MOVE SP TO USERS ERROR CALL
38 046450 112776 000111 000010     MOV      #11,@10(SP)    ;WRITE ERROR NUMBER
39 046456 000422             BR       3$
40
41 046460 032737 004000 001176 1$:  BIT      #DVA,STMP1      ;IS DEVICE AVAILABLE ?
42 046466 001021             BNE      4$              ;YES!!
43 046470 062766 000004 000010     ADD      #4,10(SP)       ;MOVE SP TO USERS ERROR CALL
44 046476 112776 000112 000010     MOV      #112,@10(SP)   ;WRITE ERROR NUMBER
45 046504 000407             BR       3$
46
47      ;HANDLE BUS TIMEOUT
48
49 046506 022626 000004 000010 2$:  CMP      (SP)+,(SP)+     ;ADJUST SP
50 046510 062766 000004 000010     ADD      #4,10(SP)       ;MOVE SP TO USERS ERROR CALL
51 046516 112776 000113 000010     MOV      #113,@10(SP)   ;WRITE BUS TIMEOUT ERROR NUMBER
52 046524 162766 000002 000010 3$:  SUB      #2,10(SP)       ;ADJUST RETURN TO 'NOP' PRECEDING
53                                     ;THE ERROR CALL
54
55      ;RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
56 4$:  MOV      (SP)+,R1        ;;POP STACK INTO R1
57     MOV      (SP)+,R0        ;;POP STACK INTO R0
58     MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
59
60 046532 012601
61 046534 012600
62 046536 012637 000006
    
```


DEVICE SELECT SUBROUTINE

046542 012637 000004
52 046546 000207

MOV
RTS

(SP)+,ERRVEC
PC

::POP STACK INTO ERRVEC
: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      :THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS DETECTED
8      :AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN THE "ERROR" TRAP
9      :OF THE CALLING ROUTINE.  SEEK STATUS IS CHECKED AS FOLLOWS:
10
11      :CALL:
12      : (1)  JSR    PC,SEKSTS
13      :      BR    ???
14      :      NOP
15      :      ERROR
16      :      JSR    PC,@(SP)+
17      :      ???
18
19      SEKSTS:
20
21      :CLEAR USERS' ERROR CALL
22      :      NOP
23      :      ADD    #4,(SP)
24      :      CLR   @ (SP)
25      :      SUB    #4,(SP)
26      :      CLR   20$
27
28      :TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
29      :LOCAL REGISTERS, I.E., "PAR" = 1 AND "DPE" = 0
30      :      BIT    #PAR,RMER1I
31      :      BEQ    1$
32      :      BIT    #DPE,RMER2I
33      :      BNE    1$
34
35      :REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
36      :      CLR    $GDDAT
37      :      MOV    RMER1I,$BDDAT
38      :      ADD    #4,(SP)
39      :      MOV   #50,@(SP)
40      :      SUB    #2,(SP)
41      :      JSR    PC,@(SP)+
42      :      SUB    #10,(SP)
43      :      BR    3$
44
45      :DETERMINE THE VALUE OF "IAE" STATUS BASED ON TRACK, SECTOR, CYLINDER,
46      :AND THE STATUS OF "SSEI" BIT DURING A SEEK OPERATION. ALSO, SET "SKI"
47      :IF CYLINDER ADDRESS IS TOO LARGE.
48      1$:  MOV    #IAE,$GDDAT
49      :      BIS    #SKI,20$
50      :      CMP    RMDCO,#560.
51      :      BHI    3$
52      :      BIC    #SKI,20$
53
54      :      CMPB  RMDAO+1,LSTRK+1
55      :      BHI    3$
56
57      :      CMPB  RMDAO,#29.
    
```

```

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

```

:MOVE (SP) TO ERROR CALL
:CLEAR ERROR NUMBER
:MOVE (SP) TO NO ERROR RETURN
:CLEAR ERROR FLAGS
    
```

```

:WAS PARITY ERROR DETECTED??
:NO!!
:WAS IT DUE TO CONTROL BUS??
:NOT SURE!!
    
```

```

:EXPECTED STATUS
:RECEIVED STATUS
:MOVE STACK TO USER'S ERROR
:ERROR #50
:MOVE SP TO RETURN FOR ERROR
:RESTORE STACK
:IAE SHOULD BE ZERO
    
```

```

:SETUP FOR IAE = 1
:SETUP FOR SKI = 1
:GREATER THAN LAST CYLINDER ?
:YES - CYLINDER IS INVALID
:CLEAR SKI ERROR FLAG
    
```

```

:GREATER THAN LAST TRACK ?
:YES - TRACK IS INVALID
    
```

```

:SECTOR > 29. ?
    
```

58	046722	101420			BLOS	2\$:BR IF NO
59	046724	032737	010000	001444	BIT	#FMT16,RMOFO	:18 BIT FORMAT ?
60	046732	001416			BEQ	3\$:YES - SECTOR IS INVALID FOR 18 BIT MODE
61	046734	123727	001420	000036	CMPB	RMDAO,#30.	:SECTOR > 30. ?
62	046742	101410			BLOS	2\$:BR IF NO
63	046744	032737	001000	001444	BIT	#SSEI,RMOFO	:IS SSEI CLEAR ?
64	046752	001406			BEQ	3\$:YES - SECTOR IS INVALID FOR 16 BIT MODE
65	046754	123727	001420	000037	CMPB	RMDAO,#31.	:SECTOR > 31. ?
66	046762	101002			BHI	3\$:YES - SECTOR IS INVALID
67							
68	046764	005037	001140		2\$: CLR	\$GDDAT	: "IAE" SHOULD = 0
69							
70							
71	046770	013737	001352	001142	3\$: MOV	RMER1I,\$BDDAT	:COMPARE EXPECTED AND RECIEVED "IAE" STATUS
72	046776	042737	175777	001142	BIC	#^CIAE,\$BDDAT	:IS IAE OK??
73	047004	023737	001140	001142	CMP	\$GDDAT,\$BDDAT	:SAVE IAE BIT FOR COMPARE
74	047012	001004			BNE	4\$:CORRECT "IAE" STATUS ?
75	047014	042737	040000	050026	BIC	#SKI,20\$:BR IF NO
76	047022	000413			BR	5\$:CLEAR SKI FLAG
77	047024				4\$:		:GO CHECK NEXT ERROR
78					:REPORT	INCORRECT "IAE" STATUS VIA USER'S ERROR CALL	
79	047024	062716	000004		ADD	#4,(SP)	
80	047030	112776	000051	000000	MOVB	#5I,@(SP)	:ERROR 51
81	047036	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
82	047042	004736			JSR	PC,@(SP)+	:REPORT INCORRECT IAE
83	047044	162716	000010		SUB	#10,(SP)	:RESTORE (SP)
84	047050	000240			NOP		
85	047052				5\$:		
86							
87					:REPORT	ANY IVC ERROR AS	
88					:	IVC ERROR WITH VOLUME VALID ZERO	
89					:	ERRONEOUS IVC ERROR, VOLUME VALID IS SET	
90	047052	032737	010000	001400	BIT	#IVC,RMER2I	:IVC ERROR??
91	047060	001427			BEQ	7\$:NO!!
92	047062	005037	001140		CLR	\$GDDAT	:EXPECTED STATUS
93	047066	013737	001400	001142	MOV	RMER2I,\$BDDAT	:RECEIVED STATUS
94	047074	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR
95	047100	112776	000060	000000	MOVB	#60,@(SP)	:ERROR 60 IF VV = 0
96	047106	032737	000100	001350	BIT	#VV,RMDSI	
97	047114	001403			BEQ	6\$	
98	047116	112776	000061	000000	MOVB	#61,@(SP)	:ERROR 61 IF VV = 1
99	047124	162716	000002		6\$: SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
100	047130	004736			JSR	PC,@(SP)+	:REPORT ERROR VIA USER
101	047132	162716	000010		SUB	#10,(SP)	:RESTORE SP
102	047136	000240			NOP		
103							
104	047140	013737	001400	001142	7\$: MOV	RMER2I,\$BDDAT	:RECEIVED STATUS
105	047146	042737	137777	001142	BIC	#^CSKI,\$BDDAT	:CLEAR DONT CARES
106	047154	013737	050026	001140	MOV	20\$,\$GDDAT	:GET EXPECTED SKI STATUS
107	047162	042737	137777	001140	BIC	#^CSKI,\$GDDAT	:CLEAR DONT CARES
108	047170	001417			BEQ	8\$:BRANCH IF 0 EXPECTED
109							
110					:REPORT	ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)	
111	047172	032737	040000	001142	BIT	#SKI,\$BDDAT	:WAS SKI DETECTED ??
112	047200	001032			BNE	9\$:YES !!
113	047202	062716	000004		ADD	#4,(SP)	:MOVE SP TO USERS ERROR CALL
114	047206	112776	000267	000000	MOVB	#267,@(SP)	:WRITE ERROR NUMBER


```

115 047214 162716 000002      SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
116 047220 004736      JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
117 047222 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR
118 047226 000443      BR       10$         ;GO TO NEXT ERROR CHECK
119 047230
120
121      8$:
121      :REPORT ERROR IF SKI IS SET
122 047230 032737 040000 001142      BIT      #SKI,$BDDAT  ;IS SKI SET ??
123 047236 001413      BEQ      9$         ;NO - SKI IS OK
124 047240 062716 000004      ADD      #4,(SP)     ;MOVE (SP) TO ERROR
125 047244 112776 000054 000000      MOV      #54,@(SP)   ;LOAD ERROR NUMBER
126 047252 162716 000002      SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
127 047256 004736      JSR      PC,@(SP)+    ;REPORT SEEK ERROR
128 047260 162716 000010      SUB      #10,(SP)    ;RESTORE (SP)
129 047264 000240      NOP
130
131      :REPORT ANY DEVICE CHECK
132 047266 032737 000200 001400      9$: BIT      #DVC,RMER2I ;WAS THERE DVC DURING SEEK??
133 047274 001420      BEQ      10$        ;NO!!
134 047276 005037 001140      CLR      $GDDAT      ;EXPECTED STATUS
135 047302 013737 001400 001142      MOV      RMER2I,$BDDAT ;RECEIVED STATUS
136 047310 062716 000004      ADD      #4,(SP)
137 047314 112776 000055 000000      MOV      #55,@(SP)   ;ERROR #55
138 047322 162716 000002      SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
139 047326 004736      JSR      PC,@(SP)+    ;REPORT ERROR VIA USER
140 047330 162716 000010      SUB      #10,(SP)    ;RESTORE SP
141 047334 000240      NOP
142
143      :REPORT ANY 'OPI' ERROR AS OPI WITH MOL = 0, OR OPI
144      :BECAUSE ON CYLINDER LATCH DIDN'T RESET
145 047336 032737 020000 001352      10$: BIT      #OPI,RMER1I ;'OPI' ERROR??
146 047344 001427      BEQ      12$        ;NO!!
147 047346 005037 001140      CLR      $GDDAT      ;EXPECTED STATUS
148 047352 013737 001352 001142      MOV      RMER1I,$BDDAT ;RECEIVED STATUS
149 047360 062716 000004      ADD      #4,(SP)
150 047364 112776 000052 000000      MOV      #52,@(SP)   ;MOVE (SP) TO ERROR
151 047372 032737 010000 001350      MOV      #52,@(SP)   ;LOAD ERROR NUMBER
152 047400 001403      BIT      #MOL,RMDSI   ;WAS MEDIUM ON LINE??
153 047402 112776 000053 000000      BEQ      11$        ;NO!!
154 047410 162716 000002      MOV      #53,@(SP)   ;YES - CHANGE ERROR NUMBER
155 047414 004736      SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
156 047416 162716 000010      JSR      PC,@(SP)+    ;REPORT 'OPI' ERROR
157 047422 000240      SUB      #10,(SP)    ;RESTORE (SP)
158
159      :SEE IF 'PIP' = 0, AND 'ATA', 'MOL' AND 'VV' = 1
160 047424 013746 001350      12$: MOV      RMDSI,-(SP)
161 047430 042716 047677      BIC      #^C<ATA!PIP!MOL!VV>,(SP)
162 047434 022726 110100      CMP      #ATA!MOL!VV,(SP)+
163 047440 001002      BNE      13$         ;ERROR IN RMDS
164 047442 000137 047776      JMP      17$         ;RMDS IS OK
165
166      :REPORT ERROR IF MOL = 0 AND OPI = 0
167 047446 032737 010000 001350      13$: BIT      #MOL,RMDSI ;IS MOL RESET??
168 047454 001030      BNE      14$        ;NO - MOL IS SET
169 047456 032737 020000 001352      BIT      #OPI,RMER1I ;WAS OPI SET
170 047464 001024      BNE      14$        ;YES - DONT REPORT ERROR
171 047466 013737 001350 001140      MOV      RMDSI,$GDDAT ;EXPECTED STATUS
    
```

```

172 047474 052737 010000 001140      BIS      #MOL,$GDDAT
173 047502 013737 001350 001142      MOV      RMDSI,$BDDAT      ;RECEIVED STATUS
174 047510 062716 000004      ADD      #4,(SP)
175 047514 112776 000062 000000      MOVVB   #62,@(SP)
176 047522 162716 000002      SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
177 047526 004736      JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
178 047530 162716 000010      SUB      #10,(SP)
179 047534 000240      NOP
180
181      ;REPORT AN ERROR IF "PIP" IS STIL SET AND SKI NOT SET
182 047536 032737 020000 001350 14$: BIT      #PIP,RMDSI      ;IS "PIP" STILL SET??
183 047544 001430      BEQ     15$              ;NO!!
184 047546 032737 040000 001400      BIT      #SKI,RMER2I     ;WAS "SKI" SET??
185 047554 001024      BNE     15$              ;YES-DONT REPORT PIP
186 047556 013737 001350 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
187 047564 042737 020000 001142      BIC      #PIP,$BDDAT
188 047572 013737 001350 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
189 047600 062716 000004      ADD      #4,(SP)          ;MOVE (SP) TO ERROR
190 047604 112776 000056 000000      MOVVB   #56,@(SP)        ;LOAD ERROR NUMBER
191 047612 162716 000002      SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
192 047616 004736      JSR      PC,@(SP)+        ;REPORT "PIP" SET AFTER SEEK
193 047620 162716 000010      SUB      #10,(SP)        ;RESTORE (SP)
194 047624 000240      NOP
195
196      ;REPORT AN ERROR IF "ATA" IS NOT SET
197 047626 032737 100000 001350 15$: BIT      #ATA,RMDSI     ;WAS "ATA" SET ??
198 047634 001024      BNE     16$              ;YES!!
199 047636 013737 001350 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
200 047644 052737 110600 001140      BIS      #ATA!MOL!DPR!DRY,$GDDAT
201 047652 013737 001350 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
202 047660 062716 000004      ADD      #4,(SP)          ;MOVE (SP) TO ERROR
203 047664 112776 000057 000000      MOVVB   #57,@(SP)        ;LOAD ERROR NUMBER
204 047672 162716 000002      SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
205 047676 004736      JSR      PC,@(SP)+        ;REPORT ATTENTION NOT SET DURING
206      ;SEEK TEST
207 047700 162716 000010      SUB      #10,(SP)        ;RESTORE (SP)
208 047704 000240      NOP
209
210      ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
211 047706 032737 000100 001350 16$: BIT      #VV,RMDSI     ;IS VV = 0 ??
212 047714 001030      BNE     17$              ;NO!!
213 047716 032737 010000 001400      BIT      #IVC,RMER2I     ;IS IVC ALSO 0 ??
214 047724 001024      BNE     17$              ;NO - IVC IS SET
215 047726 013737 001350 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
216 047734 052737 000100 001140      BIS      #VV,$GDDAT
217 047742 013737 001350 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
218 047750 062716 000004      ADD      #4,(SP)
219 047754 112776 000064 000000      MOVVB   #64,@(SP)        ;ERROR #64
220 047762 162716 000002      SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
221 047766 004736      JSR      PC,@(SP)+
222 047770 162716 000010      SUB      #10,(SP)
223 047774 000240      NOP
224 047776      17$:
225
226      ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
227 047776 000240      NOP
228 050000 062716 000004      ADD      #4,(SP)          ;MOVE (SP) TO ERROR CALL
    
```


229	050004	105776	000000		TSTB	@(SP)		:WAS ERROR CALLED??
230	050010	001403			BEQ	18\$:NO!!
231	050012	062716	000004		ADD	#4,(SP)		:MOVE TO ERROR RETURN
232	050016	000402			BR	19\$		
233								
234	050020	162716	000004	18\$:	SUB	#4,(SP)		:MOVE (SP) TO NO ERROR RETURN
235	050024	000207		19\$:	RTS	PC		:RETURN
236								
237	050026	000000		20\$:	.WORD	0		:ERROR FLAGS


```

1      .SBTTL  CONTROLLER CLEAR SUBROUTINE
2
3      ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
4      ;AND DRIVES, THEN SELECTS THE DRIVE.
5
6      ;CALL:  JSR      PC,CNTCLR
7              BR      ???
8              NOP
9              ERROR
10             ???
11
12      CNTCLR:
13      050030 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
14      050032 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
15      050034 013746 000004   MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
16      050040 013746 000006   MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
17      050044 012737 050104 000004   MOV      #1$,ERRVEC   ;SETUP FOR BUS TIMEOUT
18      050052 012737 000300 000006   MOV      #PR6,ERRVEC+2
19      050060 013700 001276          MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
20      050064 012760 000040 000010   MOV      #CLR,RMCS2(R0) ;CLEAR MASSBUS
21      050072 013701 001466          MOV      TSTQUE,R1   ;GET DEVICE UNDER TEST
22      050076 111160 000010          MOVB    (R1),RMCS2(R0) ;SELECT DEVICE
23      050102 000412          BR      2$
24
25      21: 050104 022626          CMP      (SP)+,(SP)+  ;ADJUST STACK
26      050106 062766 000004 000010   ADD      #4,10(SP)    ;MOVE SP TO USER'S ERROR CALL
27      050114 112776 000007 000010   MOVB    #7,@10(SP)   ;WRITE THE ERROR NUMBER
28      050122 162766 000002 000010   SUB      #2,10(SP)   ;ADJUST SP TO RETURN TO ERROR
29
30      25: 050130 012637 000006          MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
31      050134 012637 000004          MOV      (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
32      050140 012601          MOV      (SP)+,R1     ;;POP STACK INTO R1
33      050142 012600          MOV      (SP)+,R0     ;;POP STACK INTO R0
34      050144 000207          RTS      PC
    
```

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

.SBTTL CONTROLLER CLEAR STATUS CHECK SUBROUTINE

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

:STATUS PERTINENT TO THE DEVICE IS NOT CHECKED. IN PARTICULAR, THE
 :FOLLOWING STATUS BITS ARE NOT CHECKED:

ATA,ERR,PIP,MOL,WRL,LBT,PGM,VV,OM,UNS,SKI,DVC

```

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

CLRSTS:

:CLEAR USER'S ERROR CALL

```

      ADD    #4,(SP)      ;MOVE SP TO ERROR
      CLRB   @ (SP)      ;CLEAR ERROR NUMBER
      SUB    #4,(SP)      ;MOVE SP BACK TO NO ERROR
:REPORT ERROR IF RMCS1 NOT INITIALIZED
1$:  MOV    RMCS1I,$BDDAT ;VERIFY RMCS1
      BIC   #SC,$BDDAT   ;IGNORE SPECIAL CONDITION
      MOV   #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
      CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED, RECEIVED
      BEQ   2$           ;BRANCH IF EQUAL
      ADD   #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
      MOVB #126,@(SP)    ;WRITE ERROR NUMBER IN CALL
      SUB   #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
      JSR   PC,@(SP)+    ;REPORT ERROR VIA USER
      SUB   #10,(SP)     ;MOVE SP BACK TO NO ERROR
      NOP
    
```

:REPORT ERROR IF RMBA NOT RESET

```

2$:  CLR    $GDDAT       ;VERIFY RMBA IS ZERO
      MOV   RMBAI,$BDDAT
      BEQ   3$           ;BRANCH IF ZERO
      ADD   #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
      MOVB #127,@(SP)    ;WRITE ERROR NUMBER IN CALL
      SUB   #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
      JSR   PC,@(SP)+    ;REPORT ERROR VIA USER
      SUB   #10,(SP)     ;MOVE SP BACK TO NO ERROR
      NOP
    
```

:REPORT ERROR IF RMCS2 NOT INITIALIZED

```

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

```

050146
050146 062716 000004
050152 105076 000000
050156 162716 000004
050162 013737 001336 001142
050170 042737 100000 001142
050176 012737 004200 001140
050204 023737 001140 001142
050212 001413
050214 062716 000004
050220 112776 000126 000000
050226 162716 000002
050232 004736
050234 162716 000010
050240 000240
050242 005037 001140
050246 013737 001342 001142
050254 001413
050256 062716 000004
050262 112776 000127 000000
050270 162716 000002
050274 004736
050276 162716 000010
050302 000240
050304 013737 001346 001142
050312 010146
050314 005046
050316 013701 001466
050322 111116
050324 052716 000100
050330 012637 001140
    
```



```

58 050334 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
59 050336 023737 001140 001142  CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED & RECEIVED
60 050344 001413          BEQ      4$              ;;BRANCH IF EQUAL
61 050346 062716 000004          ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
62 050352 112776 000130 000000  MOVVB   #130,a(SP)        ;;WRITE ERROR NUMBER IN CALL
63 050360 162716 000002          SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
64 050364 004736          JSR      PC,a(SP)+        ;;REPORT ERROR VIA USER
65 050366 162716 000010          SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
66 050372 000240          NOP
67                                ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
68 050374 005037 001140          4$: CLR      $GDDAT          ;;VERIFY RMER1
69 050400 013737 001352 001142  MOV      RMER1I,$BDDAT
70 050406 042737 040000 001142  BIC      #UNS,$BDDAT      ;;IGNORE UNSAFE
71 050414 001413          BEQ      5$              ;;BRANCH IF ZERO
72 050416 062716 000004          ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
73 050422 112776 000131 000000  MOVVB   #131,a(SP)        ;;WRITE ERROR NUMBER IN CALL
74 050430 162716 000002          SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
75 050434 004736          JSR      PC,a(SP)+        ;;REPORT ERROR VIA USER
76 050436 162716 000010          SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
77 050442 000240          NOP
78                                ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
79 050444 013737 001362 001142  5$: MOV      RMMR1I,$BDDAT    ;;VERIFY RMMR
80 050452 042737 000046 001142  BIC      #WC!LS!LST,$BDDAT ;;IGNORE WORD CLOCK, SCT, TRK
81 050460 012737 000010 001140  MOV      #MWD,$GDDAT      ;;EXPECT WRITE DATA BIT
82 050466 023737 001140 001142  CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED AND RECEIVED
83 050474 001413          BEQ      6$              ;;BRANCH IF 0
84 050476 062716 000004          ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
85 050502 112776 000133 000000  MOVVB   #133,a(SP)        ;;WRITE ERROR NUMBER IN CALL
86 050510 162716 000002          SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
87 050514 004736          JSR      PC,a(SP)+        ;;REPORT ERROR VIA USER
88 050516 162716 000010          SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
89 050522 000240          NOP
90                                ;REPORT AN ERROR IF RMEC2 IS NOT RESET
91 050524 005037 001140          6$: CLR      $GDDAT          ;;EXPECT ZEROS
92 050530 013737 001404 001142  MOV      RMEC2I,$BDDAT    ;;VERIFY RMEC2=0
93 050536 001413          BEQ      7$              ;;BRANCH IF ZERO
94 050540 062716 000004          ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
95 050544 112776 000135 000000  MOVVB   #135,a(SP)        ;;WRITE ERROR NUMBER IN CALL
96 050552 162716 000002          SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
97 050556 004736          JSR      PC,a(SP)+        ;;REPORT ERROR VIA USER
98 050560 162716 000010          SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
99 050564 000240          NOP
100                                ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
101 050566 013737 001376 001142  7$: MOV      RMMR2I,$BDDAT    ;;VERIFY RMMR2
102 050574 042737 140000 001142  BIC      #RQA!RQB,$BDDAT
103 050602 012737 011777 001140  MOV      #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
104 050610 023737 001140 001142  CMP      $GDDAT,$BDDAT
105 050616 001413          BEQ      8$              ;;BRANCH IF ZERO
106 050620 062716 000004          ADD      #4,(SP)          ;;MOVE SP TO USER'S ERROR CALL
107 050624 112776 000136 000000  MOVVB   #136,a(SP)        ;;WRITE ERROR NUMBER IN CALL
108 050632 162716 000002          SUB      #2,(SP)          ;;MOVE SP TO RETURN FOR ERROR
109 050636 004736          JSR      PC,a(SP)+        ;;REPORT ERROR VIA USER
110 050640 162716 000010          SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
111 050644 000240          NOP
112                                ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC
113 050646 005037 001140          8$: CLR      $GDDAT          ;;EXPECT ALL ZEROS
114 050652 013737 001400 001142  MOV      RMER2I,$BDDAT    ;;VERIFY RMER2
    
```


115	050660	042737	040200	001142	BIC	#SKI!DVC,\$BDDAT	:IGNORE DEVICE ERRORS
116	050666	001413			BEQ	9\$:BRANCH IF OTHER BITS 0
117	050670	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
118	050674	112776	000174	000000	MOVB	#174,@(SP)	:WRITE ERROR NUMBER IN CALL
119	050702	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
120	050706	004736			JSR	PC,@(SP)+	:REPORT ERROR VIA USER
121	050710	162716	000010		SLB	#10,(SP)	:MOVE SP BACK TO NO ERROR
122	050714	000240			NOP		
123							
124	050716	013737	001350	001142	:REPORT	ERROR IF RMD5 NOT INITIALIZED	
125	050724	042737	177177	001142	9\$:	MOV RMD5I,\$BDDAT	:TEST DRIVE STATUS REGISTER
126	050732	012737	000600	001140	BIC	#*C<DRY!DPR>,\$BDDAT	
127	050740	023737	001140	001142	MOV	#DPR!DRY,\$GDDAT	:EXPECTED DRIVE STATUS
128	050746	001413			CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED & RECEIVED
129	050750	062716	000004		BEQ	10\$:BRANCH IF EQUAL
130	050754	112776	000134	000000	ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
131	050762	162716	000002		MOVB	#134,@(SP)	:WRITE ERROR NUMBER
132	050766	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
133	050770	162716	000010		JSR	PC,@(SP)+	:REPORT ERROR TO USER
134	050774	000240			SUB	#10,(SP)	:MOVE SP BACK TO NO ERROR
135	050776	062716	000004		NOP		
136	051002	105776	000000		10\$:	ADD #4,(SP)	:MOVE SP TO ERROR CALL
137	051006	001403			TSTB	@(SP)	:WAS AN ERROE DETECTED??
138	051010	062716	000004		BEQ	11\$:NO!!
139	051014	000402			ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
140	051016	162716	000004		BR	12\$	
141	051022	000240			11\$:	SUB #4,(SP)	:MOVE SP TO NO ERROR RETURN
142	051024	000207			12\$:	NOP	
					RTS	PC	

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

.SBTTL PACK ACKNOWLEDGE STATUS CHECK

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

```

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

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

ACKSTS:

```

:CLEAR USER'S ERROR CALL
ADD #4,(SP)
CLRB a(SP)
SUB #4,(SP)
    
```

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

```

:REPORT AN ERROR IF 'VV' IS 0
BIT #VV,RMDSI
BNE 1$
MOV RMDSI,$GDDAT
BIS #VV,$GDDAT
MOV RMDSI,$BDDAT
ADD #4,(SP)
MOVB #155,a(SP)
SUB #2,(SP)
JSR PC,a(SP)+
SUB #10,(SP)
NOP
    
```

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

1\$:

```

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

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

2\$:

```

:SEE IF 'UNS','OPI','RMR','ILR', OR 'ILF' IS SET
BIT #UNS!OPI!RMR!ILR!ILF,RMER1I
BEQ 7$
    
```

```

:REPORT AN ERROR IF 'UNS' IS SET
BIT #UNS,RMER1I
BEQ 3$
MOV RMER1I,$BDDAT
    
```

:WAS UNS SET??
 :NO!!
 :RECEIVED STATUS

```

051026
051026 062716 000004
051032 105076 000000
051036 162716 000004
051042 032737 000100 001350
051050 001024
051052 013737 001350 001140
051060 052737 000100 001140
051066 013737 001350 001142
051074 062716 000004
051100 112776 000155 000000
051106 162716 000002
051112 004736
051114 162716 000010
051120 000240
051122
051122 032737 010000 001350
051130 001024
051132 013737 001350 001140
051140 052737 010000 001140
051146 013737 001350 001142
051154 062716 000004
051160 112776 000041 000000
051166 162716 000002
051172 004736
051174 162716 000010
051200 000240
051202
051202 032737 060007 001352
051210 001570
051212 032737 040000 001352
051220 001424
051222 013737 001352 001142
    
```


58	051230	013737	001352	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
59	051236	042737	040000	001140	BIC	#UNS,\$GDDAT	
60	051244	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
61	051250	112776	000042	000000	MOVB	#4,@(SP)	:WRITE NUMBER OF ERROR IN CALL
62	051256	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
63	051262	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
64	051264	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
65	051270	000240			NOP		
66	051272						
67							
68							
69	051272	032737	020000	001352	:REPORT	ANY OPI ERROR	
70	051300	001424			BIT	#OPI,RMER11	:WAS OPI SET ??
71	051302	013737	001352	001142	BEQ	4\$:NO!!
72	051310	013737	001352	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
73	051316	042737	020000	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
74	051324	062716	000004		BIC	#OPI,\$GDDAT	
75	051330	112776	000043	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
76	051336	162716	000002		MOVB	#4,@(SP)	:WRITE NUMBER OF ERROR IN CALL
77	051342	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
78	051344	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
79	051350	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
80	051352				NOP		
81							
82							
83	051352	032737	000004	001352	:REPORT	ANY RMR ERROR	
84	051360	001424			BIT	#RMR,RMER11	:WAS RMR SET??
85	051362	013737	001352	001142	BEQ	5\$:NO!!
86	051370	013737	001352	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
87	051376	042737	000004	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
88	051404	062716	000004		BIC	#RMR,\$GDDAT	
89	051410	112776	000044	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
90	051416	162716	000002		MOVB	#4,@(SP)	:WRITE NUMBER OF ERROR IN CALL
91	051422	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
92	051424	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
93	051430	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
94	051432				NOP		
95							
96							
97	051432	032737	000002	001352	:REPORT	ANY ILR ERROR	
98	051440	001424			BIT	#ILR,RMER11	:WAS ILR SET??
99	051442	013737	001352	001142	BEQ	6\$:NO!!
100	051450	013737	001352	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
101	051456	042737	000002	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
102	051464	062716	000004		BIC	#ILR,\$GDDAT	
103	051470	112776	000045	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
104	051476	162716	000002		MOVB	#4,@(SP)	:WRITE NUMBER OF ERROR IN CALL
105	051502	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
106	051504	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
107	051510	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
108	051512				NOP		
109							
110							
111	051512	032737	000001	001352	:REPORT	ANY ILF ERROR	
112	051520	001424			BIT	#ILF,RMER11	:WAS ILF SET??
113	051522	013737	001352	001142	BEQ	7\$:NO!!
114	051530	013737	001352	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
					MOV	RMER11,\$GDDAT	:EXPECTED STATUS

115	051536	042737	000001	001140	BIC	#ILF,\$GDDAT	
116	051544	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
117	051550	112776	000046	000000	MOVB	#46,@(SP)	:WRITE NUMBER OF ERROR IN CALL
118	051556	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
119	051562	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
120	051564	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
121	051570	000240			NOP		
122	051572						
123							
124							
125	051572	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
126	051576	105776	000000		TSTB	@(SP)	:WAS ERROR FOUND??
127	051602	001403			BEQ	8\$:NO!!
128	051604	062716	000004		ADD	#4,(SP)	:YES - MOVE TO ERROR RETURN
129	051610	000402			BR	9\$	
130	051612	162716	000004		SUB	#4,(SP)	:MOVE SP TO NO ERROR RETURN
131	051616	000240			NOP		
132	051620	000207			RTS	PC	

7\$:

:AUGMENT RETURN ADDRESS IF ERROR WAS FOUND

8\$:

9\$:

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

.SBTTL RECALIBRATE STATUS CHECK SUBROUTINE

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

:CALL:

```

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

RCLSTS:

```

: CLEAR USER'S ERROR NUMBER
  ADD #4,(SP)
  CLR  a(SP) ;CLEAR USER'S ERROR CALL
  SUB #4,(SP) ;MOVE SP BACK TO BRANCH
    
```

```

:SEE IF 'PAR' OR 'ILF' OR 'OPI' OR 'IAE' IS SET
  BIT #OPI!PAR!ILF!IAE,RMER1I
  BEQ 5$ ;NONE ARE SET - GO TO NEXT CHECK
    
```

:REPORT ANY MASSBUS CONTROL BUS PARITY ERROR, I.E.,
 :'PAR' = 1 AND 'DPE' = 0

```

  BIT #PAR,RMER1I ;WAS 'PAR' SET??
  BEQ 1$ ;NO!!
  BIT #DPE,RMER2I ;WAS 'DPE' SET??
  BNE 1$ ;YES - NOT A REGISTER ERROR
  MOV RMER1I,$GDDAT ;EXPECTED STATUS
  BIC #PAR,$GDDAT
  MOV RMER1I,$BDDAT ;RECEIVED STATUS
  ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
  MOVB #50,a(SP) ;WRITE ERROR NUMBER IN CALL
  SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
  JSR PC,a(SP)+ ;GO REPORT ERROR
  SUB #10,(SP) ;MOVE SP BACK TO BRANCH
  NOP
    
```

1\$:

```

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

2\$:

```

15 051622
18 051622 062716 000004
19 051626 105076 000000
20 051632 162716 000004
24 051636 032737 022011 001352
25 051644 001553
29 051646 032737 000010 001352
30 051654 001430
31 051656 032737 000010 001400
32 051664 001024
33 051666 013737 001352 001140
34 051674 042737 000010 001140
35 051702 013737 001352 001142
36 051710 062716 000004
37 051714 112776 000050 000000
38 051722 162716 000002
39 051726 004736
40 051730 162716 000010
41 051734 000240
42 051736
45 051736 032737 000001 001352
46 051744 001424
47 051746 013737 001352 001140
48 051754 042737 000001 001140
49 051762 013737 001352 001142
50 051770 062716 000004
51 051774 112776 000071 000000
52 052002 162716 000002
53 052006 004736
54 052010 162716 000010
55 052014 000240
56 052016
    
```



```

58      ;REPORT ANY "OPI" ERROR AS
59      . OPI DUE TO "MOL" = 0
60      . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
61 052016 032737 020000 001352      BIT      #OPI,RMER1I      ;WAS OPI SET??
62 052024 001433                    BEQ      4$              ;NO!!
63 052026 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
64 052034 042737 020000 001140      BIC      #OPI,$GDDAT
65 052042 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
66 052050 062716 000004              ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
67 052054 112776 000072 000000      MOVB     #72,a(SP)       ;WRITE ERROR NUMBER IN USER'S CALL
68 052062 032737 010000 001350      BIT      #MOL,RMDSI      ;WAS "MOL" = 0??
69 052070 001403                    BEQ      3$              ;YES!!
70 052072 112776 000073 000000      MOVB     #73,a(SP)       ;NO - CHANGE ERROR NUMBER
71 052100 162716 000002 3$:        SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
72 052104 004736                    JSR      PC,a(SP)+        ;REPORT ERROR VIA USER
73 052106 162716 000010              SUB      #10,(SP)         ;MOVE SP BACK TO BRANCH
74 052112 000240                    NOP
75 052114                    4$:
76
77      ;REPORT AN ERROR IF "IAE" IS SET
78 052114 032737 002000 001352      BIT      #IAE,RMER1I     ;IS "IAE" SET??
79 052122 001424                    BEQ      5$              ;NO!!
80 052124 013737 001352 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
81 052132 042737 002000 001140      BIC      #IAE,$GDDAT
82 052140 013737 001352 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
83 052146 062716 000004              ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
84 052152 112776 000070 000000      MOVB     #70,a(SP)       ;WRITE ERROR NUMBER IN USER'S CALL
85 052160 162716 000002              SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
86 052164 004736                    JSR      PC,a(SP)+        ;REPORT ERROR
87 052166 162716 000010              SUB      #10,(SP)         ;MOVE SP BACK TO NO ERROR RETURN
88 052172 000240                    NOP
89 052174                    5$:
90
91      ;SEE IF "SKI" OR "IVC" OR "DVC" IS SET
92 052174 032737 050200 001400      BIT      #SKI!IVC!DVC,RMER2I
93 052202 001517                    BEQ      9$              ;NONE OF THE BITS ARE SET
94
95
96      ;REPORT ANY "IVC" ERROR AS
97      . IVC WITH VV = 0
98      . ERRONEOUS IVC ERROR
99 052204 032737 010000 001400      BIT      #IVC,RMER2I     ;WAS IVC SET??
100 052212 001433                    BEQ      7$              ;NO!!
101 052214 013737 001400 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
102 052222 042737 010000 001140      BIC      #IVC,$GDDAT
103 052230 013737 001400 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
104 052236 062716 000004              ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
105 052242 112776 000074 000000      MOVB     #74,a(SP)       ;WRITE ERROR NUMBER IN CALL
106 052250 032737 000100 001350      BIT      #VV,RMDSI       ;WAS VV = 0??
107 052256 001403                    BEQ      6$              ;YES!!
108 052260 112776 000075 000000      MOVB     #75,a(SP)       ;NO - CHANGE ERROR NUMBER
109 052266 162716 000002 6$:        SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
110 052272 004736                    JSR      PC,a(SP)+        ;REPORT ERROR VIA USER
111 052274 162716 000010              SUB      #10,(SP)         ;MOVE SP BACK TO BRANCH
112 052300 000240                    NOP
113 052302                    7$:
114

```



```

115 ;REPORT ANY "SKI" ERROR
116 052302 032737 040000 001400 BIT #SKI,RMER2I ;WAS SKI SET??
117 052310 001424 BEQ 8$ ;NO!!
118 052312 013737 001400 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
119 052320 042737 040000 001140 BIC #SKI,$GDDAT
120 052326 013737 001400 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
121 052334 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
122 052340 112776 000076 000000 MOVB #76,@(SP) ;WRITE ERROR NUMBER
123 052346 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
124 052352 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
125 052354 162716 000010 SUB #10,(SP) ;MOVE SP TO BRANCH
126 052360 000240 NOP
127 052362
128
129 ;REPORT ANY "DVC" ERROR
130 052362 032737 000200 001400 BIT #DVC,RMER2I ;WAS "DVC" SET??
131 052370 001424 BEQ 9$ ;NO!!
132 052372 013737 001400 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
133 052400 042737 000200 001140 BIC #DVC,$GDDAT
134 052406 013737 001400 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
135 052414 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
136 052420 112776 000077 000000 MOVB #77,@(SP) ;WRITE ERROR NUMBER
137 052426 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
138 052432 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
139 052434 162716 000010 SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
140 052440 000240 NOP
141 052442
142
143 ;SEE IF "PIP" AND "OM" ARE 0, AND "ATA","MOL" AND "VV" ARE 1
144 052442 013746 001350 MOV RMDSI,-(SP) ;PUT RMDS ON STACK
145 052446 042716 047676 BIC #^C<PIP!MOL!VV!OM!ATA>,(SP)
146 052452 022726 110100 CMP #ATA!MOL!VV,(SP)+
147 052456 001002 BNE 10$
148 052460 000137 053074 JMP 15$
149 052464
150
151 ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
152 ;LINE AFTER RECALIBRATE WAS INITIATED
153 052464 032737 010000 001350 BIT #MOL,RMDSI ;DID MOL DROP??
154 052472 001030 BNE 11$ ;NO!!
155 052474 032737 020000 001352 BIT #OPI,RMER1I ;WAS OPI ERROR REPORTED??
156 052502 001024 BNE 11$ ;YES - DON'T REPORT MOL=0
157 052504 013737 001350 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
158 052512 052737 010000 001140 BIS #MOL,$GDDAT
159 052520 013737 001350 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
160 052526 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
161 052532 112776 000100 000000 MOVB #100,@(SP) ;WRITE ERROR NUMBER
162 052540 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
163 052544 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
164 052546 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
165 052552 000240 NOP
166 052554
167
168 ;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0
169 052554 032737 000100 001350 BIT #VV,RMDSI ;DID "VV" DROP??
170 052562 001030 BNE 12$ ;NO!!
171 052564 032737 010000 001400 BIT #IVC,RMER2I ;WAS THERE A IVC ERROR??
    
```

172	052572	001024			BNE	12\$:YES - DONT REPORT VV=0
173	052574	013737	001350	001140	MOV	RMDSI,\$GDDAT	:EXPECTED STATUS
174	052602	013737	001350	001142	MOV	RMDSI,\$BDDAT	:RECEIVED STATUS
175	052610	052737	000100	001140	BIS	#VV,\$GDDAT	
176	052616	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
177	052622	112776	000101	000000	MOVB	#101,a(SP)	:WRITE ERROR NUMBER IN CALL
178	052630	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
179	052634	004736			JSR	PC,a(SP)+	
180	052636	162716	000010		SUB	#10,(SP)	:MOVE SP BACK TO USER'S BRANCH
181	052642	000240			NOP		
182	052644						
183							
184							
185	052644	032737	100000	001350			:REPORT AN ERROR IF ATA IS NOT SET
186	052652	001024			BIT	#ATA,RMDSI	:WAS ATA SET DURING RECALIBRATE??
187	052654	013737	001350	001140	BNE	13\$:YES!!
188	052662	052737	100000	001140	MOV	RMDSI,\$GDDAT	:EXPECTED STATUS
189	052670	013737	001350	001142	BIS	#ATA,\$GDDAT	
190	052676	062716	000004		MOV	RMDSI,\$BDDAT	:RECEIVED STATUS
191	052702	112776	000102	000000	ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
192	052710	162716	000002		MOVB	#102,a(SP)	:WRITE ERROR NUMBER IN CALL
193	052714	004736			SUB	#2,(SP)	
194	052716	162716	000010		JSR	PC,a(SP)+	
195	052722	000240			SUB	#10,(SP)	:MOVE SP TO USER'S BRANCH
196					NOP		
197	052724						
198							
199							
200							
201	052724	032737	000001	001350			:REPORT AN ERROR IF 'OM' IS NOT ZERO BECAUSE RECALIBRATE SHOULD
202	052732	001424			:ALWAYS	CLEAR OFFSET MODE	
203	052734	013737	001350	001140	BIT	#OM,RMDSI	:WAS 'OM' RESET??
204	052742	042737	000001	001140	BEQ	14\$:YES!!
205	052750	013737	001350	001142	MOV	RMDSI,\$GDDAT	:EXPECTED STATUS
206	052756	062716	000004		BIC	#OM,\$GDDAT	
207	052762	112776	000103	000000	MOV	RMDSI,\$BDDAT	:RECEIVED STATUS
208	052770	162716	000002		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
209	052774	004736			MOVB	#103,a(SP)	:WRITE ERROR NUMBER
210	052776	162716	000010		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
211	053002	000240			JSR	PC,a(SP)+	:REPORT ERROR VIA USER
212	053004				SUB	#10,(SP)	:MOVE SP TO USER'S BRANCH
213					NOP		
214							
215							
216	053004	032737	020000	001350			:REPORT AN ERROR IF 'PIP' IS STIL ON, I.E., DRIVE NOT ON
217	053012	001430			:CYLINDER		
218	053014	032737	040000	001400	BIT	#PIP,RMDSI	:IS DRIVE OFF CYLINDER??
219	053022	001024			BEQ	15\$:NO!!
220	053024	013737	001350	001140	BIT	#SKI,RMER2I	:WAS 'SKI' DETECTED??
221	053032	042737	020000	001140	BNE	15\$:YES-DONT REPORT 'PIP'
222	053040	013737	001350	001142	MOV	RMDSI,\$GDDAT	:EXPECTED STATUS
223	053046	062716	000004		BIC	#PIP,\$GDDAT	
224	053052	112776	000104	000000	MOV	RMDSI,\$BDDAT	:RECEIVED STATUS
225	053060	162716	000002		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
226	053064	004736			MOVB	#104,a(SP)	:WRITE ERROR NUMBER
227	053066	162716	000010		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
228	053072	000240			JSR	PC,a(SP)+	
					SUB	#10,(SP)	:MOVE SP BACK TO USER'S BRANCH
					NOP		


```

229 053074      15$:
230
231           ;SEE IF "ILR" OR "RMR" OR "UNS" IS SET
232 053074 032737 040006 001352  BIT    #ILR!RMR!UNS,RMER1I
233 053102 001514           BEQ    18$
234
235           ;REPORT AN ERROR IF "ILR" IS SET
236 053104 032737 000002 001352  BIT    #ILR,RMER1I      ;WAS ILR SET DURING RECALIBRATE??
237 053112 001424           BEQ    16$              ;NO!!
238 053114 013737 001352 001140  MOV    RMER1I,$GDDAT    ;EXPECTED STATUS
239 053122 042737 000002 001140  BIC    #ILR,$GDDAT
240 053130 013737 001352 001142  MOV    RMER1I,$BDDAT    ;RECEIVED STATUS
241 053136 062716 000004           ADD    #4,(SP) ;MOVE SP TO USER'S ERROR CALL
242 053142 112776 000105 000000  MOVB   #105,@(SP)      ;WRITE ERROR NUMBER IN CALL
243 053150 162716 000002           SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
244 053154 004736           JSR    PC,@(SP)+
245 053156 162716 000010           SUB    #10,(SP)        ;MOVE SP TO USER'S BRANCH
246 053162 000240           NOP
247 053164
248
249           16$:
250           ;REPORT AN ERROR IF "RMR" IS SET
251 053164 032737 000004 001352  BIT    #RMR,RMER1I      ;WAS RMR SET??
252 053172 001424           BEQ    17$              ;NO!!
253 053174 013737 001352 001140  MOV    RMER1I,$GDDAT    ;EXPECTED STATUS
254 053202 042737 000004 001140  BIC    #RMR,$GDDAT
255 053210 013737 001352 001142  MOV    RMER1I,$BDDAT    ;RECEIVED STATUS
256 053216 062716 000004           ADD    #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
257 053222 112776 000106 000000  MOVB   #106,@(SP)      ;WRITE ERROR NUMBER IN USER'S CALL
258 053230 162716 000002           SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
259 053234 004736           JSR    PC,@(SP)+
260 053236 162716 000010           SUB    #10,(SP)        ;REPORT ERROR VIA USER
261 053242 000240           NOP                    ;MOVE SP TO USER'S BRANCH
262 053244
263           17$:
264           ;REPORT AN ERROR IF "UNS" IS SET AND "DVC" IS 0
265 053244 032737 040000 001352  BIT    #UNS,RMER1I      ;WAS UNSAFE ON??
266 053252 001430           BEQ    18$              ;NO!!
267 053254 032737 000200 001400  BIT    #DVC,RMER2I      ;WAS THERE A DEVICE CHECK??
268 053262 001024           BNE    18$              ;YES - DON'T REPORT UNSAFE
269 053264 013737 001352 001140  MOV    RMER1I,$GDDAT    ;EXPECTED STATUS
270 053272 042737 040000 001140  BIC    #UNS,$GDDAT
271 053300 013737 001352 001142  MOV    RMER1I,$BDDAT    ;RECEIVED STATUS
272 053306 062716 000004           ADD    #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
273 053312 112776 000107 000000  MOVB   #107,@(SP)      ;WRITE ERROR NUMBER
274 053320 162716 000002           SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
275 053324 004736           JSR    PC,@(SP)+
276 053326 162716 000010           SUB    #10,(SP)        ;REPORT ERROR VIA USER
277 053332 000240           NOP                    ;MOVE SP BACK TO USER'S BRANCH
278 053334
279           18$:
280           ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
281 053334 062716 000004           ADD    #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
282 053340 105776 000000           TSTB   @(SP)           ;WAS AN ERROR REPORTED??
283 053344 001403           BEQ    19$              ;NO!!
284 053346 062716 000004           ADD    #4,(SP)         ;YES - AUGMENT SP RETURN
285 053352 000402           BR     20$
285 053354 162716 000004           19$: SUB    #4,(SP) ;NO ERROR - RETURN SP TO BRANCH
    
```


286 053360 000240
287 053362 000207

20\$: 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,a(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
7      :      ???      RETURN HERE IF NO MORE ERRORS
8
9      053364  DRVSTS:
10
11     :CLEAR USER'S ERROR CALL
12     053364 062716 000004  ADD #4,(SP)      ;MOVE SP TO ERROR CALL
13     053370 105076 000000  CLRB a(SP)      ;CLEAR ERROR CALL
14     053374 162716 000004  SUB #4,(SP)      ;MOVE SP TO USER'S BRANCH
15
16     053400 013737 001336 001142 1$:  MOV RMCS1I,$BDDAT ;CHECK RMCS1
17     053406 042737 173700 001142  BIC #^C<DVA!FNCMSK>,$BDDAT ;CLEAR DONT CARES
18     053414 012737 004010 001140  MOV #DVA!DRVCLR,$GDDAT ;EXPECT DVA
19     053422 023737 001140 001142  CMP $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
20     053430 001443  BEQ 3$ ;BRANCH IF EQUAL
21     053432 062716 000004  ADD #4,(SP)      ;MOVE SP TO ERROR CALL
22     053436 112776 000141 000000  MOVB #141,a(SP) ;WRITE NUMBER OF ERROR IN CALL
23     053444 162716 000002  SUB #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
24     053450 004736  JSR PC,a(SP)+   ;REPORT THE ERROR VIA USER
25     053452 162716 000010  SUB #10,(SP)     ;MOVE SP TO NO ERROR RETURN
26     053456 000240  NOP
27
28     053460 013737 001350 001142 2$:  MOV RMDSI,$BDDAT ;CHECK RMDS
29     053466 042737 021101 001142  BIC #PGM!OM!VV!PIP,$BDDAT ;CLEAR DONT CARES
30     053474 012737 010600 001140  MOV #MOL!DPR!DRY,$GDDAT ;EXPECT DRY & DPR
31     053502 023737 001140 001142  CMP $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
32     053510 001413  BEQ 3$ ;BRANCH IF EQUAL
33     053512 062716 000004  ADD #4,(SP)      ;MOVE SP TO ERROR CALL
34     053516 112776 000142 000000  MOVB #142,a(SP) ;WRITE NUMBER OF ERROR IN CALL
35     053524 162716 000002  SUB #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
36     053530 004736  JSR PC,a(SP)+   ;REPORT THE ERROR VIA USER
37     053532 162716 000010  SUB #10,(SP)     ;MOVE SP TO NO ERROR RETURN
38     053536 000240  NOP
39
40     053540 005037 001140 3$:  CLR $GDDAT ;EXPECT 0'S
41     053544 013737 001352 001142  MOV RMER1I,$BDDAT ;CHECK RMER1
42     053552 001413  BEQ 4$ ;BRANCH IF EQUAL
43     053554 062716 000004  ADD #4,(SP)      ;MOVE SP TO ERROR CALL
44     053560 112776 000143 000000  MOVB #143,a(SP) ;WRITE NUMBER OF ERROR IN CALL
45     053566 162716 000002  SUB #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
46     053572 004736  JSR PC,a(SP)+   ;REPORT THE ERROR VIA USER
47     053574 162716 000010  SUB #10,(SP)     ;MOVE SP TO NO ERROR RETURN
48     053600 000240  NOP
49
50     053602 013737 001354 001142 4$:  MOV RMA5I,$BDDAT ;CHECK ATTENTION BIT
51     053610 010146  MOV R1,-(SP) ;:PUSH R1 ON STACK
52     053612 010246  MOV R2,-(SP) ;:PUSH R2 ON STACK
53     053614 013701 001466  MOV TSTQUE,R1
54     053620 116102 000001  MOVB 1(R1),R2
55     053624 042702 177400  BIC #^CATNMSK,R2
56     053630 005102  COM R2
57     053632 040237 001142  BIC R2,$BDDAT
    
```



```

58 053636 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
59 053640 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
60 053642 005737 001142  TST      $BDDAT      ;;IS ATTENTION CLEARED??
61 053646 001413      BEQ      5$          ;;BRANCH IF ATTENTION CLEARED
62 053650 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
63 053654 112776 000144 000000  MOVB     #144,a(SP)   ;;WRITE NUMBER OF ERROR IN CALL
64 053662 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
65 053666 004736      JSR     PC,a(SP)+    ;;REPORT THE ERROR VIA USER
66 053670 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
67 053674 000240      NOP
68
69 053676 013737 001362 001142 5$:      :REPORT ERROR IF RMMR1 NOT INITIALIZED
70 053704 042737 000046 001142  MOV      RMMR1,$BDDAT  ;;CHECK RMMR
71 053712 012737 000010 001140  BIC      #WC!LS!LST,$BDDAT ;;CLEAR DONT CARES
72 053720 023737 001140 001142  MOV      #MWD,$GDDAT  ;;EXPECT WRITE DATA ON
73 053726 001413      CMP      $GDDAT,$BDDAT ;;COMPARE EXPECTED AND RECEIVED
74 053730 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
75 053734 112776 000145 000000  MOVB     #145,a(SP)   ;;WRITE NUMBER OF ERROR IN CALL
76 053742 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
77 053746 004736      JSR     PC,a(SP)+    ;;REPORT THE ERROR VIA USER
78 053750 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
79 053754 000240      NOP
80
81 053756 013737 001376 001142 6$:      :REPORT ERROR IF RMMR2 NOT INITIALIZED
82 053764 042737 140000 001142  MOV      RMMR2,$BDDAT  ;;CHECK RMMR2
83 053772 012737 011777 001140  BIC      #RQA!ROB,$BDDAT ;;CLEAR REQA, REQB
84 054000 023737 001140 001142  MOV      #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
85 054006 001413      CMP      $GDDAT,$BDDAT ;;COMPARE EXPECTED & RECEIVED
86 054010 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
87 054014 112776 000146 000000  MOVB     #146,a(SP)   ;;WRITE NUMBER OF ERROR IN CALL
88 054022 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
89 054026 004736      JSR     PC,a(SP)+    ;;REPORT THE ERROR VIA USER
90 054030 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
91 054034 000240      NOP
92 054036 005037 001140 7$:      CLR      $GDDAT      ;;EXPECT ZEROS
93
94 054042 013737 001404 001142 8$:      :REPORT ERROR IF RMEC2 NOT RESET
95 054050 001413      MOV      RMEC2,$BDDAT ;;CHECK RMEC2
96 054052 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
97 054056 112776 000150 000000  MOVB     #150,a(SP)   ;;WRITE NUMBER OF ERROR IN CALL
98 054064 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
99 054070 004736      JSR     PC,a(SP)+    ;;REPORT THE ERROR VIA USER
100 054072 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
101 054076 000240      NOP
102
103 054100 013737 001400 001142 9$:      :REPORT ERROR IF RMER2 NOT RESET
104 054106 001413      MOV      RMER2,$BDDAT ;;CHECK RMER2
105 054110 062716 000004  ADD      #4,(SP)      ;;MOVE SP TO ERROR CALL
106 054114 112776 000147 000000  MOVB     #147,a(SP)   ;;WRITE NUMBER OF ERROR IN CALL
107 054122 162716 000002      SUB      #2,(SP)      ;;MOVE SP TO RETURN FOR ERROR
108 054126 004736      JSR     PC,a(SP)+    ;;REPORT THE ERROR VIA USER
109 054130 162716 000010      SUB      #10,(SP)     ;;MOVE SP TO NO ERROR RETURN
110 054134 000240      NOP
111 054136
112
113 054136
114

```


115									
116	054136	062716	000004						
117	054142	105776	000000						
118	054146	001403							
119	054150	062716	000004						
120	054154	000402							
121	054156	162716	000004	11\$:	SUB	#4,(SP)			
122	054162	000240		12\$:	NOP				
123	054164	000207			RTS	PC			

;AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND

ADD #4,(SP) ;MOVE SP TO ERROR CALL

TSTB @(\$P) ;WAS AN ERROR DETECTED??

BEQ 11\$;NO!!

ADD #4,(SP) ;YES - MOVE SP TO ERROR RETURN

BR 12\$

SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN

NOP

RTS PC ;RETURN TO USER

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

.SBTTL DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

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

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

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

DTASTS:

:CLEAR USER'S ERROR CALL AND ERROR FLAGS

ADD #4,(SP) :MOVE SP TO USER'S ERROR
 CLRB @(SP) :CLEAR LOW ORDER BYTE OF TRAP
 SUB #4,(SP) :RESTORE SP TO NO ERROR
 CLR 49\$:CLEAR ERROR FLAGS

:REPORT ANY CONTROL BUS PARITY ERROR WHILE READING REMOTE REGISTERS,

:I.E., MCPE = 1

BIT #MCPE,RMCS1I :WAS THERE A PARITY ERROR??
 BEQ 1\$:NO!!
 MOV RMCS1I,\$GDDAT :EXPECTED STATUS
 BIC #MCPE,\$GDDAT
 MOV RMCS1I,\$BDDAT :RECEIVED STATUS
 ADD #4,(SP) :MOVE SP TO USER'S ERROR CALL
 MOVB #13,@(SP) :WRITE ERROR NUMBER
 SUB #2,(SP) :MOVE SP TO RETURN IF ERROR
 JSR PC,@(SP)+ :REPORT ERROR AND RETURN
 BR 4\$

1\$:

:REPORT ANY CONTROL BUS PARITY ERROR WHILE WRITTING REMOTE REGISTERS,

:I.E., PAR = 1 AND DPE = 0

BIT #PAR,RMER1I :WAS THERE A PARITY ERROR??
 BEQ 3\$:NO!!
 BIT #DPE,RMER2I :DATA PARITY ERROR ?
 BNE 3\$:YES!!
 MOV RMER1I,\$GDDAT :EXPECTED STATUS
 BIC #PAR,\$GDDAT
 MOV RMER1I,\$BDDAT :RECEIVED STATUS
 ADD #4,(SP) :MOVE SP TO USER'S ERROR
 MOVB #50,@(SP) :WRITE ERROR NUMBER
 BIT #MXF,RMCS2I :DID MXF GET SET??
 BNE 2\$:YES!!
 MOVB #274,@(SP) :NO - CHANGE ERROR NUMBER
 SUB #2,(SP) :MOVE SP TO RETURN IF ERROR
 JSR PC,@(SP)+ :REPORT ERROR AND RETURN
 BR 4\$

2\$:

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


```

58      :MECHANICAL POSITIONING
59
60      :FIRST TEST MXF WHICH WOULD INDICATE COMPOSITE ERROR SET WHEN FUNCTION
61      :CODE AND GO BIT WERE LOADED
62      054364      032737      001000      001346      3$:      BIT      #MXF,RMCS2I      :WAS 'MISSED TRANSFER' SET??
63      054364      001425
64      054372      001425      BEQ      5$      :NO!!
65      054374      013737      001346      001140      MOV      RMCS2I,$GDDAT      :EXPECTED STATUS
66      054402      042737      001000      001140      BIC      #MXF,$GDDAT
67      054410      013737      001346      001142      MOV      RMCS2I,$BDDAT      :RECEIVED STATUS
68      054416      062716      000004      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
69      054422      112776      000275      000000      MOVB     #275,a(SP)      :WRITE ERROR NUMBER
70      054430      162716      000002      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
71      054434      004736      JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
72      054436
73
74      :RESTORE SP TO NO ERROR RETURN AND BYPASS FURHTER STATUS CHECKING
75      054436      162716      000010      SUB      #10,(SP)      :MOVE SP TO NO ERROR
76      054442      000137      057554      JMP      46$      :SKIP TO END OF SUB
77
78      054446      5$:
79
80      :REPORT AN ERROR IF 'OPI' ERROR OCCURRED DUE TO 'MOL' = 0, OR IF 'OPI'
81      :AND 'MOL' ARE SET, BUT 'VV' IS RESET, INDICATING AN INTERMITTENT
82      :'MOL'
83      054446      032737      020000      001352      BIT      #OPI,RMER1I      :IS 'OPI' SET??
84      054454      001447      BEQ      8$      :NO!!
85      054456      013737      001352      001140      MOV      RMER1I,$GDDAT      :EXPECTED STATUS
86      054464      042737      020000      001140      BIC      #OPI,$GDDAT
87      054472      013737      001352      001142      MOV      RMER1I,$BDDAT      :RECEIVED STATUS
88      054500      032737      010000      001350      BIT      #MOL,RMDSI      :WAS MEDIUM OFF LINE??
89      054506      001404      BEQ      6$      :YES!!
90      054510      032737      000100      001350      BIT      #VV,RMDSI      :WAS 'MOL' INTERMITTENT??
91      054516      001013      BNE      7$      :NO!!
92      054520      062716      000004      6$:      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
93      054524      112776      000276      000000      MOVB     #276,a(SP)      :WRITE ERROR NUMBER IN CALL
94      054532      162716      000002      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
95      054536      004736      JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
96      054540      162716      000010      SUB      #10,(SP)      :RESTORE SP TO NO ERROR
97      054544      000413      BR       8$
98      054546
99
100     :REPORT 'OPI' ERROR, WHICH IS DUE TO 'ON CYLINDER' NOT DROPPING OR
101     :'RUN' TIMEOUT (20 MS) OR SEARCH TIMEOUT (50 MS)
102     054546      062716      000004      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
103     054552      112776      000277      000000      MOVB     #277,a(SP)      :WRITE ERROR NUMBER IN CALL
104     054560      162716      000002      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
105     054564      004736      JSR      PC,a(SP)+      :REPORT ERROR AND RETURN
106     054566      162716      000010      SUB      #10,(SP)      :RESTORE SP TO NO ERROR
107     054572      000240      NOP
108     054574
109
110     :LOOK FOR 'IVC' ERROR DURING COMMAND INITIATION
111     054574      032737      010000      001400      BIT      #IVC,RMER2I      :WAS THERE AN 'IVC' ERROR??
112     054602      001432      BEQ      10$      :NO!!
113     :REPORT 'IVC' ERROR DUE TO 'VV' = 0, OR REPORT ERRONEOUS 'IVC' ERROR
114     054604      013737      001400      001140      MOV      RMER2I,$GDDAT      :EXPECTED STATUS
    
```



```

115 054612 042737 010000 001140      BIC      #IVC,$GDDAT
116 054620 013737 001400 001142      MOV      RMER2I,$BDDAT      :RECEIVED STATUS
117 054626 062716 000004                ADD      #4,(SP)           :MOVE SP TO USER'S ERROR
118 054632 112776 000300 000000      MOVVB   #300,@(SP)       :WRITE ERROR NUMBER IN CALL
119 054640 032737 000100 001350      BIT      #VV,RMDSI       :WAS VOLUME VALID??
120 054646 001403                BEQ      9$              :NO!!
121 054650 112776 000301 000000      MOVVB   #301,@(SP)       :CHANGE ERROR NUMBER
122 054656 162716 000002      9$:     SUB      #2,(SP)     :MOVE SP TO RETURN IF ERROR
123 054662 004736                JSR      PC,@(SP)+       :REPORT "IVC" ERROR AND RETURN
124 054664 162716 000010                SUB      #10,(SP)       :RESTORE SP TO NO ERROR
125 054670                10$:
126
127      ;SEE IF "ILF" OR "RMR" IS SET
128 054670 032737 000007 001352      BIT      #ILR!ILF!RMR,RMER1I
129 054676 001510                BEQ      13$             :NO ERRORS DETECTED
130      ;REPORT AN ERROR IF "ILR" IS SET
131 054700 032737 000002 001352      BIT      #ILR,RMER1I    :WAS "ILR" DETECTED??
132 054706 001424                BEQ      11$             :NO!!
133 054710 013737 001352 001140      MOV      RMER1I,$GDDAT   :EXPECTED STATUS
134 054716 042737 000002 001140      BIC      #ILR,$GDDAT
135 054724 013737 001352 001142      MOV      RMER1I,$BDDAT   :RECEIVED STATUS
136 054732 062716 000004                ADD      #4,(SP)           :MOVE SP TO USER'S ERROR CALL
137 054736 112776 000302 000000      MOVVB   #302,@(SP)       :WRITE ERROR NUMBER IN CALL
138 054744 162716 000002                SUB      #2,(SP)           :MOVE SP TO RETURN IF ERROR
139 054750 004736                JSR      PC,@(SP)+       :REPORT ERROR AND RETURN
140 054752 162716 000010                SUB      #10,(SP)       :RESTORE SP TO NO ERROR
141 054756 000240                NOP
142 054760                11$:
143
144      ;REPORT AN ERROR IF "ILF" IS SET
145 054760 032737 000001 001352      BIT      #ILF,RMER1I    :WAS "ILF" DETECTED??
146 054766 001424                BEQ      12$             :NO!!
147 054770 013737 001352 001140      MOV      RMER1I,$GDDAT   :EXPECTED STATUS
148 054776 042737 000001 001140      BIC      #ILF,$GDDAT
149 055004 013737 001352 001142      MOV      RMER1I,$BDDAT   :RECEIVED STATUS
150 055012 062716 000004                ADD      #4,(SP)           :MOVE SP TO USER'S ERROR CALL
151 055016 112776 000303 000000      MOVVB   #303,@(SP)       :WRITE ERROR NUMBER IN CALL
152 055024 162716 000002                SUB      #2,(SP)           :MOVE SP TO RETURN IF ERROR
153 055030 004736                JSR      PC,@(SP)+       :REPORT ERROR AND RETURN
154 055032 162716 000010                SUB      #10,(SP)       :RESTORE SP TO NO ERROR
155 055036 000240                NOP
156 055040                12$:
157
158      ;REPORT AN ERROR IF "RMR" IS SET
159 055040 032737 000004 001352      BIT      #RMR,RMER1I    :WAS "RMR" DETECTED??
160 055046 001424                BEQ      13$             :NO!!
161 055050 013737 001352 001140      MOV      RMER1I,$GDDAT   :EXPECTED STATUS
162 055056 042737 000004 001140      BIC      #RMR,$GDDAT
163 055064 013737 001352 001142      MOV      RMER1I,$BDDAT   :RECEIVED STATUS
164 055072 062716 000004                ADD      #4,(SP)           :MOVE SP TO USER'S ERROR CALL
165 055076 112776 000304 000000      MOVVB   #304,@(SP)       :WRITE ERROR NUMBER IN CALL
166 055104 162716 000002                SUB      #2,(SP)           :MOVE SP TO RETURN IF ERROR
167 055110 004736                JSR      PC,@(SP)+       :REPORT ERROR AND RETURN
168 055112 162716 000010                SUB      #10,(SP)       :RESTORE SP TO NO ERROR
169 055116 000240                NOP
170 055120                13$:
171      ;DETERMINE WHETHER OR NOT "IAE" SHOULD BE SET AND CHECK FOR ERROR
    
```

172	055120	012737	002000	001140	MOV	#IAE,\$GDDAT	:SETUP FOR "IAE" = 1
173	055126	052737	040000	057602	BIS	#SKI,49\$:SETUP FOR "SKI" = 1
174	055134	023727	001446	001060	CMP	RMDC0,#560.	:GREATER THAN LAST CYLINDER ?
175	055142	101035			BHI	15\$:YES - CYLINDER IS INVALID
176	055144	042737	040000	057602	BIC	#SKI,49\$:RESET SKI FLAG
177							
178	055152	123737	001421	001335	CMPB	RMDAO+1,LSTRK+1	:GREATER THAN LAST TRACK ?
179	055160	101026			BHI	15\$:YES - TRACK IS INVALID
180							
181	055162	123727	001420	000035	CMPB	RMDAO,#29.	:IS SECTOR > 29. ?
182	055170	101420			BLOS	14\$:NO
183	055172	032737	010000	001444	BIT	#FMT16,RMOFO	:18 BIT FORMAT ?
184	055200	001416			BEQ	15\$:YES - SECTOR IS INVALID FOR 18 BIT MODE
185	055202	123727	001420	000036	CMPB	RMDAO,#30.	:IS SECTOR > 30. ?
186	055210	101410			BLOS	14\$:NO
187	055212	032737	001000	001444	BIT	#SSEI,RMOFO	:IS SSEI CLEAR ?
188	055220	001406			BEQ	15\$:YES - SECTOR IS INVALID FOR 16 BIT MODE
189	055222	123727	001420	000037	CMPB	RMDAO,#31.	:IS SECTOR > 31. ?
190	055230	101002			BHI	15\$:YES - SECTOR IS INVALID
191	055232	005037	001140		14\$: CLR	\$GDDAT	: "IAE" SHOULD = 0
192							
193	055236	013737	001352	001142	15\$: MOV	RMER1I,\$BDDAT	:GET RECEIVED STATUS
194	055244	042737	175777	001142	BIC	#^CIAE,\$BDDAT	
195	055252	023737	001140	001142	CMP	\$GDDAT,\$BDDAT	:IS "IAE" STATUS OK??
196	055260	001004			BNE	16\$:NO!!
197	055262	042737	040000	057602	BIC	#SKI,49\$:IAE OK - SKI SHOULD BE 0
198	055270	000412			BR	17\$	
199	055272	062716	000004		16\$: ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
200	055276	112776	000305	000000	MOVB	#305,a(SP)	:WRITE ERROR NUMBER
201	055304	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN IF ERROR
202	055310	004736			JSR	PC,a(SP)+	:REPORT ERROR AND RETURN
203	055312	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR
204	055316				17\$:		
205							
206							:REPORT AN ERROR IF "SKI" IS SET AND "IAE" STATUS WAS OK
207	055316	013737	001400	001142	MOV	RMER2I,\$BDDAT	:RECEIVED STATUS
208	055324	042737	137777	001142	BIC	#^CSKI,\$BDDAT	
209	055332	013737	057602	001140	MOV	49\$,\$GDDAT	:EXPECTED STATUS
210	055340	042737	137777	001140	BIC	#^CSKI,\$GDDAT	
211	055346	032737	040000	001400	BIT	#SKI,RMER2I	:WAS "SKI" SET??
212	055354	001417			BEQ	18\$:NO!!
213	055356	032737	040000	057602	BIT	#SKI,49\$:WAS SKI CAUSED BY IAE = 0??
214	055364	001032			BNE	19\$:YES - DON'T REPORT SKI
215	055366	062716	000004		ADD	#4,(SP)	:MOVE SP TO USERS ERROR CALL
216	055372	112776	000306	000000	MOVB	#306,a(SP)	:WRITE ERROR NUMBER
217	055400	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN IF ERROR
218	055404	004736			JSR	PC,a(SP)+	:REPORT ERROR AND RETURN
219	055406	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR
220	055412	000417			BR	19\$	
221							
222	055414				18\$:		
223							
224							:REPORT AN ERROR IF SKI = 0 AND IAE WAS NOT DETECTED
225	055414	032737	040000	057602	BIT	#SKI,49\$:SHOULD SKI BE SET??
226	055422	001413			BEQ	19\$:NO!!
227	055424	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
228	055430	112776	000307	000000	MOVB	#307,a(SP)	:WRITE ERROR NUMBER IN CALL


```

286 055716 032737 040000 001352 BIT #UNS,RMER1I ;IS 'UNS' SET??
287 055724 001427 BEQ 23$ ;NO!!
288 055726 032737 000200 001400 BIT #DVC,RMER2I ;WAS 'UNS' CAUSED BY 'DVC'??
289 055734 001023 BNE 23$ ;YES!!
290 055736 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
291 055744 042737 040000 001140 BIC #UNS,$GDDAT
292 055752 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
293 055760 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
294 055764 112776 000313 000000 MOVB #313,a(SP) ;WRITE ERROR NUMBER
295 055772 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
296 055776 004736 JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
297 056000 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
298 056004
299
300 ;REPORT ANY DRIVE TIMING ERROR, I.E., 'DTE' = 1
301 056004 032737 010000 001352 BIT #DTE,RMER1I ;IS DTE SET??
302 056012 001423 BEQ 24$ ;NO!!
303 056014 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
304 056022 042737 010000 001140 BIC #DTE,$GDDAT
305 056030 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
306 056036 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
307 056042 112776 000314 000000 MOVB #314,a(SP) ;WRITE ERROR NUMBER IN CALL
308 056050 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
309 056054 004736 JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
310 056056 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
311 056062
312
313 ;REPORT AN ERROR IF WRITE LOCK ERROR IS SET. SEE IF DRIVE IS NOT
314 ;WRITE PROTECTED, OR IF FUNCTION WAS NOT A WRITE
315 056062 032737 004000 001352 BIT #WLE,RMER1I ;WAS 'WLE' SET??
316 056070 001441 BEQ 27$ ;NO!!
317 056072 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
318 056100 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
319 056106 052737 004000 001140 BIS #WLE,$GDDAT
320 056114 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
321 056120 112776 000315 000000 MOVB #315,a(SP) ;WRITE ERROR NUMBER IN CALL
322 056126 032737 004000 001350 BIT #WRL,RMDSI ;WAS DRIVE WRITE PROTECTED??
323 056134 001404 BEQ 25$ ;NO!!
324 056136 032737 000010 001412 BIT #BIT3,RMCS10 ;WAS COMMAND A WRITE??
325 056144 001406 BEQ 26$ ;YES!!
326 056146 112776 000316 000000 MOVB #316,a(SP) ;CHANGE ERROR NUMBER
327 056154 042737 004000 001140 BIC #WLE,$GDDAT
328 056162 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
329 056166 004736 JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
330 056170 162716 000010 SUB #10,(SP) ;MOVE SP TO NO ERROR
331
332 056174
333
334 ;OMIT DATA ERROR CHECKS IF ANY PREVIOUS ERRORS HAVE BEEN DETECTED
335 056174 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
336 056200 105776 000000 TSTB a(SP) ;WAS ERROR DETECTED??
337 056204 001404 BEQ 28$ ;NO - DO DATA CHECKS
338 056206 162716 000004 SUB #4,(SP) ;RESTORE SP
339 056212 000137 057214 JMP 42$ ;SKIP DATA CHECKS
340 056216 162716 000004 SUB #4,(SP) ;RESTORE SP
341
342 ;CHECK HEADER ERRORS IF FUNCTION WAS NOT WRITE HEADER AND DATA, AND
    
```

```

343 ;IF HEADER COMPARE IS NOT INHIBITED
344 056222 013737 001412 057604 MOV RMCS10,50$ ;STRIP AND STORE FUNCTION CODE
345 056230 042737 177700 057604 BIC #^CFNCMSK,50$
346 056236 022737 000063 057604 CMP #WH!GO,50$ ;WAS FUNCTION WRITE HEADER & DATA??
347 056244 001512 BEQ 31$ ;YES - SKIP HEADER CHECKS
348 056246 032737 002000 001370 BIT #HCI,RMOFI ;WAS HCI SET??
349 056254 001106 BNE 31$ ;YES - SKIP HEADER CHECKS
350
351 ;SEE IF ANY HEADER ERRORS ARE SET, I.E., 'FER' OR 'HCRC' OR 'HCE'
352 056256 032737 000620 001352 BIT #HCRC!FER!HCE,RMER1I
353 056264 001533 BEQ 33$ ;NO ERRORS SET
354
355 ;REPORT HEADER CRC ERROR IF SET
356 056266 032737 000400 001352 BIT #HCRC,RMER1I ;WAS HCRC SET??
357 056274 001422 BEQ 29$ ;NO!!
358 056276 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
359 056304 042737 000400 001140 BIC #HCRC,$GDDAT
360 056312 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
361 056320 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR
362 056324 112776 000317 000000 MOVB #317,a(SP) ;WRITE ERROR NUMBER
363 056332 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
364 056336 004736 JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
365 056340 000501 BR 32$
366 056342 29$:
367
368 ;REPORT FORMAT ERROR IF SET
369 056342 032737 000020 001352 BIT #FER,RMER1I ;WAS 'FER' SET??
370 056350 001422 BEQ 30$ ;NO!!
371 056352 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
372 056360 042737 000020 001140 BIC #FER,$GDDAT
373 056366 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
374 056374 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR
375 056400 112776 000320 000000 MOVB #320,a(SP) ;WRITE ERROR NUMBER
376 056406 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
377 056412 004736 JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
378 056414 000453 BR 32$
379 056416 30$:
380
381 ;REPORT HEADER COMPARE ERROR IF SET
382 056416 032737 000200 001352 BIT #HCE,RMER1I ;WAS 'HCE' SET??
383 056424 001453 BEQ 33$ ;NO!!
384 056426 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
385 056434 042737 000200 001140 BIC #HCE,$GDDAT
386 056442 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
387 056450 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
388 056454 112776 000321 000000 MOVB #321,a(SP) ;WRITE ERROR NUMBER
389 056462 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
390 056466 004736 JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
391 056470 000425 BR 32$
392 ;THERE SHOULD BE NO HEADER ERRORS BECAUSE
393 ; .COMMAND WAS WRITE HEADER AND DATA, OR
394 ; .HEADER COMPARE INHIBIT WAS SET
395 056472 032737 000620 001352 31$: BIT #HCE!FER!HCRC,RMER1I
396 056500 001425 BEQ 33$ ;NO ERRORS WERE SET
397 056502 013737 001352 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
398 056510 042737 000620 001140 BIC #HCE!FER!HCRC,$GDDAT
399 056516 013737 001352 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
    
```



```

400 056524 062716 000004          ADD    #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
401 056530 112776 000322 000000    MOVB   #322,@(SP)      ;WRITE ERROR NUMBER
402 056536 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN IF ERROR
403 056542 004736 000000          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
404 056544 162716 000010          SUB    #10,(SP)        ;MOVE SP TO NO ERROR
405 056550 000137 057214          JMP    42$             ;OMIT FURTHER DATA CHECKS
406
407 056554          33$:
408
409          ;IF COMMAND WAS A WRITE COMMAND, GO DO WRITE ERROR CHECKS, OTHERWISE
410          ;DO READ ERROR CHECKS
411 056554 032737 000010 057604        BIT    #BIT3,50$       ;WAS THIS A WRITE COMMAND?
412 056562 001002          BNE   34$             ;NO!!
413 056564 000137 057002          JMP    39$             ;GO DO WRITE STATUS CHECK
414 056570          34$:
415
416          ;REPORT DATA CHECK IF SET
417 056570 032737 100000 001352        BIT    #DCK,RMER1I     ;DATA CHECK ERROR??
418 056576 001450          BEQ   37$             ;NO!!
419 056600 013737 001352 001140        MOV    RMER1I,$GDDAT   ;EXPECTED STATUS
420 056606 042737 100000 001140        BIC    #DCK,$GDDAT
421 056614 013737 001352 001142        MOV    RMER1I,$BDDAT   ;RECEIVED STATUS
422 056622 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USER'S ERROR
423 056626 112776 000323 000000        MOVB   #323,@(SP)      ;WRITE ERROR NUMBER
424 056634 032737 004000 001370        BIT    #ECI,RMOFI     ;WAS ECC CORRECTION DISABLED??
425 056642 001021          BNE   36$             ;YES!!
426 056644 112776 000324 000000        MOVB   #324,@(SP)      ;CHANGE TO RECOVERABLE ERROR
427 056652 032737 000100 001352        BIT    #ECH,RMER1I     ;IS ERROR RECOVERABLE??
428 056660 001007          BNE   35$             ;NO !!
429          ;DO NOT REPORT RECOVERABLE ERROR IF READ COMMAND
430 056662 032737 000020 057604        BIT    #BIT4,50$       ;WAS THIS A READ COMMAND ??
431 056670 001406          BEQ   36$             ;NO !!
432 056672 162716 000004          SUB    #4,(SP)         ;RESTORE SP
433 056676 000410          BR    37$             ;SKIP ERROR - DATA WILL BE CORRECTED
434 056700 112776 000325 000000        35$: MOVB   #325,@(SP)      ;CHANGE TO NON RECOVERABLE
435 056706 162716 000002          36$: SUB    #2,(SP)         ;MOVE SP TO RETURN IF ERROR
436 056712 004736 000010          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
437 056714 162716 000010          SUB    #10,(SP)        ;RESTORE SP TO NO ERROR
438
439 056720          37$:
440
441          ;REPORT DATA BUS PARITY ERROR IF SET, I.E., MDPE = 1
442 056720 032737 000400 001346        BIT    #MDPE,RMCS2I    ;PARITY ERROR SET??
443 056726 001423          BEQ   38$             ;NO!!
444 056730 013737 001346 001140        MOV    RMCS2I,$GDDAT   ;EXPECTED STATUS
445 056736 042737 000400 001140        BIC    #MDPE,$GDDAT
446 056744 013737 001346 001142        MOV    RMCS2I,$BDDAT   ;RECEIVED STATUS
447 056752 062716 000004          ADD    #4,(SP)         ;MOVE SP TO USER'S ERROR
448 056756 112776 000326 000000        MOVB   #326,@(SP)      ;WRITE ERROR NUMBER
449 056764 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN IF ERROR
450 056770 004736 000010          JSR    PC,@(SP)+       ;REPORT ERROR AND RETURN
451 056772 162716 000010          SUB    #10,(SP)        ;MOVE SP TO NO ERROR
452 056776 000137 057214          38$: JMP    42$             ;SKIP WRITE STATUS CHECK
453
454 057002          39$:
455
456          ;TEST TO SEE THAT OFFSET MODE WAS RESET; REPORT ERROR IF 'OM' = 1
    
```



```

457 057002 032737 000001 001350      BIT      #OM,RMDSI      :IS OFFSET ON??
458 057010 001423                      BEQ      40$          :NO
459 057012 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
460 057020 042737 000001 001140      BIC      #OM,$GDDAT
461 057026 013737 001350 001142      MOV      RMDSI,$BDDAT :RECEIVED STATUS
462 057034 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
463 057040 112776 000327 000000      MOVVB   #327,a(SP)   :WRITE ERROR NUMBER IN CALL
464 057046 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
465 057052 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
466 057054 162716 000010                      SUB      #10,(SP)    :MOVE SP TO NO ERROR
467 057060
468
469                                     ;TEST FOR DATA BUS PARITY ERROR: REPORT ERROR IF 'DPE' = 1
470 057060 032737 000010 001400      BIT      #DPE,RMER2I  :DATA PARITY ERROR??
471 057066 001423                      BEQ      41$          :NO!!
472 057070 013737 001400 001140      MOV      RMER2I,$GDDAT :EXPECTED STATUS
473 057076 042737 000010 001140      BIC      #DPE,$GDDAT
474 057104 013737 001400 001142      MOV      RMER2I,$BDDAT :RECEIVED STATUS
475 057112 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USER'S ERROR CALL
476 057116 112776 000330 000000      MOVVB   #330,a(SP)   :WRITE ERROR NUMBER
477 057124 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
478 057130 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
479 057132 162716 000010                      SUB      #10,(SP)    :MOVE SP TO NO ERROR
480 057136
481
482                                     ;TEST FOR WRITE CLOCK FAILURE: REPORT ERROR IF 'WCF' = 1
483 057136 032737 000040 001352      BIT      #WCF,RMER1I  :IS 'WCF' SET??
484 057144 001423                      BEQ      42$          :NO!!
485 057146 013737 001352 001140      MOV      RMER1I,$GDDAT :EXPECTED STATUS
486 057154 042737 000040 001140      BIC      #WCF,$GDDAT
487 057162 013737 001352 001142      MOV      RMER1I,$BDDAT :RECEIVED STATUS
488 057170 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
489 057174 112776 000331 000000      MOVVB   #331,a(SP)   :WRITE ERROR NUMBER
490 057202 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
491 057206 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
492 057210 162716 000010                      SUB      #10,(SP)    :MOVE SP TO NO ERROR
493 057214
494
495                                     ;REPORT 'DATA LATE' ERROR IF 'DLT' = 1
496 057214 032737 100000 001346      BIT      #DLT,RMCS2I  :IS 'DLT' SET??
497 057222 001423                      BEQ      43$          :NO!!
498 057224 013737 001346 001140      MOV      RMCS2I,$GDDAT :EXPECTED STATUS
499 057232 042737 100000 001140      BIC      #DLT,$GDDAT
500 057240 013737 001346 001142      MOV      RMCS2I,$BDDAT :RECEIVED STATUS
501 057246 062716 000004                      ADD      #4,(SP)      :MOVE SP TO USERS ERROR CALL
502 057252 112776 000332 000000      MOVVB   #332,a(SP)   :WRITE ERROR NUMBER
503 057260 162716 000002                      SUB      #2,(SP)      :MOVE SP TO RETURN IF ERROR
504 057264 004736                      JSR      PC,a(SP)+    :REPORT ERROR AND RETURN
505 057266 162716 000010                      SUB      #10,(SP)    :MOVE SP TO NO ERROR
506 057272
507                                     ;LOOK FOR UNEXPECTED CHANGES IN DRIVE STATUS
508 057272 013746 001350                      MOV      RMDSI,-(SP)  :STACK DRIVE STATUS
509 057276 042716 147677                      BIC      #^C<PIP!MOL!VV>,(SP) :CLEAR DONT CARES
510 057302 022726 010100                      CMP      #MOL!VV,(SP)+ :IS DRIVE STATUS OK??
511 057306 001522                      BEQ      46$          :YES!!
512
513                                     ;REPORT ERROR IF POSITIONING IN PROGRESS AND NO SEEK INCOMPLETE ERROR,
    
```

```

514                                     :I.E., PIP = 1 AND SKI = 0
515 057310 032737 020000 001350      BIT      #PIP,RMDSI      :IS 'PIP' SET??
516 057316 001430                                     :NO!!
517 057320 032737 040000 001400      BIT      #SKI,RMER2I   :WAS 'SKI' ERROR REPORTED??
518 057326 001024                                     :YES-DONT REPORT PIP
519 057330 013737 001350 001140      MOV      RMDSI,$GDDAT  :EXPECTED STATUS
520 057336 042737 020000 001140      BIC      #PIP,$GDDAT
521 057344 013737 001350 001142      MOV      RMDSI,$BDDAT :RECEIVED STATUS
522 057352 062716 000004                                     :MOVE SP TO USERS ERROR CALL
523 057356 112776 000333 000000      MOVVB   #333,a(SP)   :WRITE ERROR NUMBER
524 057364 162716 000002                                     :MOVE SP TO RETURN IF ERROR
525 057370 004736                                     :REPORT ERROR AND RETURN
526 057372 162716 000010      JSR     PC,a(SP)+    :MOVE SP TO NO ERROR
527 057376 000240      SUB     #10,(SP)
528 057400                                     :
529                                     :
530                                     :REPORT ERROR IF MEDIUM IS NOT ON LINE AND OPI ERROR WAS NOT
531                                     :REPORTED, I.E., MOL = OPI = 0
532 057400 032737 010000 001350      BIT      #MOL,RMDSI   :IS MEDIUM ON LINE??
533 057406 001027                                     :YES!!
534 057410 032737 020000 001352      BIT      #OPI,RMER1I  :WAS OPI ERROR REPORTED??
535 057416 001023                                     :YES!!
536 057420 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
537 057426 052737 010000 001140      BIS      #MOL,$GDDAT
538 057434 013737 001350 001142      MOV      RMDSI,$BDDAT :RECEIVED STATUS
539 057442 062716 000004                                     :MOVE SP TO USER'S ERROR
540 057446 112776 000334 000000      MOVVB   #334,a(SP)   :WRITE ERROR NUMBER
541 057454 162716 000002                                     :MOVE SP TO RETURN IF ERROR
542 057460 004736                                     :REPORT ERROR AND RETURN
543 057462 162716 000010      JSR     PC,a(SP)+    :MOVE SP TO NO ERROR
544 057466                                     :
545                                     :
546                                     :REPORT ERROR IF VOLUME IS NOT VALID AND 'IVC' ERROR WAS NOT
547                                     :REPORTED, I.E., VV = IVC = 0
548 057466 032737 000100 001350      BIT      #VV,RMDSI    :IS VOLUME VALID??
549 057474 001027                                     :YES!!
550 057476 032737 010000 001400      BIT      #IVC,RMER2I  :WAS IVC ERROR REPORTED??
551 057504 001033                                     :YES!!
552 057506 013737 001350 001140      MOV      RMDSI,$GDDAT :EXPECTED STATUS
553 057514 052737 000100 001140      BIS      #VV,$GDDAT
554 057522 013737 001350 001142      MOV      RMDSI,$BDDAT :RECEIVED STATUS
555 057530 062716 000004                                     :MOVE SP TO USERS ERROR CALL
556 057534 112776 000335 000000      MOVVB   #335,a(SP)   :WRITE ERROR NUMBER
557 057542 162716 000002                                     :MOVE SP TO RETURN IF ERROR
558 057546 004736                                     :REPORT ERROR AND RETURN
559 057550 162716 000010      JSR     PC,a(SP)+    :MOVE SP TO NO ERROR
560 057554                                     :
561                                     :
562                                     :AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS FOUND
563 057554 062716 000004      ADD     #4,(SP)      :MOVE SP TO ERROR CALL
564 057560 105776 000000      TSTB   a(SP)        :ANY ERROR??
565 057564 001403                                     :NO!!
566 057566 062716 000004      ADD     #4,(SP)      :YES - MOVE SP TO ERROR RETURN
567 057572 000402      BR     48$
568 057574 162716 000004      SUB     #4,(SP)      :MOVE SP TO NO ERROR RETURN
569                                     :
570 057600 000207      RTS     PC           :RETURN TO USER
    
```


571
572 057602 000000
573 057604 000000

49\$: .WORD
50\$: .WORD

:ERROR FLAGS
:TEMPORARY STORAGE

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

057606
 057606 062716 000004
 057612 105076 000000
 057616 162716 000004
 057622 013746 001350
 057626 042716 147677
 057632 022726 010100
 057636 001524
 057640 032737 010000 001350
 057646 001030
 057650 032737 020000 001352
 057656 001024
 057660 013737 001350 001140
 057666 052737 010000 001140
 057674 013737 001350 001142
 057702 062716 000004
 057706 112776 000207 000000
 057714 162716 000002
 057720 004736
 057722 162716 000010
 057726 000240
 057730
 057730 032737 000100 001350
 057736 001030

```

.SBTTL  STATIC DRIVE STATUS CHECK SUBROUTINE

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

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

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

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

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

STCDRVSTS:

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

:SEE IF 'MOL' = 'VV' = 1, AND 'PIP' = 0
MOV    RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
BIC    #'C<PIP!MOL!VV>,(SP)
CMP    #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
BEQ    3$      ;YES!!

:REPORT AN ERROR IF MOL = 0 AND 'OPI' = 0
BIT    #MOL,RMDSI ;IS MOL ON ??
BNE    1$      ;YES!!
BIT    #OPI,RMER1I ;WAS 'OPI' SET??
BNE    1$      ;YES-DONT REPORT 'MOL' = 0
MOV    RMDSI,$GDDAT ;EXPECTED STATUS
BIS    #MOL,$GDDAT
MOV    RMDSI,$BDDAT ;RECEIVED STATUS
ADD    #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB  #207,@(SP) ;WRITE ERROR NUMBER IN CALL
SUB    #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR    PC,@(SP)+ ;REPORT ERROR VIA USER
SUB    #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN

1$:

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

```

58 057740 032737 010000 001400      BIT      #IVC,RMER2I      :WAS "IVC" SET??
59 057746 001024                      BNE      2$              :YES-DONT REPORT "VV" = 0
60 057750 013737 001350 001140      MOV      RMDSI,$GDDAT    :EXPECTED STATUS
61 057756 052737 000100 001350      BIS      #VV,RMDSI
62 057764 013737 001350 001142      MOV      RMDSI,$BDDAT    :RECEIVED STATUS
63 057772 062716 000004                      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
64 057776 112776 000210 000000      MOVVB   #210,a(SP)      :WRITE ERROR NUMBER IN CALL
65 060004 162716 000002                      SUB      #2,(SP)         :MOVE SP TO RETURN FOR ERROR
66 060010 004736                      JSR      PC,a(SP)+       :REPORT ERROR VIA USER
67 060012 162716 000010                      SUB      #10,(SP)        :MOVE SP BACK TO NO ERROR
68 060016 000240
69 060020
70
71
72 060020 032737 020000 001350      ;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND "SKI" = 0
73 060026 001430                      BIT      #PIP,RMDSI      :IS DRIVE OFF CYLINDER??
74 060030 032737 040000 001400      BEQ      3$              :NO!!
75 060036 001024                      BIT      #SKI,RMER2I     :WAS "SKI" SET??
76 060040 013737 001350 001140      BNE      3$              :YES-DONT REPORT "PIP" = 1
77 060046 042737 020000 001140      MOV      RMDSI,$GDDAT    :EXPECTED STATUS
78 060054 013737 001350 001142      BIC      #PIP,$GDDAT
79 060062 062716 000004                      MOV      RMDSI,$BDDAT    :RECEIVED STATUS
80 060066 112776 000211 000000      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
81 060074 162716 000002                      MOVVB   #211,a(SP)      :WRITE ERROR NUMBER IN USER'S CALL
82 060100 004736                      SUB      #2,(SP)         :MOVE SP TO RETURN FOR ERROR
83 060102 162716 000010                      JSR      PC,a(SP)+       :REPORT ERROR VIA USER
84 060106 000240                      SUB      #10,(SP)        :MOVE SP TO NO ERROR RETURN
85 060110
86
87
88 060110 013746 001400                      ;SEE IF "SKI" = "DVC" = 0
89 060114 042726 137577                      MOV      RMER2I,-(SP)    :PUT ERROR REG 2 ON STACK
90 060120 001460                      BIC      #^C<SKI!DVC>,(SP)+
91 060122                      BEQ      6$              :BRANCH IF NO ERROR
92
93
94 060122 032737 000200 001400      ;REPORT AN ERROR IF THERE IS A DEVICE FAULT
95 060130 001424                      BIT      #DVC,RMER2I     :ANY DEVICE FAULT??
96 060132 013737 001400 001140      BEQ      5$              :NO!!
97 060140 042737 000200 001140      MOV      RMER2I,$GDDAT    :EXPECTED STATUS
98 060146 013737 001400 001142      BIC      #DVC,$GDDAT
99 060154 062716 000004                      MOV      RMER2I,$BDDAT    :RECEIVED STATUS
100 060160 112776 000212 000000      ADD      #4,(SP)         :MOVE SP TO USER'S CALL
101 060166 162716 000002                      MOVVB   #212,a(SP)      :WRITE NUMBER OF ERROR IN CALL
102 060172 004736                      SUB      #2,(SP)         :MOVE SP TO RETURN FOR ERROR
103 060174 162716 000010                      JSR      PC,a(SP)+       :REPORT ERROR VIA USER
104 060200 000240                      SUB      #10,(SP)        :MOVE SP BACK TO NO ERROR
105 060202
106
107
108 060202 032737 040000 001400      ;REPORT AN ERROR IF "SKI" = 1
109 060210 001424                      BIT      #SKI,RMER2I     :IS THERE A SEEK INCOMPLETE ERROR
110 060212 013737 001400 001140      BEQ      6$              :NO!!
111 060220 042737 040000 001140      MOV      RMER2I,$GDDAT    :EXPECTED STATUS
112 060226 013737 001400 001142      BIC      #SKI,$GDDAT
113 060234 062716 000004                      MOV      RMER2I,$BDDAT    :RECEIVED STATUS
114 060240 112776 000213 000000      ADD      #4,(SP)         :MOVE SP TO USER'S ERROR CALL
                                MOVVB   #213,a(SP)      :WRITE ERROR NUMBER IN USER'S ERROR CALL
    
```


115	060246	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
116	060252	004736			JSR	PC,@(SP)+	:REPORT ERROR VIA USER
117	060254	162716	000010		SUB	#10,(SP)	:MOVE SP BACK TO NO ERROR
118	060260	000240			NOP		
119	060262			6\$:			
120							
121							
122	060262	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
123	060266	105776	000000		TSTB	@(SP)	:WAS AN ERROR DETECTED??
124	060272	001403			BEQ	7\$:NO!!
125	060274	062716	000004		ADD	#4,(SP)	:YES - MOVE SP TO USER'S ERROR RETURN
126	060300	000402			BR	8\$	
127	060302	162716	000004	7\$:	SUB	#4,(SP)	:NO - MOVE SP TO NO ERROR RETURN
128	060306	000240		8\$:	NOP		
129	060310	000207			RTS	PC	:RETURN TO USER

1

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

060312
060312 010046
060314 010146
060316 010246
060320 010346
060322 010446
060324 010546
060326 016646 000022
060332 016646 000022
060336 016646 000022
060342 016646 000022
060346 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:
    
```

```

060350
060350 012666 000022
060354 012666 000022
060360 012666 000022
060364 012666 000022
060370 012605
060372 012604
060374 012603
060376 012602
060400 012601
060402 012600
060404 000002
    
```

```

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


.SBTTL BINARY TO ASCII AND TYPE ROUTINE

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

*CALL:

* MOV NUMBER,-(SP) ::NUMBER TO BE TYPED
 * TYPBN ::TYPE IT

060406	010146			\$TYPBN:	MOV	R1,-(SP)	::SAVE R1 ON THE STACK
060410	016601	000006			MOV	6(SP),R1	::GET THE INPUT NUMBER
060414	000261				SEC		::SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
060416	112737	000060	060460	1\$:	MOVB	#'0,\$BIN	::SET CHARACTER TO AN ASCII '0'.
060424	006101				ROL	R1	::GET THIS BIT
060426	001406				BEQ	2\$::DONE?
060430	105537	060460			ADCB	\$BIN	::NO--SET THE CHARACTER EQUAL TO THIS BIT
060434	104401	060460			TYPE	,\$BIN	::GO TYPE THIS BIT
060440	000241				CLC		::CLEAR 'C' SO CAN KEEP TRACK OF BITS
060442	000765				BR	1\$::GO DO THE NEXT BIT
060444	012601			2\$:	MOV	(SP)+,R1	::POP THE STACK INTO R1
060446	016666	000002	000004		MOV	2(SP),4(SP)	::ADJUST THE STACK
060454	012616				MOV	(SP)+,(SP)	
060456	000002				RTI		::RETURN TO USER
060460	000	000		\$BIN:	.BYTE	0,0	::STORAGE FOR ASCII CHAR. AND TERMINATOR

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

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

060462				\$TYPDS:	MOV	R0,-(SP)	;;PUSH R0 ON STACK
060462	010046				MOV	R1,-(SP)	;;PUSH R1 ON STACK
060464	010146				MOV	R2,-(SP)	;;PUSH R2 ON STACK
060466	010246				MOV	R3,-(SP)	;;PUSH R3 ON STACK
060470	010346				MOV	R5,-(SP)	;;PUSH R5 ON STACK
060472	010546				MOV	#20200,-(SP)	;;SET BLANK SWITCH AND SIGN
060474	012746	020200			MOV	20(SP),R5	;;GET THE INPUT NUMBER
060500	016605	000020			BPL	1\$;;BR IF INPUT IS POS.
060504	100004				NEG	R5	;;MAKE THE BINARY NUMBER POS.
060506	005405				MOVB	#'-,1(SP)	;;MAKE THE ASCII NUMBER NEG.
060510	112766	000055	000001	1\$:	CLR	R0	;;ZERO THE CONSTANTS INDEX
060516	005000				MOV	#SDBLK,R3	;;SETUP THE OUTPUT POINTER
060520	012703	060676			MOVB	#',(R3)+	;;SET THE FIRST CHARACTER TO A BLANK
060524	112723	000040		2\$:	CLR	R2	;;CLEAR THE BCD NUMBER
060530	005002				MOV	\$DTBL(R0),R1	;;GET THE CONSTANT
060532	016001	060666		3\$:	SUB	R1,R5	;;FORM THIS BCD DIGIT
060536	160105				BLT	4\$;;BR IF DONE
060540	002402				INC	R2	;;INCREASE THE BCD DIGIT BY 1
060542	005202				BR	3\$	
060544	000774			4\$:	ADD	R1,R5	;;ADD BACK THE CONSTANT
060546	060105				TST	R2	;;CHECK IF BCD DIGIT=0
060550	005702				BNE	5\$;;FALL THROUGH IF 0
060552	001002				TSTB	(SP)	;;STILL DOING LEADING 0'S?
060554	105716				BMI	7\$;;BR IF YES
060556	100407			5\$:	ASLB	(SP)	;;MSD?
060560	106316				BCC	6\$;;BR IF NO
060562	103003				MOVB	1(SP),-1(R3)	;;YES--SET THE SIGN
060564	116663	000001	177777	6\$:	BIS	#'0,R2	;;MAKE THE BCD DIGIT ASCII
060572	052702	000060		7\$:	BIS	#',R2	;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
060576	052702	000040			MOVB	R2,(R3)+	;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
060602	110223				TST	(R0)+	;;JUST INCREMENTING
060604	005720				CMP	R0,#10	;;CHECK THE TABLE INDEX
060606	020027	000010			BLT	2\$;;GO DO THE NEXT DIGIT
060612	002746				BGT	8\$;;GO TO EXIT
060614	003002				MOV	R5,R2	;;GET THE LSD
060616	010502				BR	6\$;;GO CHANGE TO ASCII
060620	000764			8\$:	TSTB	(SP)+	;;WAS THE LSD THE FIRST NON-ZERO?
060622	105726				BPL	9\$;;BR IF NO
060624	100003				MOVB	-1(SP),-2(R3)	;;YES--SET THE SIGN FOR TYPING
060626	116663	177777	177776	9\$:	CLRB	(R3)	;;SET THE TERMINATOR
060634	105013				MOV	(SP)+,R5	;;POP STACK INTO R5
060636	012605				MOV	(SP)+,R3	;;POP STACK INTO R3
060640	012603				MOV	(SP)+,R2	;;POP STACK INTO R2
060642	012602				MOV	(SP)+,R1	;;POP STACK INTO R1
060644	012601						

060646	012600			MOV	(SP)+,R0	::POP STACK INTO R0
060650	104401	060676		TYPE	\$DBLK	::NOW TYPE THE NUMBER
060654	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
060662	012616			MOV	(SP)+,(SP)	
060664	000002			RTI		::RETURN TO USER
060666	023420			\$DTBL:	10000.	
060670	001750				1000.	
060672	000144				100.	
060674	000012				10.	
060676				\$DBLK:	.BLKW 4	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

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

```

```

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

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

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

```

060706	017646	000000		\$TYPOS:	MOV	0(SP),-(SP)	::PICKUP THE MODE
060712	116637	000001	061131		MOVB	1(SP),SOFILL	::LOAD ZERO FILL SWITCH
060720	112637	061133			MOVB	(SP)+,SOMODE+1	::NUMBER OF DIGITS TO TYPE
060724	062716	000002			ADD	#2,(SP)	::ADJUST RETURN ADDRESS
060730	000406				BR	\$TYPON	
060732	112737	000001	061131	\$TYPOC:	MOVB	#1,SOFILL	::SET THE ZERO FILL SWITCH
060740	112737	000006	061133		MOVB	#6,SOMODE+1	::SET FOR SIX(6) DIGITS
060746	112737	000005	061130	\$TYPON:	MOVB	#5,SOCNT	::SET THE ITERATION COUNT
060754	010346				MOV	R3,-(SP)	::SAVE R3
060756	010446				MOV	R4,-(SP)	::SAVE R4
060760	010546				MOV	R5,-(SP)	::SAVE R5
060762	113704	061133			MOVB	SOMODE+1,R4	::GET THE NUMBER OF DIGITS TO TYPE
060766	005404				NEG	R4	
060770	062704	000006			ADD	#6,R4	::SUBTRACT IT FOR MAX. ALLOWED
060774	110437	061132			MOVB	R4,SOMODE	::SAVE IT FOR USE
061000	113704	061131			MOVB	SOFILL,R4	::GET THE ZERO FILL SWITCH
061004	016605	000012			MOV	12(SP),R5	::PICKUP THE INPUT NUMBER
061010	005003				CLR	R3	::CLEAR THE OUTPUT WORD
061012	006105			1\$:	ROL	R5	::ROTATE MSB INTO "C"
061014	000404				BR	3\$::GO DO MSB
061016	006105			2\$:	ROL	R5	::FORM THIS DIGIT
061020	006105				ROL	R5	
061022	006105				ROL	R5	
061024	010503				MOV	R5,R3	
061026	006103			3\$:	ROL	R3	::GET LSB OF THIS DIGIT
061030	105337	061132			DECB	SOMODE	::TYPE TH'S DIGIT?
061034	100016				BPL	7\$::BR IF "
061036	042703	177770			BIC	#177770,R3	::GET R OF JUNK
061042	001002				BNE	4\$::TEST FOR 0
061044	005704				TST	R4	::SUPPRESS THIS 0?
061046	001403				BEQ	5\$::BR IF YES
061050	005204			4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S

061052	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
061056	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
061062	110337	061126		MOVB	R3,8\$::SAVE FOR TYPING
061066	104401	061126		TYPE	,8\$::GO TYPE THIS DIGIT
061072	105337	061130	7\$:	DECB	\$OCNT	::COUNT BY 1
061076	003347			BGT	2\$::BR IF MORE TO DO
061100	002402			BLT	6\$::BR IF DONE
061102	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
061104	000744			BR	2\$::GO DO THE LAST DIGIT
061106	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
061110	012604			MOV	(SP)+,R4	::RESTORE R4
061112	012603			MOV	(SP)+,R3	::RESTORE R3
061114	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
061122	012616			MOV	(SP)+,(SP)	
061124	000002			RTI		::RETURN
061126	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
061127	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
061130	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
061131	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
061132	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

061134 105737 001173 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
061140 100002 BPL 1$ ;;BR IF YES
061142 000000 HALT ;;HALT HERE IF NO TERMINAL
061144 000430 BR 3$ ;;LEAVE
061146 010046 1$: MOV R0,-(SP) ;;SAVE R0
061150 017600 000002 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
061154 122737 000001 001242 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
061162 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
061164 132737 000100 001243 BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
061172 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
061174 010037 061204 MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
061200 004737 065546 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
061204 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
061206 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
061214 001003 BNE 60$ ;;YES,SKIP TYPE OUT
061216 112046 2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
061220 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
061222 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
061224 012600 60$: MOV (SP)+,R0 ;;RESTORE R0
061226 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
061232 000002 RTI ;;RETURN
061234 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
061240 001430 BEQ 8$
061242 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
061246 001006 BNE 5$
061250 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
061252 104401 TYPE ;;TYPE A CR AND LF
061254 001217 $CRLF
061256 105037 061464 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
061262 000755 BR 2$ ;;GET NEXT CHARACTER
061264 004737 061346 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
061270 123726 001172 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
061274 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
061276 013746 001170 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
061302 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
061306 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
061310 004737 061346 JSR PC,$TYPEC ;;GO TYPE A NULL
061314 105337 061464 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
061320 000770 BR 7$ ;;LOOP

```


;HORIZONTAL TAB PROCESSOR

061322	112716	000040		8\$:	MOVB	#' (SP)	::REPLACE TAB WITH SPACE
061326	004737	061346		9\$:	JSR	PC,\$TYPEC	::TYPE A SPACE
061332	132737	000007	061464		BITB	#7,\$CHARCNT	::BRANCH IF NOT AT
061340	001372				BNE	9\$::TAB STOP
061342	005726				TST	(SP)+	::POP SPACE OFF STACK
061344	000724				BR	2\$::GET NEXT CHARACTER
061346				\$TYPEC:			
061346	105777	117606			TSTB	@\$TKS	::CHAR IN KYBD BUFFER?
061352	100022				BPL	10\$::BR IF NOT
061354	017746	117602			MOV	@\$TKB, -(SP)	::GET CHAR
061360	042716	177600			BIC	#177600, (SP)	::STRIP EXTRANEIOUS BITS
061364	122716	000023			CMPB	#\$XOFF, (SP)	::WAS CHAR XOFF
061370	001012				BNE	102\$::BR IF NOT
061372				101\$:			
061372	105777	117562			TSTB	@\$TKS	::WAIT FOR CHAR
061376	100375				BPL	101\$	
061400	117716	117556			MOVB	@\$TKB, (SP)	::GET CHAR
061404	042716	177600			BIC	#177600, (SP)	::STRIP IT
061410	122716	000021			CMPB	#\$XON, (SP)	::WAS IT XON?
061414	001366				BNE	101\$::BR IF NOT
061416				102\$:			
061416	005726				TST	(SP)+	::FIX STACK
061420				10\$:			
061420	105777	117540			TSTB	@\$TPS	::WAIT UNTIL PRINTER IS READY
061424	100375				BPL	10\$	
061426	116677	000002	117532		MOVB	2(SP), @\$TPB	::LOAD CHAR TO BE TYPED INTO DATA REG.
061434	122766	000015	000002		CMPB	#CR, 2(SP)	::IS CHARACTER A CARRIAGE RETURN?
061442	001003				BNE	1\$::BRANCH IF NO
061444	105037	061464			CLRB	\$CHARCNT	::YES--CLEAR CHARACTER COUNT
061450	000406				BR	\$TYPEX	::EXIT
061452	122766	000012	000002	1\$:	CMPB	#LF, 2(SP)	::IS CHARACTER A LINE FEED?
061460	001402				BEQ	\$TYPEX	::BRANCH IF YES
061462	105227				INCB	(PC)+	::COUNT THE CHARACTER
061464	000000			\$CHARCNT:	WORD	0	::CHARACTER COUNT STORAGE
061466	000207			\$TYPEX:	RTS	PC	

.SBTTL SCOPE HANDLER ROUTINE

```

:*****
:*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
:*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
:*AND LOAD THE ERROR FLAG ($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
    
```

```

061470          $SCOPE:          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
061470 104410          JSR          PC,STOP
061472 004737 062222          1$:          BIT          #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
061476 032777 040000 117450          BEQ          9$          ;;NO IF SW14=0
061504 001402          JMP          $OVER          ;;JUMP OVER SCOPE ROUTINE
061506 000137 062136
061512
9$:
:#####START OF CODE FOR THE XOR TESTER#####
061512 000416          $XTSTR: BR          6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
:THIS INSTRUCTION TO A 'NOP' (NOP=240)
061514 013746 000004          MOV          @ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
061520 012737 061540 000004          MOV          #5$,@ERRVEC          ;;SET FOR TIMEOUT
061526 005737 177060          TST          @#177060          ;;TIME OUT ON XOR?
061532 012637 000004          MOV          (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
061536 000561          BR          $$VLAD          ;;GO TO THE NEXT TEST
061540 022626          5$:          CMP          (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
061542 012637 000004          MOV          (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
061546 000521          BR          7$          ;;LOOP ON THE PRESENT TEST
061550
6$:;#####END OF CODE FOR THE XOR TESTER#####
061550 032777 000400 117376          BIT          #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
061556 001421          BEQ          2$          ;;BR IF NO
061560 005046          CLR          -(SP)          ;;CLEAR A TEMP. LOCATION
061562 117716 117366          MOVB         @SWR,(SP)          ;;PICKUP THE DESIRED TEST NUMBER
061566 001414          BEQ          8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
061570 022716 000023          CMP          #23,(SP)          ;;CHECK THE NUMBER IN THE SWR
061574 002411          BLT          8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
061576 011637 001116          MOV          (SP),$TSTNM          ;;UPDATE THE TEST NUMBER
061602 005316          DEC          (SP)          ;;BACKUP BY ONE
061604 006316          ASL          (SP)          ;;SCALE THE TEST NUMBER AS AN INDEX
061606 062716 062154          ADD          #$$SW08TBL,(SP)          ;;FORM THE ADDRESS OF TEST POINTER
061612 013637 001122          MOV          @((SP)+,$LPADR)          ;;SET LOOP ADDRESS TO DESIRED TEST
061616 000547          BR          $OVER          ;;GO LOOP ON THE TEST
061620 005726          8$:          TST          (SP)+          ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
061622 105737 001117          2$:          TSTB         $ERFLG          ;;HAS AN ERROR OCCURRED?
061626 001502          BEQ          3$          ;;BR IF NO
061630 022737 177777 062640          CMP          #-1,CPSAVE          ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
061636 001455          BEQ          2003$          ;;KICK AROUND ROUTINE IF SO
061640 013746 000004          MOV          ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
061644 012737 061662 000004          MOV          #2000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
061652 013737 177766 062640          MOV          177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
061660 000406          BR          2001$
061662 012737 177777 062640 2000$:          MOV          #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
061670 012716 061676          MOV          #2001$,(SP)          ;;SETUP RETURN ADDRESS
    
```



```

061674 000002          RTI
061676 012637 000004    2001$: MOV      (SP)+,ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR

061702 022737 177777 062640 2002$: CMP      #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
061710 001430          BEQ      2003$          ;;BRANCH IF SO
061712 032737 000001 062640 BIT      #BIT00,CPSAVE  ;;SEE IF THE POWER MONITOR BIT IS ON
061720 001424          BEQ      2003$          ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
061722 042737 000001 177766 BIC      #BIT00,177766 ;;CLEAR THE BIT FOUND TO BE SET
061730 013746 001154    MOV      SWR,-(SP)      ;;SAVE SWR ADDRESS
061734 017646 000000    MOV      @ (SP),-(SP)   ;;SAVE SWR VALUE
061740 012737 000176 001154 MOV      #176,SWR      ;;GET SOFTWARE SWR ADDRESS
061746 011677 117202    MOV      (SP),@SWR     ;;GET CURRENT SWR VALUE
061752 042777 001000 117174 BIC      #BIT09,@SWR   ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
061760 104177          EMT      177          ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
061762 012676 000000    MOV      (SP)+,@(SP)   ;;RESTORE SWR TO ORIGINAL VALUE
061766 012637 001154    MOV      (SP)+,SWR     ;;RESTORE SWR ADDRESS
061772          2003$:
061772 123737 001131 001117 CMPB     SERMAX,SERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
062000 101015          BHI      3$            ;;BR IF NO
062002 032777 001000 117144 BIT      #BIT09,@SWR   ;;LOOP ON ERROR?
062010 001404          BEQ      4$            ;;BR IF NO
062012 013737 001124 001122 7$:  MOV      $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
062020 000446          BR       $OVER
062022 105037 001117 4$:  CLRB     SERFLG      ;;ZERO THE ERROR FLAG
062026 005037 001206    CLR      $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
062032 000415          BR       1$            ;;ESCAPE TO THE NEXT TEST
062034 032777 004000 117112 3$: BIT      #BIT11,@SWR   ;;INHIBIT ITERATIONS?
062042 001011          BNE     1$            ;;BR IF YES
062044 005737 001230    TST     $PASS        ;;IF FIRST PASS OF PROGRAM
062050 001406          BEQ     1$            ;;INHIBIT ITERATIONS
062052 005237 001120    INC     $ICNT        ;;INCREMENT ITERATION COUNT
062056 023737 001206 001120 CMP      $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
062064 002024          BGE     $OVER        ;;BR IF MORE ITERATION REQUIRED
062066 012737 000001 001120 1$:  MOV      #1,$ICNT     ;;REINITIALIZE THE ITERATION COUNTER
062074 013737 062152 001206 MOV      $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
062102 105237 001116 SSVLAD: INCB    $TSTNM   ;;COUNT TEST NUMBERS
062106 113737 001116 001226 MOVB    $TSTNM,$TESTN  ;;SET TEST NUMBER IN APT MAILBOX
062114 011637 001122    MOV     (SP),$LPADR   ;;SAVE SCOPE LOOP ADDRESS
062120 011637 001124    MOV     (SP),$LPERR   ;;SAVE ERROR LOOP ADDRESS
062124 005037 001210    CLR     $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
062130 112737 000001 001131 MOVB    #1,$SERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
062136 013777 001116 117012 SOVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
062144 013716 001122    MOV     $LPADR,(SP)  ;;FUDGE RETURN ADDRESS
062150 000002          RTI
062152 000005          $MXCNT: 5.          ;;FIXES PS
062154          $SSW08TBL:  ;;MAX. NUMBER OF ITERATIONS
          .REPT $TN-1
062154 000023          .WORD  TST1+2      ;;STARTING ADDRESS OF TEST 1
062156 010100          .WORD  TST2+2      ;;STARTING ADDRESS OF TEST 2
062160 011106          .WORD  TST3+2      ;;STARTING ADDRESS OF TEST 3
062162 012064          .WORD  TST4+2      ;;STARTING ADDRESS OF TEST 4
062164 013260          .WORD  TST5+2      ;;STARTING ADDRESS OF TEST 5
062166 014266          .WORD  TST6+2      ;;STARTING ADDRESS OF TEST 6
062170 015244          .WORD  TST7+2      ;;STARTING ADDRESS OF TEST 7
062172 016442          .WORD  TST10+2     ;;STARTING ADDRESS OF TEST 10
062174 017470          .WORD  TST11+2     ;;STARTING ADDRESS OF TEST 11
062176 020714          .WORD  TST12+2     ;;STARTING ADDRESS OF TEST 12

```


1

.SBTTL ERROR HANDLER ROUTINE

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO ERRYP 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
    
```

```

062250 105037 062642 $ERROR: CLR      IBSAVE      ;;CLEAR THE ITEM BYTE SAVE LOCATION
062254 104110          CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
062256 105237 001117 7$:      INCB      SERFLG      ;;SET THE ERROR FLAG
062262 001775          BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
062264 013777 001116 116664 MOV      $STSTM, @DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
062272 032777 002000 116654 BIT      #BIT10, @SWR      ;;BELL ON ERROR?
062300 001402          BEQ      1$      ;;NO - SKIP
062302 104401 001212          TYPE      .SBELL      ;;RING BELL
062306 005237 001126 1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
062312 011637 001132          MOV      (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
062316 162737 000002 001132 SUB      #2, $ERRPC
062324 117737 116602 001130 MOV      @ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
062332 032777 001000 116614 BIT      #BIT09, @SWR      ;;SEE IF LOOP ON ERROR IS SET
062340 001060          BNE      1004$      ;;BRANCH AROUND ROUTINE IF SO
062342 122737 000177 001130 CMP      #177, $ITEMB      ;;SEE IF THIS IS THE POWER FAIL CALL
062350 001454          BEQ      1004$      ;;BRANCH AROUND ROUTINE IF IT IS
062352 105737 062642          TST      IBSAVE      ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
062356 001047          BNE      1003$      ;;BRANCH IF SO
062360 022737 177777 062640 CMP      #-1, CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
062366 001445          BEQ      1004$      ;;BRANCH IF SO
062370 013746 000004          MOV      ERRVEC, -(SP) ;;SAVE CONTENTS OF ERROR VECTOR
062374 012737 062412 000004 MOV      #1000$, ERRVEC ;;SETUP 'TRAP' RETURN ADDRESS
062402 013737 177766 062640 MOV      177766, CPSAVE ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
062410 000406          BR      1001$
062412 012737 177777 062640 1000$: MOV      #-1, CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
062420 012716 062426          MOV      #1001$, (SP) ;;SETUP RETURN ADDRESS
062424 000002          RTI
062426 012637 000004 1001$: MOV      (SP)+, ERRVEC ;;RESTORE CONTENTS OF ERROR VECTOR

062432 022737 177777 062640 1002$: CMP      #-1, CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
062440 001420          BEQ      1004$      ;;BRANCH IF SO
062442 032737 000001 062640 BIT      #BIT00, CPSAVE      ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
062450 001414          BEQ      1004$      ;;BRANCH IF OK
062452 042737 000001 177766 BIC      #BIT00, 177766      ;;CLEAR THE BIT FOUND SET
062460 113737 001130 062642 MOV      $ITEMB, IBSAVE      ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
062466 112737 000177 001130 MOV      #177, $ITEMB      ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
062474 000402          BR      1004$      ;;BRANCH OVER IBSAVE CLEARING

062476 105037 062642 1003$: CLR      IBSAVE      ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
062502          1004$:
062502 032777 020000 116444 BIT      #BIT13, @SWR      ;;SKIP TYPEOUT IF SET
062510 001004          BNE      20$      ;;SKIP TYPEOUTS
062512 004737 062644          JSR      PC, ERRYP      ;;GO TO USER ERROR ROUTINE
    
```

062516	104401	001217		TYPE	,\$CRLF	
062522			20\$:			
062522	122737	000001	001242	CMPB	#APTENV,\$ENV	::RUNNING IN APT MODE
062530	001007			BNE	2\$::NO SKIP APT ERROR REPORT
062532	113737	001130	062544	MOVB	\$ITEMB,21\$::SET ITEM NUMBER AS ERROR NUMBER
062540	004737	065556		JSR	PC,\$ATY4	::REPORT FATAL ERROR TO APT
062544	000		21\$:	.BYTE	0	
062545	000			.BYTE	0	
062546	000777		22\$:	BR	22\$::APT ERROR LOOP
062550	105737	062642	2\$:	TSTB	IBSAVE	::SEE IF IBSAVE IS LOADED
062554	001005			BNE	3\$::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
062556	005777	116372		TST	@SWR	::HALT ON ERROR
062562	100002			BPL	3\$::SKIP IF CONTINUE
062564	000000			HALT		::HALT ON ERROR!
062566	104410			CKSWR		::TEST FOR CHANGE IN SOFT-SWR
062570			3\$:			
062570	032777	001000	116356	BIT	#BIT09,@SWR	::LOOP ON ERROR SWITCH SET?
062576	001402			BEQ	4\$::BR IF NO
062600	013716	001124		MOV	\$LPERR,(SP)	::FUDGE RETURN FOR LOOPING
062604	005737	001210	4\$:	TST	\$ESCAPE	::CHECK FOR AN ESCAPE ADDRESS
062610	001402			BEQ	5\$::BR IF NONE
062612	013716	001210		MOV	\$ESCAPE,(SP)	::FUDGE RETURN ADDRESS FOR ESCAPE
062616			5\$:			
062616	022737	033756	000042	CMP	#SENDAD,@#42	::ACT-11 AUTO-ACCEPT?
062624	001001			BNE	6\$::BRANCH IF NO
062626	000000			HALT		::YES
062630			6\$:			
062630	105737	062642		TSTB	IBSAVE	::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
062634	001210			BNE	7\$::BRANCH BACK TO CALL ORIGINAL ERROR
062636	000002			RTI		::RETURN
062640	000000			CPSAVE:	.WORD 0	::LOCATION TO SAVE CPU ERROR REG CONTENTS
062642	000000			IBSAVE:	.WORD 0	::LOCATION TO SAVE ITEM BYTE

.SBTTL ERROR TYPEOUT ROUTINE

```

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

2
3
4
5
6
7
8
9
10
11
12
13
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

```

062644 104414
062646 032777 020000 116300
062654 001402
062656 000137 063454

062662 104401 001217
062666 104401 063470
062672 013746 001234

062676 104403
062700 003
062701 000

062702 013700 001276
062706 016000 000026
062712 042700 177740
062716 012737 067062 062740
062724 022700 000026
062730 001004
062732 104401 063525
062736 104401
062740 000000

062742 005037 063460
062746 013737 001226 063460
062754 104401 063475
062760 013746 063460

062764 104403
062766 003
062767 000

062770 005037 063462
062774 113737 001130 063462
063002 001406
063004 104401 063505
063010 013746 063462

063014 104403
063016 003
063017 000
063020 104401 063514
    
```

```

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

:TYPE UNIT NUMBER, DRIVE TYPE, TEST NUMBER, ERROR NUMBER, AND
:PROGRAM COUNTER
1$: TYPE ,SCLF ;TYPE 'DRV#'
TYPE ,ERTY00 ;:SAVE $UNIT FOR TYPEOUT
MOV $UNIT,-(SP) ;:TYPE DRIVE NUMBER
;:GO TYPE--OCTAL ASCII
;:TYPE 3 DIGIT(S)
;:SUPPRESS LEADING ZEROS

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

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

```

46 063024 013746 001132      MOV      $ERRPC,-(SP)      ;;SAVE $ERRPC FOR TYPEOUT
                                ;;TYPE PROGRAM COUNTER
                                ;;GO TYPE--OCTAL ASCII
063030 104403      TYPOS
063032 006        .BYTE      6          ;;TYPE 6 DIGIT(S)
063033 001        .BYTE      1          ;;TYPE LEADING ZEROS
47
48      ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
49 063034 005737 063462      6$:      TST      ERRNMB      ;WAS AN ERROR CALLED?
50 063040 001002          BNE      7$          ;BR IF YES
51 063042 000137 063454          JMP      27$         ;NO--EXIT
52
53 063046 104401 001217      7$:      TYPE      ,SCRLF      ;YES-TYPE CRLF
54 063052 105037 063466          CLRB     BOTFLG      ;CLEAR BOT FLAG
55 063056 105037 063467          CLRB     CHRCNT      ;CLEAR CHARACTER COUNTER
56 063062 013700 063462          MOV      ERRNMB,R0   ;R0 POINTS TO FIRST OF
57 063066 122700 000177          CMPB    #177,R0      ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
58 063072 001003          BNE      8$          ;BRANCH IF NOT
59 063074 012700 063532          MOV      #PFECH,R0   ;MOVE POWER FAIL ERROR CALL TABLE TO R0
60 063100 000405          BR
61 063102 006300          8$:      ASL      R0          ;FOUR ENTRIES IN ERROR
62 063104 006300          ASL      R0          ;TABLE
63 063106 006300          ASL      R0
64 063110 062700 001574          ADD     #SERRTB-8.,R0
65 063114 011001          9$:      MOV      (R0),R1      ;R1 POINTS TO ERROR MESSAGE
66
67 063116 001507          BEQ      19$         ;BRANCH IF NO ERROR MESSAGE
68
69      ;TYPE THE ERROR MESSAGE
70 063120 012102          10$:     MOV      (R1)+,R2      ;R2=ADDRESS OF MESSAGE STRING
71 063122 001505          BEQ      19$         ;BRANCH IF END OF MESSAGE
72 063124 010237 063272          MOV      R2,18$      ;LOAD ADDRESS OF STRING
73 063130 005037 063464          CLR     BOTADR      ;CLEAR BOT ADDRESS
74 063134 112203          11$:     MOVB     (R2)+,R3      ;END OF STRING??
75 063136 001454          BEQ      17$         ;YES!!
76 063140 122703 000015          CMPB    #CR,R3      ;CARRIAGE RETURN??
77 063144 001003          BNE     12$         ;NO!!
78 063146 105037 063467          CLRB     CHRCNT      ;YES-CLEAR CHAR COUNT
79 063152 000770          BR      11$         ;GET NEXT CHARACTER
80 063154 122703 000012          12$:     CMPB    #LF,R3      ;LINE FEED??
81 063160 001765          BEQ      11$         ;YES-GET NEXT CHARACTER
82 063162 122703 000011          CMPB    #HT,R3      ;HORIZONTAL TAB??
83 063166 001007          BNE     14$         ;NO!!
84 063170 105237 063467          13$:     INCB     CHRCNT      ;ADJUST CHARACTER COUNT
85 063174 132737 000007 063467          BITB    #7,CHRCNT
86 063202 001372          BNE     13$
87 063204 000407          BR      15$
88 063206 105237 063467          14$:     INCB     CHRCNT      ;INCREMENT CHARACTER COUNT
89 063212 122703 000040          CMPB    #' ,R3      ;SPACE??
90 063216 001002          BNE     15$         ;NO!!
91 063220 010237 063464          MOV     R2,BOTADR    ;SAVE ADDRESS OF SPACE
92 063224 122737 000100 063467          15$:     CMPB    #64.,CHRCNT ;END OF LINE??
93 063232 103340          BHS     11$         ;NO!!
94 063234 013704 063464          MOV     BOTADR,R4    ;GET ADDRESS OF LAST SPACE
95 063240 001007          BNE     16$         ;BRANCH IF SPACE DETECTED
96 063242 104401 001217          TYPE    ,SCRLF      ;TYPE CRLF
97 063246 105037 063467          CLRB     CHRCNT      ;CLEAR CHARACTER COUNT
98 063252 013702 063272          MOV     18$,R2      ;SET UP R2 FOR TESTING
    
```



```

99 063256 000726          BR      11$
100 063260 105044          CLRB   -(R4)          ;REPLACE SPACE
101 063262 112737 177777 063466  MOVB  #-1,BOTFLG     ;SET BOT FLAG
102 063270 104401          TYPE   ;TYPE ERROR MESSAGE STRING
103 063272 000000          .WORD ;STRING ADDRESS GOES HERE
104 063274 105737 063466  TSTB  BOTFLG         ;WAS STRING TRUNCATED??
105 063300 001707          BEQ   10$            ;NO!!
106 063302 104401 001217  TYPE   ,SCRLF        ;YES-TYPE CRLF
107 063306 105037 063466  CLRB  BOTFLG         ;CLEAR BOT FLAG
108 063312 105037 063467  CLRB  CHRCNT         ;CLEAR CHARACTER COUNT
109 063316 013702 063464  MOV   BOTADR,R2      ;SETUP R2 FOR TESTING
110 063322 010237 063272  MOV   R2,18$        ;SETUP 18$ FOR TYPING
111 063326 112742 000040  MOVB  #'-(R2)        ;RESTORE SPACE
112 063332 105722          TSTB  (R2)+          ;RESTORE R2
113 063334 000677          BR    11$           ;TYPE REST OF STRING
114
115          ;TYPE ERROR HEADER AND ERROR DATA
116 063336 016001 000002 19$:  MOV   2(R0),R1     ;R1 POINTS TO ERROR HEADER TABLE
117 063342 001444          BEQ   27$           ;BRANCH IF NO HEADER
118 063344 104401 001217  TYPE   ,SCRLF        ;(ASSUME NO DATA)
119 063350 016002 000004  MOV   4(R0),R2      ;R2 POINTS TO DATA ADDRESS TABLE
120 063354 016003 000006  MOV   6(R0),R3      ;R3 POINTS TO FORMAT TABLE
121 063360 012137 063370 20$:  MOV   (R1)+,21$    ;PUT HEADER ADDRESS FOR TYPE
122 063364 001433          BEQ   27$           ;BRANCH IF END OF HEADERS
123          ;(ASSUME END OF DATA)
124 063366 104401          TYPE   ;HEADER ADDRESS GOES HERE
125 063370 000000 21$:  .WORD  0
126 063372 104401 001217  TYPE   ,SCRLF
127 063376 005702          TST   R2            ;DATA WITH HEADER??
128 063400 001767          BEQ   20$           ;NO!!
129 063402 012204          MOV   (R2)+,R4      ;R4 POINTS TO DATA ADDRESS
130 063404 012305          MOV   (R3)+,R5      ;R5 POINTS TO FORMAT
131 063406 105725 22$:  TSTB  (R5)+          ;WHAT KIND OF DATA??
132 063410 100407          BMI   24$           ;BINARY
133 063412 001403          BEQ   23$           ;OCTAL
134 063414 013446          MOV   @ (R4)+,-(SP) ;DECIMAL
135 063416 104405          TYPDS
136 063420 000405          BR    25$
137 063422 013446 23$:  MOV   @ (R4)+,-(SP)
138 063424 104402          TYPOC
139 063426 000402          BR    25$
140 063430 013446 24$:  MOV   @ (R4)+,-(SP)
141 063432 104406          TYPBN
142 063434 005714 25$:  TST   (R4)          ;MORE DATA??
143 063436 001403          BEQ   26$           ;NO!!
144 063440 104401 063522  TYPE   ,ERTY04      ;YES-TYPE 2 SPACES
145 063444 000760          BR    22$           ;AND CONTINUE
146 063446 104401 001217 26$:  TYPE   ,SCRLF        ;TYPE ONE BLANK LINE
147 063452 000742          BR    20$           ;BEFORE NEXT HEADER
148 063454 104415 27$:  RESREG
149 063456 000207          RTS   PC
150
151 063460 000000  TSTNMB: .WORD  0    ;TEST NUMBER
152 063462 000000  ERRNMB: .WORD  0    ;ERROR NUMBER
153 063464 000000  BOTADR: .WORD  0    ;BEGINNING OF TEXT ADDRESS
154 063466 000     BOTFLG: .BYTE  0    ;BOT FLAG
155 063467 000     CHRCNT: .BYTE  0    ;CHARACTER COUNT
    
```

156						
157	063470	104	122	126	ERTY00:	.ASCIZ @DRV#a
158	063475	054	040	124	ERTY01:	.ASCIZ @, TEST#a
159	063505	054	040	105	ERTY02:	.ASCIZ @, ERR#a
160	063514	054	040	120	ERTY03:	.ASCIZ @, PC=@
161	063522	040	040	000	ERTY04:	.ASCIZ @ @
162	063525	040	055	040	ERTY05:	.ASCIZ @ - @
163						.EVEN
164	063532	063542	063630	063646	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4 ;WORDS DEFINING TABLES BELOW
165	063542	063546	000000		PFECH1:	+.4,0
166	063546	120	117	127		.ASCIZ ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
167						.EVEN
168	063630	063634	000000		PFECH2:	+.4,0
169	063634	103	120	125		.ASCIZ ?CPUERREG?
170						.EVEN
171	063646	063650			PFECH3:	+.2
172	063650	062640	000000			.WORD CPSAVE,0
173	063654	063656			PFECH4:	+.2
174	063656	000	000			.BYTE 0,0

1

.SBTTL TTY INPUT ROUTINE

063660 000000
 063662 000000
 063664 000000
 063666 063667

```

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

```

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

063670 005037 063660
 063674 012737 063666 063662
 063702 013737 063662 063664
 063710 012737 063740 000060
 063716 012737 000200 000062
 063724 005777 115232
 063730 012777 000100 115222
 063736 000207

```

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

```

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

063740 117746 115216
 063744 042716 177600
 063750 021627 000021
 063754 001002
 063756 005726
 063760 000002
 063762
 063762 021627 000003
 063766 001007
 063770 104401 065066
 063774 004737 063670
 064000 005726
 064002 000137 065130
 064006 021627 000007
 064012 001004
 064014 022737 000176 001154
 064022 001500
 064024
 064024 022737 000001 063660
 064032 001004
 064034 104401 001212

```

$TKSRV: MOV     $TKQVEC,-(SP) ;;PICKUP THE CHARACTER
        BIC     #^C177,(SP)  ;;STRIP THE JUNK
        CMP     (SP),#$XON  ;;IS IT A RANDOM XON?
        BNE     30$         ;;BRANCH IF NO
        TST    (SP)+        ;;CLEAN RANDOM XON OFF STACK
        RTI     30$        ;;RETURN
30$:    CMP     (SP),#3      ;;IS IT A CONTROL C?
        BNE     1$         ;;BRANCH IF NO
        TYPE   ,SCNTLC     ;;TYPE A CONTROL-C (^C)
        JSR    PC,$TKINT   ;;INIT THE KEYBOARD
        TST    (SP)+        ;;CLEAN UP STACK
        JMP    SHUT2       ;;CONTROL C RESTART
1$:    CMP     (SP),#7      ;;IS IT A CONTROL G?
        BNE     2$         ;;BRANCH IF NO
        CMP     #SWREG,SWR  ;;IS SOFT-SWR SELECTED?
        BEQ    6$         ;;GO TO SWR CHANGE
2$:    CMP     #1,$TKCNT    ;;IS THE QUEUE FULL?
        BNE     3$         ;;BRANCH IF NO
        TYPE   ,SBELL      ;;RING THE TTY BELL
  
```

```

064040 005726          TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
064042 000451          BR        5$        ;;EXIT
064044 021627 000023  3$:      CMP      (SP),#23  ;;IS IT A CONTROL-S?
064050 001021          BNE      32$       ;;BRANCH IF NO
064052 005077 115102          CLR      @STKS     ;;DISABLE TTY KEYBOARD INTERRUPTS
064056 005726          TST      (SP)+      ;;CLEAN CHAR OFF STACK
064060 105777 115074  31$:      TSTB    @STKS     ;;WAIT FOR A CHAR
064064 100375          BPL      31$       ;;LOOP UNTIL ITS THERE
064066 117746 115070          MOVB    @STKB,-(SP) ;;GET THE CHARACTER
064072 042716 177600          BIC     #'^C177,(SP) ;;MAKE IT 7-BIT ASCII
064076 022627 000021          CMP     (SP)+,#21  ;;IS IT A CONTROL-Q?
064102 001366          BNE      31$       ;;BRANCH IF NO
064104 012777 000100 115046          MOV     #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
064112 000002          RTI                    ;;RETURN
064114 005237 063660  32$:      INC     $TKCNT     ;;COUNT THIS CHARACTER
064120 021627 000140          CMP     (SP),#140 ;;IS IT UPPER CASE?
064124 002405          BLT     4$         ;;BRANCH IF YES
064126 021627 000175          CMP     (SP),#175 ;;IS IT A SPECIAL CHAR?
064132 003002          BGT     4$         ;;BRANCH IF YES
064134 042716 000040          BIC     #40,(SP)   ;;MAKE IT UPPER CASE
064140 112677 177516  4$:      MOVB    (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
064144 005237 063662          INC     $TKQIN     ;;UPDATE THE POINTER
064150 023727 063662 063667          CMP     $TKQIN,$$TKQEND ;;GO OFF THE END?
064156 001003          BNE      5$        ;;BRANCH IF NO
064160 012737 063666 063662          MOV     $$TKQSRST,$$TKQIN ;;RESET THE POINTER
064166 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.

```

```

064170 022737 000176 001154 $CKSWR: CMP     #SWREG,$SWR  ;;IS THE SOFT-SWR SELECTED
064176 001124          BNE      15$       ;;EXIT IF NOT
064200 105777 114754          TSTB    @STKS     ;;IS A CHAR WAITING?
064204 100121          BPL      15$       ;;IF NOT, EXIT
064206 117746 114750          MOVB    @STKB,-(SP) ;;YES
064212 042716 177600          BIC     #'^C177,(SP) ;;MAKE IT 7-BIT ASCII
064216 021627 000007          CMP     (SP),#7   ;;IS IT A CONTROL-G?
064222 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.

```

```

064224 123727 001150 000001 6$:      CMPB    $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
064232 001674          BEQ     2$         ;;BRANCH IF YES
064234 005726          TST     (SP)+      ;;CLEAR CONTROL-G OFF STACK
064236 004737 063670          JSR     PC,$TKINT  ;;FLUSH THE TTY INPUT QUEUE
064242 005077 114712          CLR     @STKS     ;;DISABLE TTY KEYBOARD INTERRUPTS
064246 112737 000001 001151          MOVB    #1,$INTAG  ;;SET INTERRUPT MODE INDICATOR

064254 104401 065100          TYPE    ,SCNTLG    ;;ECHO THE CONTROL-G (^G)
064260 104401 065105          SGTSWR: TYPE    ,SMSWR    ;;TYPE CURRENT CONTENTS
064264 013746 000176          MOV     SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
064270 104402          TYPOC           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

```


064272	104401	065116		TYPE	.SMNEW	::PROMPT FOR NEW SWR
064276	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
064300	005046			CLR	-(SP)	::THE NEW SWR
064302	105777	114652	7\$:	TSTB	@STKS	::CHAR THERE?
064306	100375			BPL	7\$::IF NOT TRY AGAIN
064310	117746	114646		MOVB	@STKB, -(SP)	::PICK UP CHAR
064314	042716	177600		BIC	#^C177, (SP)	::MAKE IT 7-BIT ASCII
064320	021627	000003		CMP	(SP), #3	::IS IT A CONTROL-C?
064324	001015			BNE	9\$::BRANCH IF NOT
064326	104401	065066		TYPE	.SCNTLC	::YES, ECHO CONTROL-C (^C)
064332	062706	000006		ADD	#6, SP	::CLEAN UP STACK
064336	123727	001151	000001	CMPB	\$INTAG, #1	::REENABLE TTY KEYBOARD INTERRUPTS?
064344	001003			BNE	8\$::BRANCH IF NO
064346	012777	000100	114604	MOV	#100, @STKS	::ALLOW TTY KEYBOARD INTERRUPTS
064354	000137	065130	8\$:	JMP	SHUT2	::CONTROL-C RESTART
064360	021627	000025	9\$:	CMP	(SP), #25	::IS IT A CONTROL-U?
064364	001005			BNE	10\$::BRANCH IF NOT
064366	104401	065073		TYPE	.SCNTLU	::YES, ECHO CONTROL-U (^U)
064372	062706	000006	20\$:	ADD	#6, SP	::IGNORE PREVIOUS INPUT
064376	000737			BR	19\$::LET'S TRY IT AGAIN
064400	021627	000015	10\$:	CMP	(SP), #15	::IS IT A <CR>?
064404	001022			BNE	16\$::BRANCH IF NO
064406	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
064412	001403			BEQ	11\$::BRANCH IF YES
064414	016677	000002	114532	MOV	2(SP), @SWR	::SAVE NEW SWR
064422	062706	000006	11\$:	ADD	#6, SP	::CLEAR UP STACK
064426	104401	001217	14\$:	TYPE	.SCRLF	::ECHO <CR> AND <LF>
064432	123727	001151	000001	CMPB	\$INTAG, #1	::RE-ENABLE TTY KBD INTERRUPTS?
064440	001003			BNE	15\$::BRANCH IF NOT
064442	012777	000100	114510	MOV	#100, @STKS	::RE-ENABLE TTY KBD INTERRUPTS
064450	000002		15\$:	RTI		::RETURN
064452	004737	061346	16\$:	JSR	PC, \$TYPEC	::ECHO CHAR
064456	021627	000060		CMP	(SP), #60	::CHAR < 0?
064462	002420			BLT	18\$::BRANCH IF YES
064464	021627	000067		CMP	(SP), #67	::CHAR > 7?
064470	003015			BGT	18\$::BRANCH IF YES
064472	042726	000060		BIC	#60, (SP)+	::STRIP-OFF ASCII
064476	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
064502	001403			BEQ	17\$::BRANCH IF YES
064504	006316			ASL	(SP)	::NO, SHIFT PRESENT
064506	006316			ASL	(SP)	::CHAR OVER TO MAKE
064510	006316			ASL	(SP)	::ROOM FOR NEW ONE.
064512	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
064516	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
064522	000667			BR	7\$::GET THE NEXT ONE
064524	104401	001216	18\$:	TYPE	.\$QUES	::TYPE ?<CR><LF>
064530	000720			BR	20\$::SIMULATE CONTROL-U
				.DSABL	LSB	

::*****

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *   RDCHR          ::GET A CHARACTER FROM THE QUEUE
: *   RETURN HERE   ::CHARACTER IS ON THE STACK
: *                ::WITH PARITY BIT STRIPPED OFF
:
064532 011646          SRDCHR: MOV    (SP),-(SP)      ::PUSH DOWN THE PC AND
064534 016666 000004 000002  MOV    4(SP),2(SP)    ::THE PS
064542 005066 000004          CLR    4(SP)          ::GET READY FOR A CHARACTER
064546 005046          CLR    -(SP)         ::PUT NEW PS ON STACK
064550 012746 064556          MOV    #64$,-(SP)      ::PUT NEW PC ON STACK
064554 000002          RTI                    ::POP NEW PC AND PS
064556
064556 005737 063660 64$:  TST    $TKCNT          ::WAIT ON A CHARACTER
064562 001775 1$:  BEQ    1$
064564 005337 063660          DEC    $TKCNT          ::DECREMENT THE COUNTER
064570 117766 177070 000004  MOVB  @$TKQOUT,4(SP)  ::GET ONE CHARACTER
064576 005237 063664          INC    $TKQOUT        ::UPDATE THE POINTER
064602 023727 063664 063667  CMP    $TKQOUT,$$TKQEND ::DID IT GO OFF OF THE END?
064610 001003          BNE    2$            ::BRANCH IF NO
064612 012737 063666 063664  MOV    $$TKQRT,$$TKQOUT ::RESET THE POINTER
064620 000002          RTI                    ::RETURN
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *   RDLIN          ::INPUT A STRING FROM THE TTY
: *   RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                ::TERMINATOR WILL BE A BYTE OF ALL 0'S
:
064622 010346          SRDLIN: MOV    R3,-(SP)      ::SAVE R3
064624 005046          CLR    -(SP)         ::CLEAR THE RUBOUT KEY
064626 012703 065056 1$:  MOV    $$TTYIN,R3     ::GET ADDRESS
064632 022703 065066 2$:  CMP    $$TTYIN+8.,R3  ::BUFFER FULL?
064636 101456          BLOS   4$            ::BR IF YES
064640 104411          RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
064642 112613          MOVB  (SP)+,(R3)     ::GET CHARACTER
064644 122713 000177 10$:  CMPB  #177,(R3)     ::IS IT A RUBOUT
064650 001022          BNE    5$            ::BR IF NO
064652 005716          TST    (SP)         ::IS THIS THE FIRST RUBOUT?
064654 001007          BNE    6$            ::BR IF NO
064656 112737 000134 065054  MOVB  #'\",9$        ::TYPE A BACK SLASH
064664 104401 065054          TYPE  ,9$
064670 012716 177777          MOV    #-1,(SP)     ::SET THE RUBOUT KEY
064674 005303 6$:  DEC    R3            ::BACKUP BY ONE
064676 020327 065056          CMP    R3,$$TTYIN  ::STACK EMPTY?
064702 103434          BLO   4$            ::BR IF YES
064704 111337 065054          MOVB  (R3),9$       ::SETUP TO TYPEOUT THE DELETED CHAR.
064710 104401 065054          TYPE  ,9$
064714 000746          BR    2$            ::GO TYPE
064716 005716          TST    (SP)         ::GO READ ANOTHER CHAR.
064720 001406          BEQ    7$            ::RUBOUT KEY SET?
064722 112737 000134 065054  MOVB  #'\",9$        ::TYPE A BACK SLASH
064730 104401 065054          TYPE  ,9$
064734 005016          CLR    (SP)         ::CLEAR THE RUBOUT KEY
064736 122713 000025 7$:  CMPB  #25,(R3)     ::IS CHARACTER A CTRL U?
064742 001003          BNE    8$            ::BR IF NO
  
```


064744	104401	065073			TYPE	SCNTLU	::TYPE A CONTROL 'U'	
064750	000726				BR	1\$::GO START OVER	
064752	122713	000022	8\$:		CMPB	#22,(R3)	::IS CHARACTER A '^R'?	
064756	001011				BNE	3\$::BRANCH IF NO	
064760	105013				CLRB	(R3)	::CLEAR THE CHARACTER	
064762	104401	001217			TYPE	,\$CRLF	::TYPE A 'CR' & 'LF'	
064766	104401	065056			TYPE	,\$TTYIN	::TYPE THE INPUT STRING	
064772	000717				BR	2\$::GO PICKUP ANOTHER CHACTER	
064774	104401	001216	4\$:		TYPE	,\$QUES	::TYPE A '?'	
065000	000712				BR	1\$::CLEAR THE BUFFER AND LOOP	
065002	111337	065054	3\$:		MOVB	(R3),9\$::ECHO THE CHARACTER	
065006	104401	065054			TYPE	,\$9\$		
065012	122723	000015			CMPB	#15,(R3)+	::CHECK FOR RETURN	
065016	001305				BNE	2\$::LOOP IF NOT RETURN	
065020	105063	177777			CLRB	-1(R3)	::CLEAR RETURN (THE 15)	
065024	104401	001220			TYPE	,\$LF	::TYPE A LINE FEED	
065030	005726				TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK	
065032	012603				MOV	(SP)+,R3	::RESTORE R3	
065034	011646				MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE	
065036	016666	000004	000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT	
065044	012766	065056	000004		MOV	,\$TTYIN,4(SP)		
065052	000002				RTI		::RETURN	
065054	000			9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE	
065055	000				.BYTE	0	::TERMINATOR	
065056				,\$TTYIN:	.BLKB	8	::RESERVE 8 BYTES FOR TTY INPUT	
065066	136	103	015	,\$CNTLC:	.ASCIZ	/^C/<15><12>	::CONTROL 'C'	
065073	136	125	015	,\$CNTLU:	.ASCIZ	/^U/<15><12>	::CONTROL 'U'	
065100	136	107	015	,\$CNTLG:	.ASCIZ	/^G/<15><12>	::CONTROL 'G'	
065105	015	012	123	,\$MSWR:	.ASCIZ	<15><12>/SWR = /		
065116	040	040	116	,\$MNEW:	.ASCIZ	/ NEW = /		
					.EVEN			
2								
3	065130	012737	177777	001326	SHUT2:	MOV	#-1,CTLFG	::SET THE CONTROL-C FLAG
4	065136	105737	001116			TSTB	,\$STNM	::DOING ANY TESTS ?
5	065142	001002				BNE	1\$::BR IF YES
6	065144	000137	033776			JMP	SHUT	::NO--RESPOND TO ^C
7	065150	000002			1\$:	RTI		::EXIT FROM INTERRUPT

1

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

::*****
 ::*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 ::*CHANGE IT TO BINARY.

::*CALL:

::* RDOCT
 ::* RETURN HERE
 ::*

:::READ AN OCTAL NUMBER
 :::LOW ORDER BITS ARE ON TOP OF THE STACK
 :::HIGH ORDER BITS ARE IN \$HIOCT

065152	011646		\$RDOCT:	MOV	(SP),-(SP)	:::PROVIDE SPACE FOR THE
065154	016666	000004 000002		MOV	4(SP),2(SP)	:::INPUT NUMBER
065162	010046			MOV	R0,-(SP)	:::PUSH R0 ON STACK
065164	010146			MOV	R1,-(SP)	:::PUSH R1 ON STACK
065166	010246			MOV	R2,-(SP)	:::PUSH R2 ON STACK
065170	104412		1\$:	RDLIN		:::READ AN ASCII LINE
065172	012600			MOV	(SP)+,R0	:::GET ADDRESS OF 1ST CHARACTER
065174	005001			CLR	R1	:::CLEAR DATA WORD
065176	005002			CLR	R2	
065200	112046		2\$:	MOVB	(R0)+,-(SP)	:::PICKUP THIS CHARACTER
065202	001412			BEQ	3\$:::IF ZERO GET OUT
065204	006301			ASL	R1	:::*2
065206	006102			ROL	R2	
065210	006301			ASL	R1	:::*4
065212	006102			ROL	R2	
065214	006301			ASL	R1	:::*8
065216	006102			ROL	R2	
065220	042716	177770		BIC	#^C7,(SP)	:::STRIP THE ASCII JUNK
065224	062601			ADD	(SP)+,R1	:::ADD IN THIS DIGIT
065226	000764			BR	2\$:::LOOP
065230	005726		3\$:	TST	(SP)+	:::CLEAN TERMINATOR FROM STACK
065232	010166	000012		MOV	R1,12(SP)	:::SAVE THE RESULT
065236	010237	065252		MOV	R2,\$HIOCT	
065242	012602			MOV	(SP)+,R2	:::POP STACK INTO R2
065244	012601			MOV	(SP)+,R1	:::POP STACK INTO R1
065246	012600			MOV	(SP)+,R0	:::POP STACK INTO R0
065250	000002			RTI		:::RETURN
065252	000000		\$HIOCT:	.WORD	0	:::HIGH ORDER BITS GO HERE

.SBTTL TRAP DECODER

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

065254	016646	000002	\$TRAP:	MOV	2(SP),-(SP)	::ASSUME THE STATUS OF
065260	042716	000020		BIC	#20,(SP)	:: THE CALLER--DO NOT ALLOW
065264	012746	065272		MOV	#1\$,-(SP)	:: T-BIT TRAPS
065270	000002			RTI		::SET THE NEW STATUS
065272	010046		1\$:	MOV	R0,-(SP)	::SAVE R0
065274	016600	000002		MOV	2(SP),R0	::GET TRAP ADDRESS
065300	005740			TST	-(R0)	::BACKUP BY 2
065302	111000			MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
065304	006300			ASL	R0	::POSITION FOR INDEXING
065306	016000	065326		MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
065312	000200			RTS	R0	::GO TO ROUTINE

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

065314	011646		\$TRAP2:	MOV	(SP),-(SP)	::MOVE THE PC DOWN
065316	016666	000004		MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
065324	000002	000002		RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

			:	ROUTINE		
			:	-----		
065326	065314		\$TRPAD:	.WORD	\$TRAP2	
065330	061134			\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
065332	060732			\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
065334	060706			\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
065336	060746			\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
065340	060462			\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
065342	060406			\$TYPBN	::CALL=TYPBN	TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
065344	064260			\$GTSWR	::CALL=GTSWR	TRAP+7(104407) GET SOFT-SWR SETTING
065346	064170			\$CKSWR	::CALL=CKSWR	TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
065350	064532			\$RDCHR	::CALL=RDCHR	TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
065352	064622			\$RDLIN	::CALL=RDLIN	TRAP+12(104412) TTY TYPEIN STRING ROUTINE
065354	065152			\$RDOCT	::CALL=RDOCT	TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
065356	060312			\$SAVREG	::CALL=SAVREG	TRAP+14(104414) SAVE R0-R5 ROUTINE
065360	060350			\$RESREG	::CALL=RESREG	TRAP+15(104415) RESTORE R0-R5 ROUTINE

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
065540 112737 000001 066004 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
065546 112737 000001 066002 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
065554 000403
065556 112737 000001 066004 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
065564
065564 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
065566 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
065570 105737 066002 TSTB SMFLG ::SHOULD TYPE A MESSAGE?
065574 001450 BEQ 5$ ::IF NOT: BR
065576 122737 000001 001242 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
065604 001031 BNE 3$ ::IF NOT: BR
065606 132737 000100 001243 BITB #APTSPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
065614 001425 BEQ 3$ ::IF NOT: BR
065616 017600 000004 MOV @4(SP),R0 ::GET MESSAGE ADDR.
065622 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
065630 005737 001222 1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
065634 001375 BNE 1$ ::IF NOT: WAIT
065636 010037 001236 MOV R0,$MSGAD ::PUT ADDR IN MAILBOX
065642 105720 2$: TSTB (R0)+ ::FIND END OF MESSAGE
065644 001376 BNE 2$
065646 163700 001236 SUB $MSGAD,R0 ::SUB START OF MESSAGE
065652 006200 ASR R0 ::GET MESSAGE LNTH IN WORDS
065654 010037 001240 MOV R0,$MSGLGT ::PUT LENGTH IN MAILBOX
065660 012737 000004 001222 MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
065666 000413 BR 5$
065670 017637 000004 065714 3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
065676 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
065704 013746 177776 MOV 177776,-(SP) ::PUSH 177776 ON STACK
065710 004737 061134 JSR PC,$TYPE ::CALL TYPE MACRO
065714 000000 4$: .WORD 0
065716 5$:
065716 105737 066004 10$: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
065722 001416 BEQ 12$ ::IF NOT: BR
065724 005737 001242 TST $ENV ::RUNNING UNDER APT?
065730 001413 BEQ 12$ ::IF NOT: BR
065732 005737 001222 11$: TST $MSGTYPE ::FINISHED LAST MESSAGE?
065736 001375 BNE 11$ ::IF NOT: WAIT
065740 017637 000004 001224 MOV @4(SP),$FATAL ::GET ERROR #
065746 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
065754 005237 001222 INC $MSGTYPE ::TELL APT TO TAKE ERROR
065760 105037 066004 12$: CLRB $FFLG ::CLEAR FATAL FLAG
065764 105037 066003 CLRB $LFLG ::CLEAR LOG FLAG
065770 105037 066002 CLRB $MFLG ::CLEAR MESSAGE FLAG
065774 012601 MOV (SP)+,R1 ::POP STACK INTO R1
065776 012600 MOV (SP)+,R0 ::POP STACK INTO R0
066000 000207 RTS PC ::RETURN
066002 000 $MFLG: .BYTE 0 ::MESSG. FLAG
066003 000 $LFLG: .BYTE 0 ::LOG FLAG
066004 000 $FFLG: .BYTE 0 ::FATAL FLAG
        .EVEN
000200 APTSIZE = 200
000001 APTENV = 001
000100 APTSPOOL = 100
000040 APTCSUP = 040
  
```

.SBTTL CONSOLE MESSAGES

2					
3	066006	200	103	101	SCTMSG: .ASCIZ <CRLF>@CANNOT RECOVER THE BAD SECTOR FILES ON THIS DEVICE@
4	066072	075	000		EQUALS: .ASCIZ @=@
5	066074	101	114	114	ALL: .ASCIZ @ALL@<CRLF>
6	066101	040	077	040	QUES: .ASCIZ @ ? @
7	066105	054	040	000	COMMA: .ASCIZ @, @
8	066110	200	124	117	MSHELP: .ASCII <CRLF>@TO ENSURE THAT NO BAD HEADERS ARE LEFT ON THE DISK@
9	066176	200	104	125	.ASCII <CRLF>@DURING ERROR FREE OPERATION, THIS PROGRAM SHOULD BE@
10	066264	200	102	105	.ASCII <CRLF>@BE HALTED BY TYPING A (^C) CONTROL C. AS A RESULT,@
11	066352	200	124	110	.ASCII <CRLF>@THE PROGRAM WILL BE HALTED WHEN THE DRIVE UNDER TEST@
12	066440	200	110	101	.ASCII <CRLF>@HAS COMPLETED TESTING.@
13	066467	200			.ASCII <CRLF>
14	066470	200	124	131	.ASCIZ <CRLF>@TYPE HELP TEXT (L) N ? @
15	066521	200	122	115	CNSLO1: .ASCIZ <CRLF>@RMCS1=@
16	066531	040	114	111	CNSLO2: .ASCIZ @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
17	066573	122	115	126	CNSLO3: .ASCIZ @RMVEC=@
18	066602	040	114	111	CNSLO4: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
19	066636	200	124	131	CNSLO7: .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
20	066723	200	101	116	.ASCIZ <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
21	067000	200			CNSLO8: .ASCII <CRLF>
22	067001	040	077	111	CNSLO9: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
23	067022	200	104	122	MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
24	067036	104	122	111	MSGDRV: .ASCIZ /DRIVE/
25	067044	200	125	116	SYSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
26	067062	122	115	070	SRM80: .ASCIZ /RM80/
27	067067	040	116	117	NOTRM: .ASCIZ / NOT AN RM80/
28	067104	040	114	117	LODEV: .ASCIZ / LOAD DEVICE/
29	067121	040	116	117	NOTPRS: .ASCIZ / NOT PRESENT/
30	067136	040	116	117	NOTAVL: .ASCIZ / NOT AVAILABLE/
31	067155	040	117	106	UNTOFF: .ASCIZ / OFFLINE/
32	067166	040	117	116	UNTON: .ASCIZ / ONLINE/
33	067176	200	104	122	DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED/
34	067225	116	117	116	NONE: .ASCIZ /NONE/
35	067232	116	000		N: .ASCIZ /N/
36	067234	131	000		Y: .ASCIZ /Y/
37	067236	040			BLNKS4: .ASCII / /
38	067237	040			BLNKS3: .ASCII / /
39	067240	040			BLNKS2: .ASCII / /
40	067241	040	000		BLNKS1: .ASCIZ / /
41					.EVEN

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

.SBTTL FUNCTION CODE TABLE

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

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

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

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

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

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

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

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

: BIT 08 IS NOT USED.

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

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

: BIT 05 IS NOT USED.

: BIT 04 IS NOT USED.

: BIT 03 IS NOT USED.

: BIT 02 IS NOT USED.

: BIT 01 IS NOT USED.

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

067244

067244 020000

FNCDTB:

.WORD OPI

:FUNCTION CODE TABLE

:NOP

58	067246	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	067250	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	067252	130000	.WORD	ATA!OPI!IVC	:RECALIBRATE
61	067254	020000	.WORD	OPI	:DRIVE CLEAR
62	067256	030000	.WORD	OPI!IVC	:RELEASE
63	067260	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	067262	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	067264	020000	.WORD	OPI	:READ IN PRESET
66	067266	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	067270	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	067272	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	067274	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	067276	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	067300	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	067302	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	067304	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	067306	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	067310	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	067312	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	067314	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	067316	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	067320	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	067322	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	067324	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	067326	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	067330	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	067332	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	067334	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	067336	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	067340	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	067342	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)

1
2
3 067344 001
4 067345 002
5 067346 004
6 067347 010
7 067350 020
8 067351 040
9 067352 100
10 067353 200

.SBTTL ATTENTION (ATA) TABLE

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

Line	Address	Value	Category	Count
1				
2				
3	067354		RGDTPT: MIXED:	0.
4	067354			1.
5	067354	000000		3.
6	067356	000001		7.
7	067360	000003		15.
8	067362	000007		31.
9	067364	000017		63.
10	067366	000037		127.
11	067370	000077		255.
12	067372	000177		511.
13	067374	000377		1023.
14	067376	000777		2047.
15	067400	001777		4095.
16	067402	003777		8191.
17	067404	007777		16383.
18	067406	017777		32767.
19	067410	037777	ONES:	65535.
20	067412	077777		65535.
21	067414	177777		32767.
22	067416	177777		16383.
23	067420	077777		8191.
24	067422	037777		4095.
25	067424	017777		2047.
26	067426	007777		1023.
27	067430	003777		511.
28	067432	001777		255.
29	067434	000777		127.
30	067436	000377		63.
31	067440	000177		31.
32	067442	000077		15.
33	067444	000037		7.
34	067446	000017		3.
35	067450	000007		1.
36	067452	000003		0.
37	067454	000001	ZEROS:	0.
38	067456	000000		1.
39	067460	000000		2.
40	067462	000001		4.
41	067464	000002		8.
42	067466	000004		16.
43	067470	000010		32.
44	067472	000020		64.
45	067474	000040		128.
46	067476	000100		256.
47	067500	000200		512.
48	067502	000400		1024.
49	067504	001000		2048.
50	067506	002000		4096.
51	067510	004000		8192.
52	067512	010000		16384.
53	067514	020000		32768.
54	067516	040000		32768.
55	067520	100000		16384.
56	067522	100000		
57	067524	040000		

58	067526	020000	.WORD	8192.
59	067530	010000	.WORD	4096.
60	067532	004000	.WORD	2048.
61	067534	002000	.WORD	1024.
62	067536	001000	.WORD	512.
63	067540	000400	.WORD	256.
64	067542	000200	.WORD	128.
65	067544	000100	.WORD	64.
66	067546	000040	.WORD	32.
67	067550	000020	.WORD	16.
68	067552	000010	.WORD	8.
69	067554	000004	.WORD	4.
70	067556	000002	.WORD	2.
71	067560	000001	.WORD	1.
72	067562	000000	.WORD	0.
73	067564	177777	.WORD	65535.
74	067566	177776	.WORD	65534.
75	067570	177774	.WORD	65532.
76	067572	177770	.WORD	65528.
77	067574	177760	.WORD	65520.
78	067576	177740	.WORD	65504.
79	067600	177700	.WORD	65472.
80	067602	177600	.WORD	65408.
81	067604	177400	.WORD	65280.
82	067606	177000	.WORD	65024.
83	067610	176000	.WORD	64512.
84	067612	174000	.WORD	63488.
85	067614	170000	.WORD	61440.
86	067616	160000	.WORD	57344.
87	067620	140000	.WORD	49152.
88	067622	100000	.WORD	32768.
89	067624	000000	.WORD	0.
90	067626	000000	.WORD	0.
91	067630	100000	.WORD	32768.
92	067632	140000	.WORD	49152.
93	067634	160000	.WORD	57344.
94	067636	170000	.WORD	61440.
95	067640	174000	.WORD	63488.
96	067642	176000	.WORD	64512.
97	067644	177000	.WORD	65024.
98	067646	177400	.WORD	65280.
99	067650	177600	.WORD	65408.
100	067652	177700	.WORD	65472.
101	067654	177740	.WORD	65504.
102	067656	177760	.WORD	65520.
103	067660	177770	.WORD	65528.
104	067662	177774	.WORD	65532.
105	067664	177776	.WORD	65534.
106	067666	177777	.WORD	65535.
107	067670	125252	.WORD	43690.
108	067672	152525	.WORD	43690./2
109	067674	125252	.WORD	43690.
110	067676	177777	.WORD	65535.
111	067700	177776	.WORD	65534.
112	067702	177775	.WORD	65533.
113	067704	177773	.WORD	65531.
114	067706	177767	.WORD	65527.

EARLY:

115	067710	177757	.WORD	65519.
116	067712	177737	.WORD	65503.
117	067714	177677	.WORD	65471.
118	067716	177577	.WORD	65407.
119	067720	177377	.WORD	65279.
120	067722	176777	.WORD	65023.
121	067724	175777	.WORD	64511.
122	067726	173777	.WORD	63487.
123	067730	167777	.WORD	61439.
124	067732	157777	.WORD	57343.
125	067734	137777	.WORD	49151.
126	067736	077777	.WORD	32767.
127	067740	077777	.WORD	32767.
128	067742	137777	.WORD	49151.
129	067744	157777	.WORD	57343.
130	067746	167777	.WORD	61439.
131	067750	173777	.WORD	63487.
132	067752	175777	.WORD	64511.
133	067754	176777	.WORD	65023.
134	067756	177377	.WORD	65279.
135	067760	177577	.WORD	65407.
136	067762	177677	.WORD	65471.
137	067764	177737	.WORD	65503.
138	067766	177757	.WORD	65519.
139	067770	177767	.WORD	65527.
140	067772	177773	.WORD	65531.
141	067774	177775	.WORD	65533.
142	067776	177776	.WORD	65534.
143	070000	177777	.WORD	65535.
144	070002			

ENRGDT:

				.SBTTL ERROR MESSAGE TABLE			
1							
2							
3	070002	074442	000000	EMT1:	.WORD	EMS1,0	
4	070006	074511	074526	000000	EMT2:	.WORD	EMS2,EMS3,0
5	070014	074511	074571	000000	EMT3:	.WORD	EMS2,EMS4,0
6	070022	074634	074664	000000	EMT4:	.WORD	EMS5,EMS6,0
7	070030	074634	074776	000000	EMT5:	.WORD	EMS5,EMS10,0
8	070036	101427	076657	000000	EMT6:	.WORD	EMS167,EMS64,0
9	070044	077425	101454	000000	EMT7:	.WORD	EMS110,EMS170,0
10	070052	074731	000000	EMT10:	.WORD	EMS7,0	
11	070056	074776	000000	EMT11:	.WORD	EMS10,0	
12	070062	075040	075051	000000	EMT12:	.WORD	EMS11,EMS12,0
13	070070	075112	075123	075134	EMT13:	.WORD	EMS13,EMS14,EMS15,EMS16,0
14	070102	075206	076657	000000	EMT14:	.WORD	EMS17,EMS64,0
15	070110	075040	075271	000000	EMT15:	.WORD	EMS11,EMS21,0
16	070116	075040	075314	075443	EMT16:	.WORD	EMS11,EMS22,EMS27,0
17	070126	075040	075330	075454	EMT17:	.WORD	EMS11,EMS23,EMS30,0
18	070136	075040	075356	075454	EMT20:	.WORD	EMS11,EMS24,EMS30,0
19	070146	075040	075405	075443	EMT21:	.WORD	EMS11,EMS25,EMS27,0
20	070156	075040	075422	075443	EMT22:	.WORD	EMS11,EMS26,EMS27,0
21	070166	075040	075464	075454	EMT23:	.WORD	EMS11,EMS31,EMS30,0
22	070176	075040	075513	075454	EMT24:	.WORD	EMS11,EMS32,EMS30,0
23	070206	075040	075542	075454	EMT25:	.WORD	EMS11,EMS33,EMS30,0
24	070216	075040	075570	075454	EMT26:	.WORD	EMS11,EMS34,EMS30,0
25	070226	075040	075641	075454	EMT27:	.WORD	EMS11,EMS35,EMS30,0
26	070236	075040	075670	075454	EMT30:	.WORD	EMS11,EMS36,EMS30,0
27	070246	075040	075717	075454	EMT31:	.WORD	EMS11,EMS37,EMS30,0
28	070256	075040	075745	075454	EMT32:	.WORD	EMS11,EMS40,EMS30,0
29	070266	075040	075774	075454	EMT33:	.WORD	EMS11,EMS41,EMS30,0
30	070276	075040	076022	075454	EMT34:	.WORD	EMS11,EMS42,EMS30,0
31	070306	075040	076051	075454	EMT35:	.WORD	EMS11,EMS43,EMS30,0
32	070316	075040	076100	075454	EMT36:	.WORD	EMS11,EMS44,EMS30,0
33	070326	075040	076153	075454	EMT37:	.WORD	EMS11,EMS45,EMS30,0
34	070336	076743	075246	000000	EMT40:	.WORD	EMS66,EMS20,0
35	070344	077131	100631	077137	EMT41:	.WORD	EMS75,EMS141,EMS76,0
36	070354	100722	100732	077065	EMT42:	.WORD	EMS144,EMS145,EMS72,EMS76,0
37	070366	076245	076363	077137	EMT43:	.WORD	EMS47,EMS53,EMS76,0
38	070376	077170	076363	077137	EMT44:	.WORD	EMS77,EMS53,EMS76,0
39	070406	077216	076363	077137	EMT45:	.WORD	EMS100,EMS53,EMS76,0
40	070416	077244	076363	077137	EMT46:	.WORD	EMS101,EMS53,EMS76,0
41	070426	076675	076657	000000	EMT47:	.WORD	EMS65,EMS64,0
42	070434	075112	075134	076631	EMT50:	.WORD	EMS13,EMS15,EMS63,0
43	070444	075040	076216	075454	EMT51:	.WORD	EMS11,EMS46,EMS30,EMS67,0
44	070456	076245	076363	077013	EMT52:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
45	070474	076245	076363	077013	EMT53:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
46	070512	076274	076363	077013	EMT54:	.WORD	EMS50,EMS53,EMS67,0
47	070522	100657	100674	076363	EMT55:	.WORD	EMS142,EMS143,EMS53,EMS67,0
48	070534	076323	077065	077013	EMT56:	.WORD	EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
49	070552	101427	077075	077013	EMT57:	.WORD	EMS167,EMS73,EMS67,0
50	070562	076446	077013	077625	EMT60:	.WORD	EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
51	070600	077052	076446	077013	EMT61:	.WORD	EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
52	070620	100610	077013	077625	EMT62:	.WORD	EMS140,EMS67,EMS115,EMS47,EMS70,0
53	070634	102627	102652	075246	EMT63:	.WORD	EMS225,EMS226,EMS20,0
54	070644	101015	101057	077040	EMT64:	.WORD	EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
55	070664	076351	102054	102235	EMT65:	.WORD	EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
56	070704	101366	077321	077065	EMT66:	.WORD	EMS165,EMS103,EMS72,EMS124,0
57	070716	101366	077321	077065	EMT67:	.WORD	EMS165,EMS103,EMS72,EMS171,0

58	070730	076216	075246	101261	EMT70:	.WORD	EMS46,EMS20,EMS163,0
59	070740	077052	077244	101261	EMT71:	.WORD	EMS71,EMS101,EMS163,0
60	070750	076245	077065	101261	EMT72:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS140,EMS141,0
61	070766	076245	077065	101261	EMT73:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS141,EMS72,0
62	071004	076446	076363	101261	EMT74:	.WORD	EMS56,EMS53,EMS163,0
63	071014	077052	076446	101261	EMT75:	.WORD	EMS71,EMS56,EMS163,EMS115,EMS150,EMS152,EMS72,0
64	071034	076274	076363	101261	EMT76:	.WORD	EMS50,EMS53,EMS163,0
65	071044	100657	100674	076363	EMT77:	.WORD	EMS142,EMS143,EMS53,EMS163,0
66	071056	077131	100631	101261	EMT100:	.WORD	EMS75,EMS141,EMS163,EMS115,EMS47,EMS70,0
67	071074	077131	101015	101261	EMT101:	.WORD	EMS75,EMS150,EMS163,EMS115,EMS56,EMS73,0
68	071112	101427	077075	101261	EMT102:	.WORD	EMS167,EMS73,EMS163,0
69	071122	101000	101033	077112	EMT103:	.WORD	EMS147,EMS151,EMS74,EMS163,0
70	071134	076323	077112	101261	EMT104:	.WORD	EMS51,EMS74,EMS163,EMS115,EMS50,EMS70,0
71	071152	101366	077216	101261	EMT105:	.WORD	EMS165,EMS100,EMS163,0
72	071162	101366	077170	101261	EMT106:	.WORD	EMS165,EMS77,EMS163,0
73	071172	100722	100732	076363	EMT107:	.WORD	EMS144,EMS145,EMS53,EMS163,EMS115,EMS143,EMS70,0
74	071212	077425	077465	077630	EMT110:	.WORD	EMS110,EMS112,EMS116,EMS111,0
75	071224	077551	074571	000000	EMT111:	.WORD	EMS113,EMS4,0
76	071232	077551	074526	000000	EMT112:	.WORD	EMS113,EMS3,0
77	071240	077425	077625	077630	EMT113:	.WORD	EMS110,EMS115,EMS116,EMS117,EMS114,0
78	071254	000000			EMT114:	.WORD	
79	071256	077702	100342	100366	EMT115:	.WORD	EMS121,EMS132,EMS133,0
80	071266	077745	100342	100366	EMT116:	.WORD	EMS122,EMS132,EMS133,0
81	071276	100002	100342	100366	EMT117:	.WORD	EMS123,EMS132,EMS133,0
82	071306	100045	100342	100366	EMT120:	.WORD	EMS124,EMS132,EMS133,0
83	071316	100077	100342	100366	EMT121:	.WORD	EMS125,EMS132,EMS133,0
84	071326	100142	100342	100366	EMT122:	.WORD	EMS126,EMS132,EMS133,0
85	071336	100553	100342	100366	EMT123:	.WORD	EMS137,EMS132,EMS133,0
86	071346	100244	100342	100366	EMT124:	.WORD	EMS130,EMS132,EMS133,0
87	071356	100302	100342	100366	EMT125:	.WORD	EMS131,EMS132,EMS133,0
88	071366	077702	100342	100411	EMT126:	.WORD	EMS121,EMS132,EMS134,EMS123,0
89	071400	077745	100342	100411	EMT127:	.WORD	EMS122,EMS132,EMS134,EMS123,0
90	071412	100002	100342	100411	EMT130:	.WORD	EMS123,EMS132,EMS134,EMS123,0
91	071424	100045	100342	100411	EMT131:	.WORD	EMS124,EMS132,EMS134,EMS123,0
92	071436	100077	100342	100411	EMT132:	.WORD	EMS125,EMS132,EMS134,EMS123,0
93	071450	100142	100342	100411	EMT133:	.WORD	EMS126,EMS132,EMS134,EMS123,0
94	071462	100553	100342	100411	EMT134:	.WORD	EMS137,EMS132,EMS134,EMS123,0
95	071474	100244	100342	100411	EMT135:	.WORD	EMS130,EMS132,EMS134,EMS123,0
96	071506	100302	100342	100411	EMT136:	.WORD	EMS131,EMS132,EMS134,EMS123,0
97	071520	077702	100342	100453	EMT137:	.WORD	EMS121,EMS132,EMS135,0
98	071530	100002	100342	100453	EMT140:	.WORD	EMS123,EMS132,EMS135,0
99	071540	077702	100342	100526	EMT141:	.WORD	EMS121,EMS132,EMS136,0
100	071550	100553	100342	100526	EMT142:	.WORD	EMS137,EMS132,EMS136,0
101	071560	100045	100342	100526	EMT143:	.WORD	EMS124,EMS132,EMS136,0
102	071570	100077	100342	100526	EMT144:	.WORD	EMS125,EMS132,EMS136,0
103	071600	100142	100342	100526	EMT145:	.WORD	EMS126,EMS132,EMS136,0
104	071610	100302	100342	100526	EMT146:	.WORD	EMS131,EMS132,EMS136,0
105	071620	101503	100342	100526	EMT147:	.WORD	EMS171,EMS132,EMS136,0
106	071630	100244	100342	100526	EMT150:	.WORD	EMS130,EMS132,EMS136,0
107	071640	100610	077625	100631	EMT151:	.WORD	EMS140,EMS115,EMS141,EMS70,0
108	071652	100657	077625	100674	EMT152:	.WORD	EMS142,EMS115,EMS143,EMS72,0
109	071664	100722	100732	100761	EMT153:	.WORD	EMS144,EMS145,EMS146,EMS115,EMS143,EMS72,0
110	071702	100722	100732	075246	EMT154:	.WORD	EMS144,EMS145,EMS20,EMS115,EMS143,EMS70,0
111	071720	101015	101057	101125	EMT155:	.WORD	EMS150,EMS152,EMS154,EMS153,0
112	071732	101000	101033	101125	EMT156:	.WORD	EMS147,EMS151,EMS154,EMS155,0
113	071744	101000	101033	101161	EMT157:	.WORD	EMS147,EMS151,EMS156,EMS157,0
114	071756	101231	101161	101214	EMT160:	.WORD	EMS161,EMS156,EMS160,0

ERROR MESSAGE TABLE

115	071766	075405	075443	101161	EMT161:	.WORD	EMS25,EMS27,EMS156,EMS160,0
116	072000	075422	075443	101161	EMT162:	.WORD	EMS26,EMS27,EMS156,EMS160,0
117	072012	076245	101261	100610	EMT163:	.WORD	EMS47,EMS163,EMS140,0
118	072022	076245	075246	000000	EMT164:	.WORD	EMS47,EMS20,0
119	072030	076446	075246	000000	EMT165:	.WORD	EMS56,EMS20,0
120	072036	076216	075246	000000	EMT166:	.WORD	EMS46,EMS20,0
121	072044	102555	075246	000000	EMT167:	.WORD	EMS224,EMS20,0
122	072052	102013	100631	075454	EMT170:	.WORD	EMS203,EMS141,EMS30,EMS202,0
123	072064	101427	101125	101177	EMT171:	.WORD	EMS167,EMS154,EMS157,0
124	072074	101427	101125	101141	EMT172:	.WORD	EMS167,EMS154,EMS155,0
125	072104	101503	100342	100366	EMT173:	.WORD	EMS171,EMS132,EMS133,0
126	072114	101503	100342	100411	EMT174:	.WORD	EMS171,EMS132,EMS134,EMS123,0
127	072126	077551	101656	000000	EMT175:	.WORD	EMS113,EMS177,0
128	072134	101700	101715	000000	EMT176:	.WORD	EMS200,EMS201,0
129	072142	000000			EMT177:	.WORD	
130	072144	102013	076323	075454	EMT200:	.WORD	EMS203,EMS51,EMS30,EMS202,0
131	072156	102013	102026	075454	EMT201:	.WORD	EMS203,EMS204,EMS30,EMS202,0
132	072170	102013	076274	075454	EMT202:	.WORD	EMS203,EMS50,EMS30,EMS202,0
133	072202	102013	100674	075454	EMT203:	.WORD	EMS203,EMS143,EMS30,EMS202,0
134	072214	101015	077112	101763	EMT204:	.WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
135	072232	076274	077321	077065	EMT205:	.WORD	EMS50,EMS103,EMS72,0
136	072242	077330	077075	101763	EMT206:	.WORD	EMS104,EMS73,EMS202,0
137	072252	077131	100631	077625	EMT207:	.WORD	EMS75,EMS141,EMS115,EMS140,0
138	072264	077131	101015	000000	EMT210:	.WORD	EMS75,EMS150,0
139	072272	076323	077065	077625	EMT211:	.WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
140	072306	100657	076363	077625	EMT212:	.WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
141	072322	076274	077321	076363	EMT213:	.WORD	EMS50,EMS103,EMS53,0
142	072332	076351	102054	101402	EMT214:	.WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
143	072352	076351	077660	076446	EMT215:	.WORD	EMS52,EMS117,EMS56,EMS163,0
144	072364	076446	075454	076657	EMT216:	.WORD	EMS56,EMS30,EMS64,0
145	072374	076216	075454	076657	EMT217:	.WORD	EMS46,EMS30,EMS64,0
146	072404	075464	075454	076657	EMT220:	.WORD	EMS31,EMS30,EMS64,0
147	072414	076245	075454	076657	EMT221:	.WORD	EMS47,EMS30,EMS64,0
148	072424	076351	077660	077170	EMT222:	.WORD	EMS52,EMS117,EMS77,0
149	072434	076351	077660	076631	EMT223:	.WORD	EMS52,EMS117,EMS63,0
150	072444	076351	077660	102627	EMT224:	.WORD	EMS52,EMS117,EMS225,EMS115,EMS226,0
151	072460	102627	102701	102712	EMT225:	.WORD	EMS225,EMS227,EMS230,EMS115,EMS156,EMS231,EMS57,0
152	072500	075040	102712	075454	EMT226:	.WORD	EMS11,EMS230,EMS30,0
153	072510	102627	102652	077625	EMT227:	.WORD	EMS225,EMS226,EMS115,EMS230,EMS72,0
154	072524	000000			EMT230:	.WORD	
155	072526	000000			EMT231:	.WORD	
156	072530	000000			EMT232:	.WORD	
157	072532	000000			EMT233:	.WORD	
158	072534	000000			EMT234:	.WORD	
159	072536	000000			EMT235:	.WORD	
160	072540	000000			EMT236:	.WORD	
161	072542	000000			EMT237:	.WORD	
162	072544	000000			EMT240:	.WORD	
163	072546	000000			EMT241:	.WORD	
164	072550	000000			EMT242:	.WORD	
165	072552	000000			EMT243:	.WORD	
166	072554	000000			EMT244:	.WORD	
167	072556	000000			EMT245:	.WORD	
168	072560	101427	100342	102113	EMT246:	.WORD	EMS167,EMS132,EMS207,0
169	072570	101427	100342	102140	EMT247:	.WORD	EMS167,EMS132,EMS210,EMS125,0
170	072602	101427	102151	102140	EMT250:	.WORD	EMS167,EMS211,EMS210,EMS207,EMS206,0
171	072616	102167	102212	000000	EMT251:	.WORD	EMS212,EMS213,0

172	072624	101366	102167	000000	EMT252: .WORD	EMS165,EMS212,0
173	072632	101000	101033	101161	EMT253: .WORD	EMS147,EMS151,EMS156,EMS210,EMS26,EMS27,0
174	072650	102013	077244	075454	EMT254: .WORD	EMS203,EMS101,EMS30,0
175	072660	102013	101427	075454	EMT255: .WORD	EMS203,EMS167,EMS30,0
176	072670	102013	077216	075454	EMT256: .WORD	EMS203,EMS100,EMS30,0
177	072700	075040	076216	075454	EMT257: .WORD	EMS11,EMS46,EMS30,EMS102,0
178	072712	076351	077660	076446	EMT260: .WORD	EMS52,EMS117,EMS56,EMS102,0
179	072724	076351	102054	102357	EMT261: .WORD	EMS52,EMS205,EMS220,EMS206,EMS115,EMS51,EMS72,0
180	072744	077330	077075	101763	EMT262: .WORD	EMS104,EMS73,EMS202,0
181	072754	076274	076363	077272	EMT263: .WORD	EMS50,EMS53,EMS102,0
182	072764	076446	076363	077272	EMT264: .WORD	EMS56,EMS53,EMS102,EMS115,EMS150,EMS152,EMS70,0
183	073004	077052	076446	077272	EMT265: .WORD	EMS71,EMS56,EMS102,EMS115,EMS150,EMS152,EMS72,0
184	073024	100657	100674	076363	EMT266: .WORD	EMS142,EMS143,EMS53,EMS102,0
185	073036	076274	100761	077625	EMT267: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,0
186	073054	076245	076363	077272	EMT270: .WORD	EMS47,EMS53,EMS102,EMS115,EMS140,0
187	073070	076245	076363	077272	EMT271: .WORD	EMS47,EMS53,EMS102,EMS115,EMS141,EMS72,0
188	073106	077131	100631	077272	EMT272: .WORD	EMS75,EMS141,EMS102,EMS115,EMS47,EMS73,0
189	073124	076323	077112	077272	EMT273: .WORD	EMS51,EMS74,EMS102,EMS115,EMS50,EMS70,0
190	073142	076631	076363	076523	EMT274: .WORD	EMS63,EMS53,EMS57,EMS115,EMS41,EMS146,0
191	073160	077630	076363	075774	EMT275: .WORD	EMS116,EMS53,EMS41,EMS57,0
192	073172	076245	076363	076523	EMT276: .WORD	EMS47,EMS53,EMS57,EMS115,EMS140,0
193	073206	076245	076363	076523	EMT277: .WORD	EMS47,EMS53,EMS57,EMS115,EMS141,EMS72,0
194	073224	076446	076363	076523	EMT300: .WORD	EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS70,0
195	073244	077052	076446	076363	EMT301: .WORD	EMS71,EMS56,EMS53,EMS57,EMS115,EMS150,EMS152,EMS72,0
196	073266	101366	077216	077321	EMT302: .WORD	EMS165,EMS100,EMS103,EMS57,0
197	073300	101366	077244	077321	EMT303: .WORD	EMS165,EMS101,EMS103,EMS57,0
198	073312	101366	077170	077321	EMT304: .WORD	EMS165,EMS77,EMS103,EMS57,0
199	073324	076216	075454	076657	EMT305: .WORD	EMS46,EMS30,EMS64,EMS57,0
200	073336	076274	076363	076523	EMT306: .WORD	EMS50,EMS53,EMS57,0
201	073346	076274	100761	077625	EMT307: .WORD	EMS50,EMS146,EMS115,EMS52,EMS117,EMS46,EMS57,0
202	073366	100657	100674	076363	EMT310: .WORD	EMS142,EMS143,EMS53,EMS57,0
203	073400	077330	077321	076363	EMT311: .WORD	EMS104,EMS103,EMS53,EMS57,0
204	073412	077357	077321	076363	EMT312: .WORD	EMS105,EMS103,EMS53,EMS57,0
205	073424	100722	100732	077321	EMT313: .WORD	EMS144,EMS145,EMS103,EMS57,EMS115,EMS143,EMS70,0
206	073444	076022	077321	076363	EMT314: .WORD	EMS42,EMS103,EMS53,EMS57,0
207	073456	075464	077321	076363	EMT315: .WORD	EMS31,EMS103,EMS53,EMS57,0
208	073470	077052	075464	077321	EMT316: .WORD	EMS71,EMS31,EMS103,EMS57,0
209	073502	076051	077321	076523	EMT317: .WORD	EMS43,EMS103,EMS57,0
210	073512	076153	077321	076523	EMT320: .WORD	EMS45,EMS103,EMS57,0
211	073522	076100	077321	076523	EMT321: .WORD	EMS44,EMS103,EMS57,0
212	073532	077406	075246	000000	EMT322: .WORD	EMS106,EMS20,0
213	073540	075670	077321	076523	EMT323: .WORD	EMS36,EMS103,EMS57,0
214	073550	101556	075670	077321	EMT324: .WORD	EMS173,EMS36,EMS103,EMS57,0
215	073562	101536	075670	077321	EMT325: .WORD	EMS172,EMS36,EMS103,EMS57,0
216	073574	075112	101573	075134	EMT326: .WORD	EMS13,EMS174,EMS15,EMS35,EMS53,EMS175,0
217	073612	101000	101033	077112	EMT327: .WORD	EMS147,EMS151,EMS74,EMS175,0
218	073624	076743	076363	101601	EMT330: .WORD	EMS66,EMS53,EMS175,0
219	073634	075542	077321	076363	EMT331: .WORD	EMS33,EMS103,EMS53,EMS175,0
220	073646	075745	077321	076363	EMT332: .WORD	EMS40,EMS103,EMS53,EMS57,0
221	073660	076323	077112	076523	EMT333: .WORD	EMS51,EMS74,EMS57,EMS115,EMS50,EMS70,0
222	073676	077131	100631	076523	EMT334: .WORD	EMS75,EMS141,EMS57,EMS115,EMS47,EMS73,0
223	073714	077131	101015	101057	EMT335: .WORD	EMS75,EMS150,EMS152,EMS57,EMS115,EMS56,EMS73,0
224	073734	076551	076564	076613	EMT336: .WORD	EMS60,EMS61,EMS62,0
225	073744	077630	077660	075570	EMT337: .WORD	EMS116,EMS117,EMS34,0
226	073754	075570	076363	076375	EMT340: .WORD	EMS34,EMS53,EMS54,EMS111,0
227	073766	076417	075570	000000	EMT341: .WORD	EMS55,EMS34,0
228	073774	076351	077660	076446	EMT342: .WORD	EMS52,EMS117,EMS56,EMS57,0

229	074006	076351	077660	076153	EMT343: .WORD	EMS52,EMS117,EMS45,EMS57,0
230	074020	076351	077660	076100	EMT344: .WORD	EMS52,EMS117,EMS44,EMS57,0
231	074032	076351	077660	102377	EMT345: .WORD	EMS52,EMS117,EMS221,0
232	074042	077406	075246	077625	EMT346: .WORD	EMS106,EMS20,EMS115,EMS223,EMS72,0
233	074056	076351	102054	102447	EMT347: .WORD	EMS52,EMS205,EMS222,EMS206,0
234	074070	077131	101015	077272	EMT350: .WORD	EMS75,EMS150,EMS102,EMS115,EMS56,EMS73,0
235	074106	101427	077075	077272	EMT351: .WORD	EMS167,EMS73,EMS102,0
236	074116	102253	000000		EMT352: .WORD	EMS215,0
237	074122	102324	102013	102274	EMT353: .WORD	EMS217,EMS203,EMS216,0
238	074132	102377	075246	000000	EMT354: .WORD	EMS221,EMS20,0

1	074140	103004	103610	103665	EHT1:	.WORD	EH1,STSH1,STSH2,STSH4,0
2	074152	103610	103665	104012	EHT2:	.WORD	STSH1,STSH2,STSH4,0
3							
4	074162	103023	000000		EHT110:	.WORD	EH110,0
5	074166	103032	000000		EHT111:	.WORD	EH111,0
6							
7	074172	103051	000000		EHT114:	.WORD	EH114,0
8	074176	103100	103610	103665	EHT223:	.WORD	EH223,STSH1,STSH2,STSH4,0
9	074210	103126	103610	103665	EHT256:	.WORD	EH256,STSH1,STSH2,STSH4,0
10							
11	074222	103202	103610	103665	EHT336:	.WORD	EH336,STSH1,STSH2,STSH4,0
12	074234	103241	103610	103665	EHT337:	.WORD	EH337,STSH1,STSH2,STSH4,0
13	074246	103376	103610	103665	EHT344:	.WORD	EH344,STSH1,STSH2,STSH4,0
14							
15	074260	103534	000000		EHT353:	.WORD	EH353,0

1	074264	104050	104144	104162	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
2	074274	104144	104162	104214	EDT2:	.WORD	STSD1,STSD2,STSD4
3							
4	074302	104056			EDT110:	.WORD	ED110
5	074304	104062			EDT111:	.WORD	ED111
6							
7	074306	104070			EDT114:	.WORD	ED114
8	074310	104100	104144	104162	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
9							
10	074320	104110	104144	104162	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
11	074330	104122	104144	104162	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
12	074340	104122	104144	104162	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
13							
14	074352	104134			EDT353:	.WORD	ED353

1	074354	104227	104245	104245	EFT1:	.WORD	EF111,STSF,STSF,STSF
2	074364	104245	104245	104245	EFT2:	.WORD	STSF,STSF,STSF
3							
4	074372	104226			EFT110:	.WORD	EF110
5	074374	104227			EFT111:	.WORD	EF111
6							
7	074376	104231			EFT114:	.WORD	EF114
8	074400	104231	104245	104245	EFT223:	.WORD	EF114,STSF,STSF,STSF
9							
10	074410	104234	104245	104245	EFT336:	.WORD	EF336,STSF,STSF,STSF
11	074420	104234	104245	104245	EFT337:	.WORD	EF336,STSF,STSF,STSF
12	074430	104234	104245	104245	EFT344:	.WORD	EF336,STSF,STSF,STSF
13							
14	074440	104231			EFT353:	.WORD	EF114

		.SBTTL		ERROR MESSAGE STRINGS	
1					
2					
3	074442	127	122	117	EMS1: .ASCIZ @WRONG UNIT SELECTED (RMCS2, BITS 0-2) a
4	074511	104	105	126	EMS2: .ASCIZ @DEVICE WENT a
5	074526	125	116	101	EMS3: .ASCIZ @UNAVAILABLE 'DVA' (RMCS1, BIT 11) a
6	074571	116	117	116	EMS4: .ASCIZ @NONEXISTENT 'NED' (RMCS2, BIT 12) a
7	074634	103	117	115	EMS5: .ASCIZ @COMMAND NOT COMPLETED, a
8	074664	103	117	116	EMS6: .ASCIZ @CONTROLLER NOT READY (RMCS1, BIT 7) a
9	074731	104	122	111	EMS7: .ASCIZ @DRIVE NOT READY 'DRY' (RMDS, BIT 7) a
10	074776	107	117	040	EMS10: .ASCIZ @GO NOT RESET 'GO' (RMCS1, BIT 0) a
11	075040	111	116	126	EMS11: .ASCIZ @INVALID a
12	075051	106	125	116	EMS12: .ASCIZ @FUNCTION CODE (RMCS1, BITS 1-5) a
13	075112	115	101	123	EMS13: .ASCIZ @MASSBUS a
14	075123	103	117	116	EMS14: .ASCIZ @CONTROL a
15	075134	102	125	123	EMS15: .ASCIZ @BUS PARITY ERROR a
16	075156	042	115	103	EMS16: .ASCIZ @'MCPE' (RMCS1, BIT 13) a
17	075206	124	122	101	EMS17: .ASCIZ @TRANSFER ERROR (RMCS1, BIT 14) a
18	075246	123	110	117	EMS20: .ASCIZ @SHOULD NOT BE SET a
19	075271	127	117	122	EMS21: .ASCIZ @WORD COUNT (RMWC) a
20	075314	102	125	123	EMS22: .ASCIZ @BUS (RMBA) a
21	075330	042	114	102	EMS23: .ASCIZ @'LBT' (RMDS, BIT 10) a
22	075356	042	101	117	EMS24: .ASCIZ @'AOE' (RMER1, BIT 09) a
23	075405	104	111	123	EMS25: .ASCIZ @DISK (RMDA) a
24	075422	103	131	114	EMS26: .ASCIZ @CYLINDER (RMDC) a
25	075443	101	104	104	EMS27: .ASCIZ @ADDRESS a
26	075454	123	124	101	EMS30: .ASCIZ @STATUS a
27	075464	042	127	114	EMS31: .ASCIZ @'WLE' (RMER1, BIT 11) a
28	075513	042	125	120	EMS32: .ASCIZ @'UPE' (RMCS2, BIT 13) a
29	075542	042	127	103	EMS33: .ASCIZ @'WCF' (RMER1, BIT 5) a
30	075570	127	122	111	EMS34: .ASCIZ @WRITE CHECK ERROR-'WCE' (RMCS2, BIT 14) a
31	075641	042	115	104	EMS35: .ASCIZ @'MDPE' (RMCS2, BIT 8) a
32	075670	042	104	103	EMS36: .ASCIZ @'DCK' (RMER1, BIT 15) a
33	075717	042	105	103	EMS37: .ASCIZ @'ECH' (RMER1, BIT 6) a
34	075745	042	104	114	EMS40: .ASCIZ @'DLT' (RMCS2, BIT 15) a
35	075774	042	115	130	EMS41: .ASCIZ @'MXF' (RMCS2, BIT 9) a
36	076022	042	104	124	EMS42: .ASCIZ @'DTE' (RMER1, BIT 12) a
37	076051	042	110	103	EMS43: .ASCIZ @'HCRC' (RMER1, BIT 8) a
38	076100	110	105	101	EMS44: .ASCIZ @HEADER COMPARE ERROR 'HCE' (RMER1, BIT 7) a
39	076153	106	117	122	EMS45: .ASCIZ @FORMAT ERROR 'FER' (RMER1, BIT 4) a
40	076216	042	111	101	EMS46: .ASCIZ @'IAE' (RMER1, BIT 10) a
41	076245	042	117	120	EMS47: .ASCIZ @'OPI' (RMER1, BIT 13) a
42	076274	042	123	113	EMS50: .ASCIZ @'SKI' (RMER2, BIT 14) a
43	076323	042	120	111	EMS51: .ASCIZ @'PIP' (RMDS, BIT 13) a
44	076351	124	110	105	EMS52: .ASCIZ @THE RM80 a
45	076363	104	105	124	EMS53: .ASCIZ @DETECTED a
46	076375	101	124	040	EMS54: .ASCIZ @AT AN UNEXPECTED a
47	076417	111	116	103	EMS55: .ASCIZ @INCORRECT DATA DURING a
48	076446	111	116	126	EMS56: .ASCIZ @INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
49	076523	104	125	122	EMS57: .ASCIZ @DURING DATA TRANSFER a
50	076551	104	101	124	EMS60: .ASCIZ @DATA READ a
51	076564	104	117	105	EMS61: .ASCIZ @DOES NOT COMPARE WITH a
52	076613	104	101	124	EMS62: .ASCIZ @DATA WRITTEN a
53	076631	042	120	101	EMS63: .ASCIZ @'PAR' (RMER1, BIT 3) a
54	076657	111	123	040	EMS64: .ASCIZ @IS INCORRECT a
55	076675	103	117	115	EMS65: .ASCIZ @COMPOSITE ERROR 'ERR' (RMDS, BIT 14) a
56	076743	104	101	124	EMS66: .ASCIZ @DATA PARITY ERROR 'DPE' (RMER2, BIT 3) a
57	077013	104	125	122	EMS67: .ASCIZ @DURING SEEK COMMAND a

58	077040	111	123	040	EMS70:	.ASCIZ	@IS RESET @
59	077052	105	122	122	EMS71:	.ASCIZ	@ERRONEOUS @
60	077065	111	123	040	EMS72:	.ASCIZ	@IS SET @
61	077075	104	111	104	EMS73:	.ASCIZ	@DID NOT SET @
62	077112	104	111	104	EMS74:	.ASCIZ	@DID NOT RESET @
63	077131	114	117	123	EMS75:	.ASCIZ	@LOST @
64	077137	104	125	122	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
65	077170	042	122	115	EMS77:	.ASCIZ	@'RMR' (RMR1, BIT 2) @
66	077216	042	111	114	EMS100:	.ASCIZ	@'ILR' (RMR1, BIT 1) @
67	077244	042	111	114	EMS101:	.ASCIZ	@'ILF' (RMR1, BIT 0) @
68	077272	104	125	122	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
69	077321	105	122	122	EMS103:	.ASCIZ	@ERROR @
70	077330	042	114	102	EMS104:	.ASCIZ	@'LBC' (RMR2, BIT 10) @
71	077357	042	114	123	EMS105:	.ASCIZ	@'LSC' (RMR2, BIT 11) @
72	077406	110	105	101	EMS106:	.ASCIZ	@HEADER ERRORS @
73	077425	102	125	123	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
74	077454	101	104	104	EMS111:	.ASCIZ	@ADDRESS @
75	077465	127	110	105	EMS112:	.ASCII	@WHEN READING/WRITING RH REGISTERS @
76	077527	101	124	040	.ASCIZ	@AT THE FOLLOWING @	
77	077551	124	110	105	EMS113:	.ASCIZ	@THE SELECTED DEVICE IS @
78	077601	116	117	116	EMS114:	.ASCIZ	@NONEXISTENT DEVICE @
79	077625	015	012	000	EMS115:	.ASCIZ	<CR><LF>
80	077630	124	110	105	EMS116:	.ASCIZ	@THE MASSBUS CONTROLLER @
81	077660	106	101	111	EMS117:	.ASCIZ	@FAILED TO DETECT @
82	077702				EMS120:		:NOT USED
83	077702	103	117	116	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
84	077745	102	125	123	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMAA, @
85	100002	103	117	116	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
86	100045	105	122	122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMR1, @
87	100077	101	124	124	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMAA, @
88	100142	115	101	111	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMR #1, @
89	100205	105	103	103	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
90	100244	105	103	103	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2, @
91	100302	115	101	111	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMR2, @
92	100342	116	117	124	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
93	100366	125	116	111	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
94	100411	103	117	116	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
95	100453	122	110	040	EMS135:	.ASCIZ	@RH CONTROLLER ERROR CLEAR (RMCS1, BIT 14) @
96	100526	104	122	111	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
97	100553	104	122	111	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMDA @
98	100610	115	105	104	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
99	100631	042	115	117	EMS141:	.ASCIZ	@'MOL' (RMDA, BIT 12) @
100	100657	104	122	111	EMS142:	.ASCIZ	@DRIVE FAULT @
101	100674	042	104	126	EMS143:	.ASCIZ	@'DVC' (RMR2, BIT 7) @
102	100722	125	116	123	EMS144:	.ASCIZ	@UNSAFE @
103	100732	042	125	116	EMS145:	.ASCIZ	@'UNS' (RMR1, BIT 14) @
104	100761	123	110	117	EMS146:	.ASCIZ	@SHOULD BE SET @
105	101000	117	106	106	EMS147:	.ASCIZ	@OFFSET MODE @
106	101015	126	117	114	EMS150:	.ASCIZ	@VOLUME VALID @
107	101033	042	117	115	EMS151:	.ASCIZ	@'DM' (RMDA, BIT 0) @
108	101057	042	126	126	EMS152:	.ASCIZ	@'VV' (RMDA, BIT 6) @
109	101103	120	101	103	EMS153:	.ASCIZ	@PACK ACK COMMAND @
110	101125	116	117	124	EMS154:	.ASCIZ	@NOT SET BY @
111	101141	117	106	106	EMS155:	.ASCIZ	@OFFSET COMMAND @
112	101161	116	117	124	EMS156:	.ASCIZ	@NOT RESET BY @
113	101177	122	124	103	EMS157:	.ASCIZ	@RTC COMMAND @
114	101214	122	111	120	EMS160:	.ASCIZ	@RIP COMMAND @

ERROR MESSAGE STRINGS

115	101231	117	106	106	EMS161:	.ASCIZ	@OFFSET REGISTER (RMOF) @
116	101261				EMS162:		:@UNUSED@
117	101261	104	125	122	EMS163:	.ASCIZ	@DURING RECALIBRATE @
118	101305	111	123	040	EMS164:	.ASCII	@IS INTERMITTENT OR DRIVE DIDNT DROP ON @
119	101354	103	131	114		.ASCIZ	@CYLINDER @
120	101366	125	116	105	EMS165:	.ASCIZ	@UNEXPECTED @
121	101402	122	105	103	EMS166:	.ASCIZ	@RECALIBRATE COMMAND @
122	101427	042	101	124	EMS167:	.ASCIZ	@'ATA' (RMDS, BIT15) @
123	101454	127	110	105	EMS170:	.ASCIZ	@WHEN READING REGISTER @
124	101503	105	122	122	EMS171:	.ASCIZ	@ERROR REGISTER #2, RMER2, @
125	101536	116	117	116	EMS172:	.ASCIZ	@NONRECOVERABLE @
126	101556	122	105	103	EMS173:	.ASCIZ	@RECOVERABLE @
127	101573	104	101	124	EMS174:	.ASCIZ	@DATA @
128	101601	104	125	122	EMS175:	.ASCIZ	@DURING WRITE COMMAND @
129	101627	042	117	120	EMS176:	.ASCIZ	@'DPE' (RMER2, BIT 13) @
130	101656	111	116	040	EMS177:	.ASCIZ	@IN WRITE PROTECT @
131	101700	103	101	116	EMS200:	.ASCIZ	@CAN NOT SET @
132	101715	104	111	101	EMS201:	.ASCIZ	@DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 0) @
133	101763	104	125	122	EMS202:	.ASCIZ	@DURING DIAGNOSTIC MODE @
134	102013	111	116	103	EMS203:	.ASCIZ	@INCORRECT @
135	102026	042	127	122	EMS204:	.ASCIZ	@'WRL' (RMDS, BIT 11) @
136	102054	105	130	105	EMS205:	.ASCIZ	@EXECUTED @
137	102066	127	111	124	EMS206:	.ASCIZ	@WITH COMP ERROR SET @
138	102113	042	107	117	EMS207:	.ASCIZ	@'GO' (RMCS1, BIT 0) @
139	102140	127	122	111	EMS210:	.ASCIZ	@WRITING @
140	102151	127	101	123	EMS211:	.ASCIZ	@WAS RESET BY @
141	102167	120	122	117	EMS212:	.ASCIZ	@PROGRAM INTERRUPT @
142	102212	127	101	123	EMS213:	.ASCIZ	@WAS NOT GENERATED @
143	102235	123	105	105	EMS214:	.ASCIZ	@SEEK COMMAND @
144	102253	120	122	117	EMS215:	.ASCIZ	@PROGRAM TIMEOUT @
145	102274	104	125	122	EMS216:	.ASCIZ	@DURING LOOK AHEAD TEST @
146	102324	1 4	117	117	EMS217:	.ASCIZ	@LOOK AHEAD REGISTER,RMLA, @
147	102357	123	105	101	EMS220:	.ASCIZ	@SEARCH COMMAND @
148	102377	102	101	104	EMS221:	.ASCIZ	@BAD SECTOR ERROR 'BSE' (RMER2, BIT 15) @
149	102447	101	040	104	EMS222:	.ASCIZ	@A DATA TRANSFER COMMAND @
150	102500	110	105	101	EMS223:	.ASCIZ	@HEADER COMPARE INHIBIT 'HCI' (RMOF, BIT 10) @
151	102555	116	117	116	EMS224:	.ASCIZ	@NONEXISTENT MEMORY 'NEM' (RMCS2, BIT 11) @
152	102627	123	113	111	EMS225:	.ASCIZ	@SKIP SECTOR ERROR @
153	102652	042	123	123	EMS226:	.ASCIZ	@'SSE' (RMER2, BIT 05) @
154	102701	111	116	110	EMS227:	.ASCIZ	@INHIBIT @
155	102712	042	123	123	EMS230:	.ASCIZ	@'SSEI' (RMOF, BIT 09) @
156	102741	105	116	104	EMS231:	.ASCIZ	@END OF TRACK @
157	102757	101	106	124	EMS232:	.ASCIZ	@AFTER DATA TRANSFER @

1	103004	105	130	120	EH1:	.ASCIZ	@EXPCTD	RECEVD@			
2	103023	102	125	123	EH110:	.ASCIZ	@BUSADRA				
3	103032	040	122	115	EH111:	.ASCIZ	@ RMCS2	RMCS1@			
4											
5	103051	122	105	103	EH114:	.ASCIZ	@RECEVD	SNGPRT	DULPRT@		
6	103100	105	130	120	EH223:	.ASCIZ	@EXPCTD	RECEVD	DATA@		
7	103126	105	130	120	EH256:	.ASCII	@EXPCTD	RECEVD	RGSTR@<CRLF>		
8	103154	123	124	101		.ASCIZ	@STATUS	STATUS	INDEX@		
9											
10	103202	107	104	101	EH336:	.ASCIZ	@GDADRS	GDDATA	BDADRS	BDDATA@	
11	103241	122	115	103	EH337:	.ASCII	@RMCS2	STATUS	FAILING	DATA@<CRLF>	
12	103300	137	137	137		.ASCII	@			@<CRLF>	
13	103337	105	130	120		.ASCIZ	@EXPCTD	RECEVD	--BIT--	ADRESS@	
14											
15	103376	122	115	105	EH344:	.ASCII	@RMER1	STATUS	HEADER	FAILING@<CRLF>	
16	103436	137	137	137		.ASCII	@		WORD	BIT@<CRLF>	
17	103474	105	130	120		.ASCIZ	@EXPCTD	RECEVD	NUMBER	POSITON@	
18	103534	105	130	120	EH353:	.ASCII	@EXPCTD	RECEVD@<CRLF>			
19	103553	074	103	122		.ASCIZ	@<CRLF>	RMLA	RMLA	RMOF @	
20											
21	103610	040	122	115	STSH1:	.ASCII	@ RMCS1	RMCS2	RMDS	RMER1	RMER2@
22	103655	040	040	040		.ASCIZ	@	RMAS@			
23	103665	040	122	115	STSH2:	.ASCII	@ RMWC	RMBA	RMDA	RMOF	RMDC@
24	103732	040	040	040		.ASCIZ	@	RMEC1	RMEC2@		
25	103754	040	122	115	STSH3:	.ASCIZ	@ RMDA	RMDC	RMOF	RMLA@	
26	104012	040	122	115	STSH4:	.ASCIZ	@ RMMR1	RMMR2	RMDT	RMSN@	
27						.EVEN					

1	104050	001140	001142	000000	ED1:	.WORD	SGDDAT,SBDDAT,0
2	104056	001276	000000		ED110:	.WORD	SBASE,0
3	104062	001174	001176	000000	ED111:	.WORD	STMP0,STMP1,0
4							
5	104070	001364	001176	001200	ED114:	.WORD	RMDTI,STMP1,STMP2,0
6	104100	001140	001142	001174	ED223:	.WORD	SGDDAT,SBDDAT,STMP0,0
7							
8	104110	001134	001140	001136	ED336:	.WORD	SGDADR,SGDDAT,SBADR,SBDDAT,0
9							
10	104122	001140	001142	001174	ED337:	.WORD	SGDDAT,SBDDAT,STMP0,STMP1,0
11	104134	001140	001142	001444	ED353:	.WORD	SGDDAT,SBDDAT,RMOFO,0
12							
13	104144	001336	001346	001350	STSD1:	.WORD	RMCS1I,RMCS2I,RMDSI,RMER1I,RMER2I,RMASI,0
14	104162	001340	001342	001344	STSD2:	.WORD	RMWCI,RMBAI,RMDAI,RMOFI,RMDCI,RMEC1I
15	104176	001404	000000			.WORD	RMEC2I,0
16	104202	001344	001372	001370	STSD3:	.WORD	RMDAI,RMDCI,RMOFI,RMLAI,0
17	104214	001362	001376	001364	STSD4:	.WORD	RMMR1I,RMMR2I,RMDTI,RMSNI,0

1	104226	000			EF110:	.BYTE	0
2	104227	000	000		EF111:	.BYTE	0,0
3	104231	000	000	000	EF114:	.BYTE	0,0,0
4	104234	000	000	000	EF336:	.BYTE	0,0,0,0
5	104240	000	000	000	EF337:	.BYTE	0,0,0,0,0
6							
7	104245	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0
8					.EVEN		


```

1          :STORAGE FOR GENERAL DATA TRANSFERRS
2 104254   BUFFER:
3 104254   BUFSIZE: .BLKW 258.
4 105260   BUFTWO: .BLKW 258.
5
6          :STORAGE FOR MANUFACTURES 16 BIT MODE BAD SECTOR FILE
7 106264   000000 000000 MFGFIL: .WORD 0,0 :2 HEADER WORDS
8 106270   .BLKW 256. :256. WORDS OF DATA
9 107270   177777 .WORD -1 :TERMINATOR IF FILE IS FULL
10
11         :STORAGE FOR USERS 16 BIT MODE BAD SECTOR FILE
12 107272   000000 000000 USRFIL: .WORD 0,0 :2 HEADER WORDS
13 107276   .BLKW 256. :256. WORDS OF DATA
14 110276   177777 .WORD -1 :TERMINATOR IF FILE IS FULL
15
16         :STORAGE FOR SKIP SECTOR FILE
17 110300   000000 000000 SSFIL: .WORD 0,0 :2 HEADER WORDS
18 110304   .BLKW 28. :2 CYLINDERS OF SKIP SECTOR ENTRIES
19 110374   177777 .WORD -1 :TERMINATOR
20
21         .-BUFFER
22
23 104254   HELP:
24 104254   200 .ASCII <CRLF>
25 104255   200 .ASCII <CRLF>
26 104256   114 111 123 .ASCII @LIST OF TESTS@<CRLF>
27 104274   055 055 055 .ASCII @-----@<CRLF>
28 104312   124 061 011 .ASCII @T1 CONTROLLER ACCESS TEST@<CRLF>
29 104344   124 062 011 .ASCII @T2 WRITE, READ ZEROS@<CRLF>
30 104371   124 063 011 .ASCII @T3 WRITE, WRITE CHECK ZEROS@<CRLF>
31 104425   124 064 011 .ASCII @T4 WRITE, WRITE CHECK ZEROS W/ WCE ERROR@<CRLF>
32 104476   124 065 011 .ASCII @T5 WRITE, READ ONES@<CRLF>
33 104522   124 066 011 .ASCII @T6 WRITE, WRITE CHECK ONES@<CRLF>
34 104555   124 067 011 .ASCII @T7 WRITE, WRITE CHECK ONES W/ WCE ERROR@<CRLF>
35 104625   124 061 060 .ASCII @T10 WRITE, WRITE CHECK MULTIPLE SECTORS@<CRLF>
36 104675   124 061 061 .ASCII @T11 WRITE, READ W/ IMPLIED SEEK@<CRLF>
37 104735   124 061 062 .ASCII @T12 WRITE, WRITE CHECK W/ HEAD SWITCHING@<CRLF>
38 105006   124 061 063 .ASCII @T13 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK@<CRLF>
39 105062   124 061 064 .ASCII @T14 WRITE, READ W/ HCE ERROR@<CRLF>
40 105117   124 061 065 .ASCII @T15 WRITE, READ W/ HCI & SSEI@<CRLF>
41 105155   124 061 066 .ASCII @T16 WRITE, READ W/ IVC ERROR@<CRLF>
42 105212   124 061 067 .ASCII @T17 WRITE, READ W/ ABORT@<CRLF>
43 105243   124 062 060 .ASCII @T20 WRITE, READ EARLY PEAK SHIFT@<CRLF>
44 105304   124 062 061 .ASCII @T21 WRITE, READ EACH TRACK W/ MIXED PEAK SHIFT@<CRLF>
45 105363   124 062 062 .ASCII @T22 READ, WRITE CHECK MULTIPLE SECTORS IN OFFSET MODE@<CRLF>
46 105451   124 062 063 .ASCII @T23 FORMAT FE CYLINDERS@<CRLF>
47 105501   200 .ASCII <CRLF>
48 105502   117 120 105 .ASCII @OPERATIONAL SWITCH SETTINGS@<CRLF>
49 105536   055 055 055 .ASCII @-----@<CRLF>
50 105572   123 127 111 .ASCII @SWITCH USE@<CRLF>
51 105607   055 055 055 .ASCII @-----@<CRLF>
52 105644   040 040 061 .ASCII @ 15 HALT ON ERROR@<CRLF>
53 105670   040 040 061 .ASCII @ 14 LOOP ON TEST@<CRLF>
54 105713   040 040 061 .ASCII @ 13 INHIBIT ERROR TYPEOUTS@<CRLF>
55 105750   040 040 061 .ASCII @ 12 @<CRLF>
56 105757   040 040 061 .ASCII @ 11 INHIBIT ITERATIONS@<CRLF>
57 106010   040 040 061 .ASCII @ 10 BELL ON ERROR@<CRLF>
    
```

ERROR MESSAGE STRINGS

58	106034	040	040	040	.ASCII	@	9	LOOP ON ERROR@<CRLF>
59	106060	040	040	040	.ASCII	@	8	LOOP ON TEST IN SWR<7:0>@<CRLF>
60	106117	040	040	040	.ASCII	@	7	TN128@<CRLF>
61	106133	040	040	040	.ASCII	@	6	TN64@<CRLF>
62	106146	040	040	040	.ASCII	@	5	TN32@<CRLF>
63	106161	040	040	040	.ASCII	@	4	TN16@<CRLF>
64	106174	040	040	040	.ASCII	@	3	TN8@<CRLF>
65	106206	040	040	040	.ASCII	@	2	TN4@<CRLF>
66	106220	040	040	040	.ASCII	@	1	TN2@<CRLF>
67	106232	040	040	040	.ASCIZ	@	0	TN1@<CRLF>
68								
69	000200			.END		200		

ABASE = 176700
 ACDW1 = 000000
 ACDW2 = 000000
 ACKSTS 051026
 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 066074
 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
 APTCSU= 000040
 APTENV= 000001
 APTSIZ= 000200
 APTSPO= 000100
 ARGS = 000004
 ASNDA 001520
 ASNDC 001516
 ASWREG= 000000
 ATA = 100000
 ATESTN= 000000

ATNMSK= 000377
 ATNTBL 067344
 AUNIT = 000000
 AUSWR = 000000
 AVECT1= 120254
 AVECT2= 000000
 A16 = 000400
 A17 = 001000
 BACK = 000001
 BADSCT 034762
 BADTMO 005344
 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
 BLNKS1 067241
 BLNKS2 067240
 BLNKS3 067237
 BLNKS4 067236
 BOTADR 063464
 BOTFLG 063466
 BPTVEC= 000014
 BSE = 100000
 BUFFER 104254

BUFONE 104254
 BUFTWO 105260
 CC = 004000
 CH = 002000
 CHGADR 001330
 CHRCNT 063467
 CKSWR = 104410
 CLKADR 001522
 CLKVCT 001524
 CLR = 000040
 CLRSTS 050146
 CMNSTA 007332
 CMPBUF 037462
 CMPERR 046124
 CNSL01 066521
 CNSL02 066531
 CNSL03 066573
 CNSL04 066602
 CNSL07 066636
 CNSL08 067000
 CNSL09 067001
 CNTCLR 050030
 COMMA 066105
 CONT = 000100
 CPSAVE 062640
 CR = 000015
 CRLF = 000200
 CTLFG 001326
 CYLMSK= 001777
 DBCK = 100000
 DBEN = 040000
 DBL = 002000
 DCK = 100000
 DDISP = 177570
 DEBL = 020000
 DEVSEL 046336
 DISPLA 001156
 DISPRE 000174
 DLT = 100000
 DMD = 000001
 DPE = 000010
 DPEHI = 040000
 DPELO = 020000
 DPR = 000400
 DRIVES 067176
 DRQ = 004000
 DRVCLR= 000010
 DRVSTS 053364
 DRY = 000200
 DSWR = 177570
 DTASTS 054166
 DTE = 010000
 DTO = 010000
 DULPRT= 024026
 DVA = 004000
 DVC = 000200
 EARLY 067670

EBL = 020000
 ECH = 000100
 ECI = 004000
 ECRC = 001000
 EDT1 074264
 EDT110 074302
 EDT111 074304
 EDT114 074306
 EDT2 074274
 EDT223 074310
 EDT336 074320
 EDT337 074330
 EDT344 074340
 EDT353 074352
 ED1 104050
 ED110 104056
 ED111 104062
 ED114 104070
 ED223 104100
 ED336 104110
 ED337 104122
 ED353 104134
 EECC = 000020
 EFT1 074354
 EFT110 074372
 EFT111 074374
 EFT114 074376
 EFT2 074364
 EFT223 074400
 EFT336 074410
 EFT337 074420
 EFT344 074430
 EFT353 074440
 EF110 104226
 EF111 104227
 EF114 104231
 EF336 104234
 EF337 104240
 EHT1 074140
 EHT110 074162
 EHT111 074166
 EHT114 074172
 EHT2 074152
 EHT223 074176
 EHT256 074210
 EHT336 074222
 EHT337 074234
 EHT344 074246
 EHT353 074260
 EH1 103004
 EH110 103023
 EH111 103032
 EH114 103051
 EH223 103100
 EH256 103126
 EH336 103202
 EH337 103241

EH344 103376
 EH353 103534
 EMS1 074442
 EMS10 074776
 EMS100 077216
 EMS101 077244
 EMS102 077272
 EMS103 077321
 EMS104 077330
 EMS105 077357
 EMS106 077406
 EMS11 075040
 EMS110 077425
 EMS111 077454
 EMS112 077465
 EMS113 077551
 EMS114 077601
 EMS115 077625
 EMS116 077630
 EMS117 077660
 EMS12 075051
 EMS120 077702
 EMS121 077702
 EMS122 077745
 EMS123 100002
 EMS124 100045
 EMS125 100077
 EMS126 100142
 EMS127 100205
 EMS13 075112
 EMS130 100244
 EMS131 100302
 EMS132 100342
 EMS133 100366
 EMS134 100411
 EMS135 100453
 EMS136 100526
 EMS137 100553
 EMS14 075123
 EMS140 100610
 EMS141 100631
 EMS142 100657
 EMS143 100674
 EMS144 100722
 EMS145 100732
 EMS146 100761
 EMS147 101000
 EMS15 075134
 EMS150 101015
 EMS151 101033
 EMS152 101057
 EMS153 101103
 EMS154 101125
 EMS155 101141
 EMS156 101161
 EMS157 101177
 EMS16 075156

SYMBOL TABLE

EMS160	101214	EMS33	075542	EMT114	071254	EMT177	072142	EMT260	072712
EMS161	101231	EMS34	075570	EMT115	071256	EMT2	070006	EMT261	072724
EMS162	101261	EMS35	075641	EMT116	071266	EMT20	070136	EMT262	072744
EMS163	101261	EMS36	075670	EMT117	071276	EMT200	072144	EMT263	072754
EMS164	101305	EMS37	075717	EMT12	070062	EMT201	072156	EMT264	072764
EMS165	101366	EMS4	074571	EMT120	071306	EMT202	072170	EMT265	073004
EMS166	101402	EMS40	075745	EMT121	071316	EMT203	072202	EMT266	073024
EMS167	101427	EMS41	075774	EMT122	071326	EMT204	072214	EMT267	073036
EMS17	075206	EMS42	076022	EMT123	071336	EMT205	072232	EMT27	070226
EMS170	101454	EMS43	076051	EMT124	071346	EMT206	072242	EMT270	073054
EMS171	101503	EMS44	076100	EMT125	071356	EMT207	072252	EMT271	073070
EMS172	101536	EMS45	076153	EMT126	071366	EMT21	070146	EMT272	073106
EMS173	101556	EMS46	076216	EMT127	071400	EMT210	072264	EMT273	073124
EMS174	101573	EMS47	076245	EMT13	070070	EMT211	072272	EMT274	073142
EMS175	101601	EMS5	074634	EMT130	071412	EMT212	072306	EMT275	073160
EMS176	101627	EMS50	076274	EMT131	071424	EMT213	072322	EMT276	073172
EMS177	101656	EMS51	076323	EMT132	071436	EMT214	072332	EMT277	073206
EMS2	074511	EMS52	076351	EMT133	071450	EMT215	072352	EMT3	070014
EMS20	075246	EMS53	076363	EMT134	071462	EMT216	072364	EMT30	070236
EMS200	101700	EMS54	076375	EMT135	071474	EMT217	072374	EMT300	073224
EMS201	101715	EMS55	076417	EMT136	071506	EMT22	070156	EMT301	073244
EMS202	101763	EMS56	076446	EMT137	071520	EMT220	072404	EMT302	073266
EMS203	102013	EMS57	076523	EMT14	070102	EMT221	072414	EMT303	073300
EMS204	102026	EMS6	074664	EMT140	071530	EMT222	072424	EMT304	073312
EMS205	102054	EMS60	076551	EMT141	071540	EMT223	072434	EMT305	073324
EMS206	102066	EMS61	076564	EMT142	071550	EMT224	072444	EMT306	073336
EMS207	102113	EMS62	076613	EMT143	071560	EMT225	072460	EMT307	073346
EMS21	075271	EMS63	076631	EMT144	071570	EMT226	072500	EMT31	070246
EMS210	102140	EMS64	076657	EMT145	071600	EMT227	072510	EMT310	073366
EMS211	102151	EMS65	076675	EMT146	071610	EMT23	070166	EMT311	073400
EMS212	102167	EMS66	076743	EMT147	071620	EMT230	072524	EMT312	073412
EMS213	102212	EMS67	077013	EMT15	070110	EMT231	072526	EMT313	073424
EMS214	102235	EMS7	074731	EMT150	071630	EMT232	072530	EMT314	073444
EMS215	102253	EMS70	077040	EMT151	071640	EMT233	072532	EMT315	073456
EMS216	102274	EMS71	077052	EMT152	071652	EMT234	072534	EMT316	073470
EMS217	102324	EMS72	077065	EMT153	071664	EMT235	072536	EMT317	073502
EMS22	075314	EMS73	077075	EMT154	071702	EMT236	072540	EMT32	070256
EMS220	102357	EMS74	077112	EMT155	071720	EMT237	072542	EMT320	073512
EMS221	102377	EMS75	077131	EMT156	071732	EMT24	070176	EMT321	073522
EMS222	102447	EMS76	077137	EMT157	071744	EMT240	072544	EMT322	073532
EMS223	102500	EMS77	077170	EMT16	070116	EMT241	072546	EMT323	073540
EMS224	102555	EMTVEC=	000030	EMT160	071756	EMT242	072550	EMT324	073550
EMS225	102627	EMT1	070002	EMT161	071766	EMT243	072552	EMT325	073562
EMS226	102652	EMT10	070052	EMT162	072000	EMT244	072554	EMT326	073574
EMS227	102701	EMT100	071056	EMT163	072012	EMT245	072556	EMT327	073612
EMS23	075330	EMT101	071074	EMT164	072022	EMT246	072560	EMT33	070266
EMS230	102712	EMT102	071112	EMT165	072030	EMT247	072570	EMT330	073624
EMS231	102741	EMT103	071122	EMT166	072036	EMT25	070206	EMT331	073634
EMS232	102757	EMT104	071134	EMT167	072044	EMT250	072602	EMT332	073646
EMS24	075356	EMT105	071152	EMT17	070126	EMT251	072616	EMT333	073660
EMS25	075405	EMT106	071162	EMT170	072052	EMT252	072624	EMT334	073676
EMS26	075422	EMT107	071172	EMT171	072064	EMT253	072632	EMT335	073714
EMS27	075443	EMT11	070056	EMT172	072074	EMT254	072650	EMT336	073734
EMS3	074526	EMT110	071212	EMT173	072104	EMT255	072660	EMT337	073744
EMS30	075454	EMT111	071224	EMT174	072114	EMT256	072670	EMT34	070276
EMS31	075464	EMT112	071232	EMT175	072126	EMT257	072700	EMT340	073754
EMS32	075513	EMT113	071240	EMT176	072134	EMT26	070216	EMT341	073766

SYMBOL TABLE

EMT342	073774	ERTY00	063470	ILRG60=	000060	OFD	=	000200	RMAS	=	000016			
EMT343	074006	ERTY01	063475	ILRG62=	000062	OFFSET=		000014	RMASI		001354			
EMT344	074020	ERTY02	063505	ILRG64=	000064	OM	=	000001	RMASO		001430			
EMT345	074032	ERTY03	063514	ILRG66=	000066	ONES		067414	RMBA	=	000004			
EMT346	074042	ERTY04	063522	ILRG70=	000070	OPE	=	020000	RMBAE	=	000050			
EMT347	074056	ERTY05	063525	ILRG72=	000072	OPI	=	020000	RMBAEI		001406			
EMT35	070306	ESRC	=	004000	ILRG74=	000074	OR	=	000200	RMBAEO		001462		
EMT350	074070	FER	=	000020	ILRG76=	000076	PACACK=		000022	RMBAI		001342		
EMT351	074106	FIND	=	000001	IOTVEC=	000020	PAKACK=		000022	RMBAO		001416		
EMT352	074116	FMT16	=	010000	IPCK0	=	000001	PAR	=	000010	RMCS1	=	000000	
EMT353	074122	FNCDTB		067244	IPCK1	=	000002	PAT	=	000020	RMCS1I		001336	
EMT354	074132	FNCMSK=		000077	IPCK2	=	000004	PDA	=	000400	RMCS10		001412	
EMT36	070316	F0	=	000002	IPCK3	=	000010	PFECH		063532	RMCS2	=	000010	
EMT37	070326	F1	=	000004	IR	=	000100	PFECH1		063542	RMCS2I		001346	
EMT4	070022	F2	=	000010	IVC	=	010000	PFECH2		063630	RMCS20		001422	
EMT40	070336	F3	=	000020	LBC	=	002000	PFECH3		063446	RMCS3	=	000052	
EMT41	070344	F4	=	000040	LBT	=	002000	PFECH4		063654	RMCS3I		001410	
EMT42	070354	GENBUF		037214	LF	=	000012	PGE	=	002000	RMCS30		001464	
EMT43	070366	GET		040110	LODEV		067104	PGM	=	001000	RMDA	=	000006	
EMT44	070376	GETBUF		001336	LS	=	000004	PHA	=	000200	RMDAI		001344	
EMT45	070406	GETINX		001526	LSC	=	004000	PIP	=	020000	RMDAO		001420	
EMT46	070416	GETSSF		037070	LST	=	000002	PIRQ	=	177772	RMDB	=	000022	
EMT47	070426	GETSTS		040024	LSTRK		001334	PIRQE=		000240	RMDBI		001360	
EMT5	070030	GO	=	000001	MCLK	=	004000	PLFS	=	002000	RMDBO		001434	
EMT50	070434	GTSWR	=	104407	MCPE	=	020000	PRIERR		041126	RMDC	=	000034	
EMT51	070444	HCE	=	000200	MDF	=	000100	PRO	=	000000	RMDCI		001372	
EMT52	070456	HCI	=	002000	MDPE	=	000400	PR1	=	000040	RMDCO		001446	
EMT53	070474	HCRC	=	000400	MEDENB		001512	PR2	=	000100	RMDS	=	000012	
EMT54	070512	HELP		104254	MFGFIL		106264	PR3	=	000140	RMDSI		001350	
EMT55	070522	HT	=	000011	MI	=	000004	PR4	=	000200	RMDSO		001424	
EMT56	070534	IAE	=	002000	MIXED		067354	PR5	=	000240	RMDT	=	000026	
EMT57	070552	IBSAVE		062642	MOC	=	000400	PR6	=	000300	RMDTI		001364	
EMT6	070036	IDXMSK=		000077	MOH	=	020000	PR7	=	000340	RMDTO		001440	
EMT60	070562	IE	=	000100	MOL	=	010000	PS	=	177776	RMEC1	=	000044	
EMT61	070600	ILF	=	000001	MRD	=	002000	PSEL	=	002000	RMEC1I		001402	
EMT62	070620	ILF02	=	000002	MS	=	000040	PSW	=	177776	RMEC10		001456	
EMT63	070634	ILF24	=	000024	MSC	=	000002	PUT		040360	RMEC2	=	000046	
EMT64	070644	ILF26	=	000026	MSTRVS		067022	PUTBUF		001412	RMEC2I		001404	
EMT65	070664	ILF30	=	000030	MSE	=	100000	PUTINX		001555	RMEC20		001460	
EMT66	070704	ILF32	=	000032	MSER	=	000200	PWRVEC=		000024	RMER1	=	000014	
EMT67	070716	ILF34	=	000034	MSGDRV		067036	QUES		066101	RMER1I		001352	
EMT7	070044	ILF36	=	000036	MSHELP		066110	RCLSTS		051622	RMER10		001426	
EMT70	070730	ILF40	=	000040	MUR	=	001000	RD	=	000070	RMER2	=	000042	
EMT71	070740	ILF42	=	000042	MWD	=	000010	RDCHR	=	104411	RMER2I		001400	
EMT72	070750	ILF44	=	000044	MWP	=	000010	RDLIN	=	104412	RMER20		001454	
EMT73	070766	ILF46	=	000046	MXF	=	001000	RDOCT	=	104413	RMHR	=	000036	
EMT74	071004	ILF54	=	000054	N		067232	RDY	=	000200	RMHRI		001374	
EMT75	071014	ILF56	=	000056	NDTMSK=		115760	READY		007524	RMHRO		001450	
EMT76	071034	ILF64	=	000064	NED	=	010000	RECAL	=	000006	RMLA	=	000020	
EMT77	071044	ILF66	=	000066	NEM	=	004000	RESREG=		104415	RMLAI		001356	
ENRGDT	070002	ILF74	=	000074	NONE		067225	RESVEC=		000010	RMLAO		001432	
EQUALS	066072	ILF76	=	000076	NOP	=	000000	REX	=	010000	RMR1	=	000024	
ERR	=	040000	ILR	=	000002	NOTAVL		067136	RG	=	040000	RMR1I		001362
ERRNMB		063462	ILRG50=		000050	NOTPRS		067121	RGDTPT		067354	RMR10		001436
ERROR	=	104000	ILRG52=		000052	NOTRM		067067	RH	=	000072	RMR2	=	000040
ERRTYP		062644	ILRG54=		000054	NSA	=	100000	RIP	=	000020	RMR2I		001376
ERRVEC=		000004	ILRG56=		000056	OCC	=	100000	RELEASE=		000012	RMR20		001452

SYMBOL TABLE

RMOF = 000032
 RMOF1 001370
 RMOFO 001444
 RMR = 000004
 RMSN = 000030
 RMSNI 001366
 RMSNO 001442
 RMWC = 000002
 RMWCI 001340
 RMWCO 001414
 RQA = 100000
 RQB = 040000
 RTC = 000016
 R6 = %000006
 R7 = %000007
 SADMSK= 000377
 SAVREG= 104414
 SA1 = 000001
 SA16 = 000020
 SA2 = 000002
 SA4 = 000004
 SA8 = 000010
 SC = 100000
 SCOPE = 000004
 SCTMSG 066006
 SCTMSK= 003700
 SCO = 000100
 SC1 = 000200
 SC2 = 000400
 SC3 = 001000
 SC4 = 002000
 SEARCH= 000030
 SECERR 041760
 SEEK = 000004
 SEKSTS 046550
 SHUT 033776
 SHUT2 065130
 SIZCLK 040610
 SKI = 040000
 SNGPRT= 020026
 SSE = 000040
 SSEI = 001000
 SSF = 020000
 SSFENB 001514
 SSFIL 110300
 STACK = 001100
 STANDA 006654
 START 005434
 START1 005424
 START2 005440
 STCDRV 057606
 STIMER 041074
 STKLMT= 177774
 STOP 062222
 STSD1 104144
 STSD2 104162
 STSD3 104202

STSD4 104214
 STSF 104245
 STSH1 103610
 STSH2 103665
 STSH3 103754
 STSH4 104012
 SWR 001154
 SWREG 000176
 SWO = 000001
 SW00 = 000001
 SW01 = 000002
 SW02 = 000004
 SW03 = 000010
 SW04 = 000020
 SW05 = 000040
 SW06 = 000100
 SW07 = 000200
 SW08 = 000400
 SW09 = 001000
 SW1 = 000002
 SW10 = 002000
 SW11 = 004000
 SW12 = 010000
 SW13 = 020000
 SW14 = 040000
 SW15 = 100000
 SW2 = 000004
 SW3 = 000010
 SW4 = 000020
 SW5 = 000040
 SW6 = 000100
 SW7 = 000200
 SW8 = 000400
 SW9 = 001000
 SYSTAT 067044
 TADMSK= 177400
 TAG = 020000
 TAGADR= 001114
 TAP = 040000
 TA1 = 000400
 TA2 = 001000
 TA4 = 002000
 TAB = 004000
 TBITVE= 000014
 TIMEOUT 040732
 TKVEC = 000060
 TPVEC = 000064
 TRAPVE= 000034
 TRE = 040000
 TRTVEC= 000014
 TST = 010000
 TSTNMB 063460
 TSTPRP 034032
 TSTQUE 001466
 TST1 007676
 TST10 016440
 TST11 017466

TST12 020712
 TST13 021734
 TST14 022546
 TST15 024416
 TST16 025720
 TST17 026740
 TST2 010076
 TST20 027656
 TST21 030664
 TST22 031712
 TST23 033074
 TST3 011104
 TST4 012062
 TST5 013256
 TST6 014264
 TST7 015242
 TYPBN = 104406
 TYPDS = 104405
 TYPE = 104401
 TYPOC = 104402
 TYPON = 104404
 TYPOS = 104403
 UNS = 040000
 UNTMSK= 000007
 UNTOFF 067155
 UNTON 067166
 UPE = 020000
 USE = 040000
 USRFIL 107272
 U0 = 000001
 U1 = 000002
 U2 = 000004
 VV = 000100
 WC = 000040
 WCD = 000050
 WCE = 040000
 WCEHI = 010000
 WCELO = 004000
 WCF = 000040
 WCH = 000052
 WD = 000060
 WH = 000062
 WLE = 004000
 WRL = 004000
 XSIZ 006364
 XXDP 001332
 Y 067234
 ZEROS 067456
 \$APTHD 001100
 \$ATYC 065564
 \$ATY1 065540
 \$ATY3 065546
 \$ATY4 065556
 \$AUTOB 001150
 \$BASE 001276
 \$BDADR 001136
 \$BDDAT 001142

\$BELL 001212
 \$BIN 060460
 \$CDW1 001302
 \$CDW2 001304
 \$CHARC 061464
 \$CKSWR 064170
 \$CMTAG 001114
 \$CM3 = 000000
 \$CM4 = 000005
 \$CNTLC 065066
 \$CNTLG 065100
 \$CNTLU 065073
 \$CPUOP 001250
 \$CRLF 001217
 \$DBLK 060676
 \$DDW0 001306
 \$DDW1 001310
 \$DDW2 001312
 \$DDW3 001314
 \$DDW4 001316
 \$DDW5 001320
 \$DDW6 001322
 \$DDW7 001324
 \$DEVCT 001232
 \$DEVM 001300
 \$DOAGN 033766
 \$DTBL 060666
 \$ENDAD 033756
 \$ENDCT 033614
 \$ENULL 033772
 \$ENV 001242
 \$ENVM 001243
 \$EOP 033560
 \$EOPCT 033606
 \$EOSP 033522
 \$ERFLG 001117
 \$ERMAX 001131
 \$ERROR 062250
 \$ERRPC 001132
 \$ERRTB 001604
 \$ERTTL 001126
 \$ESCAP 001210
 \$ETABL 001242
 \$ETEND 001326
 \$FATAL 001224
 \$FFLG 066004
 \$FILLC 001172
 \$FILLS 001171
 \$GDADR 001134
 \$GDDAT 001140
 \$GET42 033746
 \$GTSWR 064260
 \$GT42P 033742
 \$HD = 000000
 \$HIBTS 001100
 \$HIOCT 065252
 \$ICNT 001120

\$ILLUP 065522
 \$INTAG 001151
 \$ITEMB 001130
 \$LF 001220
 \$LFLG 066003
 \$LPADR 001122
 \$LPERR 001124
 \$MADR1 001254
 \$MADR2 001260
 \$MADR3 001264
 \$MADR4 001270
 \$MAIL 001222
 \$MAMS1 001252
 \$MAMS2 001256
 \$MAMS3 001262
 \$MAMS4 001266
 \$MBADR 001102
 \$MFLG 066002
 \$MNEW 065116
 \$MSGAD 001236
 \$MSGLG 001240
 \$MSGTY 001222
 \$MSWR 065105
 \$MTYP1 001253
 \$MTYP2 001257
 \$MTYP3 001263
 \$MTYP4 001267
 \$MXCNT 062152
 \$NULL 001170
 \$NWTST= 000001
 \$OCNT 061130
 \$OMODE 061132
 \$OVER 062136
 \$PASS 001230
 \$PASTM 001106
 \$POWER 065530
 \$PURDN 065362
 \$PURMG 065516
 \$PURUP 065434
 \$QUES 001216
 \$RDCHR 064532
 \$RDLIN 064622
 \$RDOCT 065152
 \$RDSZ = 000010
 \$RESRE 060350
 \$RM80 067062
 \$RTNAD 033770
 \$SAVRE 060312
 \$SAVR6 065526
 \$SCOPE 061470
 \$SETUP= 000137
 \$STUP = 177777
 \$SVLAD 062102
 \$SVPC = 000210
 \$SHR = 167400
 \$SWREG 001244
 \$SWRMK= 000000

SYMBOL TABLE

\$SW08T	062154	\$TKS	001160	\$TPS	001164	\$TYPE	061134	\$VECT1	001272
\$TESTN	001226	\$TKSRV	063740	\$STRAP	065254	\$TYPEC	061346	\$VECT2	001274
\$TIMES	001206	\$TMP0	001174	\$STRAP2	065314	\$TYPEX	061466	\$XOFF =	000023
\$TKB	001162	\$TMP1	001176	\$STRP =	000016	\$TYPOC	060732	\$XON =	000021
\$TKCNT	063660	\$TMP2	001200	\$STRPAD	065326	\$STYPON	060746	\$XTSTR	061512
\$TKINT	063670	\$TMP3	001202	\$STSTM	001104	\$TYPOS	060706	\$SGET4=	000000
\$TKQEN=	063667	\$TMP4	001204	\$STSTM	001116	\$UNIT	001234	\$SSW08=	000024
\$TKQIN	063662	\$STN =	000024	\$STTYIN	065056	\$UNITM	001110	\$OFILL	061131
\$TKQOU	063664	\$STPB	001166	\$STYPBN	060406	\$USWR	001246	\$.SX =	001100
\$TKQSR	063666	\$STPFLG	001173	\$STYPDS	060462				

. ABS. 110376 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 62208 WORDS (243 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 70 PAGES
CZRNFA.BIC,CZRNFA/C=CZRNFA.DOC,CZRNFA,[20,0]SYSMAC/M

	13-J82#	13-J85#	13-J89#	13-K02	13-K02#	13-K14#	13-K20#	13-K23#	13-K30#	13-K33#	13-K35#	13-K38#	13-K42#	13-K44#
	13-K46#	13-K58	13-K58#	13-K71#	13-K80#	13-K83#	13-K91#	13-K94#	13-K96#	13-K99#	13-L03#	13-L05#	13-L07#	13-L26#
	13-L40	13-L40#	13-L54#	13-L57#	13-L59#	13-L61#	13-L65#	13-L67#						
ASNDA	7-0#	16-307*	16-344	16-374*	16-377	16-380	16-382*	16-383*	16-384	16-386*	16-425			
ASNDC	7-0#	16-306*	16-343	16-387*	16-388	16-424								
ASWREG	6-0	6-0												
ATA	4-559#	25-134	25-135	25-137	28-161	28-162	28-197	28-200	32-145	32-146	32-185	32-188	50-58	50-59
	50-60	50-63	50-64	50-67	50-68	50-69	50-70	50-71	50-72	50-73	50-74	50-75	50-76	50-79
	50-80	50-83	50-84	50-87	50-88									
ATESTN	6-0	6-0												
ATNMSK	4-596#	33-55												
ATNTBL	10-78	10-106	10-110	11-97	12-17	51-3#								
AUNIT	6-0	6-0												
AUSWR	6-0	6-0												
AVECT1	4-783#	6-0	6-0											
AVECT2	6-0	6-0												
BACK	13-41	13-41#	13-52	13-52#	13-68	13-68#	13-71#	13-73#	13-76	13-76#	13-80	13-80#	13-80#	13-93
	13-93#	13-110	13-110#	13-113#	13-115#	13-118	13-118#	13-122	13-122#	13-122#	13-124	13-124#	13-124#	13-126
	13-126#	13-126#	13-139	13-139#	13-155	13-155#	13-158#	13-160#	13-163	13-163#	13-167	13-167#	13-167#	13-169
	13-169#	13-169#	13-171	13-171#	13-171#	13-175	13-175#	13-188	13-188#	13-199	13-199#	13-215	13-215#	13-218#
	13-220#	13-223	13-223#	13-227	13-227#	13-227#	13-240	13-240#	13-257	13-257#	13-260#	13-262#	13-265	13-265#
	13-269	13-269#	13-269#	13-271	13-271#	13-271#	13-273	13-273#	13-273#	13-286	13-286#	13-301	13-301#	13-304#
	13-306#	13-309	13-309#	13-313	13-313#	13-313#	13-315	13-315#	13-315#	13-317	13-317#	13-317#	13-330	13-330#
	13-341	13-341#	13-356	13-356#	13-359#	13-361#	13-364	13-364#	13-368	13-368#	13-368#	13-381	13-381#	13-398
	13-398#	13-401#	13-403#	13-406	13-406#	13-410	13-410#	13-410#	13-412	13-412#	13-412#	13-414	13-414#	13-414#
	13-428	13-428#	13-443	13-443#	13-446#	13-448#	13-451	13-451#	13-455	13-455#	13-455#	13-460	13-460#	13-460#
	13-488	13-488#	13-488#	13-504	13-504#	13-515	13-515#	13-533	13-533#	13-536#	13-538#	13-541	13-541#	13-545
	13-545#	13-545#	13-559	13-559#	13-576	13-576#	13-579#	13-581#	13-584	13-584#	13-588	13-588#	13-588#	13-590
	13-590#	13-590#	13-592	13-592#	13-592#	13-605	13-605#	13-621	13-621#	13-624#	13-626#	13-629	13-629#	13-633
	13-633#	13-633#	13-635	13-635#	13-635#	13-637	13-637#	13-637#	13-639	13-639#	13-652	13-652#	13-663	13-663#
	13-681	13-681#	13-684#	13-686#	13-689	13-689#	13-693	13-693#	13-693#	13-707	13-707#	13-724	13-724#	13-727#
	13-729#	13-732	13-732#	13-736	13-736#	13-736#	13-738	13-738#	13-738#	13-740	13-740#	13-740#	13-753	13-753#
	13-768	13-768#	13-771#	13-773#	13-776	13-776#	13-780	13-780#	13-780#	13-782	13-782#	13-782#	13-784	13-784#
	13-784#	13-797	13-797#	13-808	13-808#	13-826	13-826#	13-829#	13-831#	13-834	13-834#	13-838	13-838#	13-838#
	13-852	13-852#	13-869	13-869#	13-872#	13-874#	13-877	13-877#	13-881	13-881#	13-881#	13-883	13-883#	13-883#
	13-885	13-885#	13-885#	13-899	13-899#	13-914	13-914#	13-917#	13-919#	13-922	13-922#	13-926	13-926#	13-926#
	13-930	13-930#	13-930#	13-960	13-960#	13-960#	13-985	13-985#	13-988	13-988#	13-:06	13-:06#	13-:09#	13-:11#
	13-:14	13-:14#	13-:18	13-:18#	13-:18#	13-:40	13-:40#	13-:55	13-:55#	13-:58#	13-:60#	13-:63	13-:63#	13-:67
	13-:67#	13-:67#	13-:69	13-:69#	13-:69#	13-:71	13-:71#	13-:71#	13-:84	13-:84#	13-:99	13-:99#	13-:02#	13-:04#
	13-:07	13-:07#	13-:11	13-:11#	13-:11#	13-:13	13-:13#	13-:13#	13-:15	13-:15#	13-:15#	13-:31	13-:31#	13-:43
	13-:43#	13-:61	13-:61#	13-:64#	13-:66#	13-:69	13-:69#	13-:73	13-:73#	13-:73#	13-:86	13-:86#	13-:91	13-:91#
	13-:94#	13-:96#	13-:99	13-:99#	13-<03	13-<03#	13-<03#	13-<21	13-<21#	13-<24#	13-<27	13-<27#	13-<31	13-<31#
	13-<31#	13-<33	13-<33#	13-<33#	13-<35	13-<35#	13-<35#	13-<48	13-<48#	13-<53	13-<53#	13-<56#	13-<58#	13-<61
	13-<61#	13-<65	13-<65#	13-<65#	13-<83	13-<83#	13-<86#	13-<89	13-<89#	13-<93	13-<93#	13-<95	13-<95#	13-<95#
	13-<95#	13-<97	13-<97#	13-<97#	13-<99	13-<99#	13-33	13-33#	13-48	13-48#	13-51#	13-53#	13-56	13-56#
	13-60	13-60#	13-60#	13-78	13-78#	13-81#	13-83#	13-86	13-86#	13-95	13-95#	13-95#	13-97	13-97#
	13-97#	13-99	13-99#	13-99#	13->12	13->12#	13->16	13->16#	13->19#	13->21#	13->24	13->24#	13->28	13->28#
	13->28#	13->43	13->43#	13->46#	13->48#	13->51	13->51#	13->60	13->60#	13->60#	13->62	13->62#	13->62#	13->64
	13->64#	13->64#	13->98	13->98#	13-?16	13-?16#	13-?19#	13-?21#	13-?24	13-?24#	13-?33	13-?33#	13-?33#	13-?35
	13-?35#	13-?35#	13-?37	13-?37#	13-?37#	13-?50	13-?50#	13-?65	13-?65#	13-?68#	13-?70#	13-?73	13-?73#	13-?82
	13-?82#	13-?82#	13-?84	13-?84#	13-?84#	13-?86	13-?86#	13-?86#	13-206	13-206#	13-217	13-217#	13-241	13-241#
	13-244#	13-246#	13-249	13-249#	13-253	13-253#	13-253#	13-266	13-266#	13-283	13-283#	13-286#	13-288#	13-291
	13-291#	13-295	13-295#	13-295#	13-A14	13-A14#	13-A14#	13-A30	13-A30#	13-A30#	13-A43	13-A43#	13-A43#	13-A55
	13-A55#	13-A55#	13-A64	13-A64#	13-A64#	13-A76	13-A76#	13-A93	13-A93#	13-A96#	13-A98#	13-B01	13-B01#	13-B05
	13-B05#	13-B05#	13-B22	13-B22#	13-B22#	13-B38	13-B38#	13-B38#	13-B51	13-B51#	13-B51#	13-B63	13-B63#	13-B63#
	13-B72	13-B72#	13-B72#	13-B97	13-B97#	13-C08	13-C08#	13-C34	13-C34#	13-C37#	13-C39#	13-C42	13-C42#	13-C46

	13-C46#	13-C46#	13-C59	13-C59#	13-C77	13-C77#	13-C80#	13-C82#	13-C85	13-C85#	13-C89	13-C89#	13-C89#	13-D12
	13-D12#	13-D12#	13-D14	13-D14#	13-D14#	13-D27	13-D27#	13-D43	13-D43#	13-D46#	13-D48#	13-D51	13-D51#	13-D55
	13-D55#	13-D55#	13-D78	13-D78#	13-D78#	13-D80	13-D80#	13-D80#	13-D82	13-D82#	13-E11	13-E11#	13-E18	13-E18#
	13-E39	13-E39#	13-E42#	13-E44#	13-E47	13-E47#	13-E51	13-E51#	13-E51#	13-E56	13-E56#	13-E56#	13-E64	13-E64#
	13-E64#	13-E77	13-E77#	13-E99	13-E99#	13-F02#	13-F04#	13-F07	13-F07#	13-F11	13-F11#	13-F11#	13-F16	13-F16#
	13-F16#	13-F24	13-F24#	13-F24#	13-F44	13-F44#	13-F63	13-F63#	13-F66#	13-F69	13-F69#	13-F79#	13-F82	13-F82#
	13-F84	13-F84#	13-F84#	13-F86	13-F86#	13-F86#	13-F99	13-F99#	13-G18	13-G18#	13-G21#	13-G24	13-G24#	13-G35#
	13-G38	13-G38#	13-G40	13-G40#	13-G40#	13-G42	13-G42#	13-G42#	13-G55	13-G55#	13-G66	13-G66#	13-G82	13-G82#
	13-G85#	13-G87#	13-G90	13-G90#	13-G94	13-G94#	13-G94#	13-H07	13-H07#	13-H24	13-H24#	13-H27#	13-H29#	13-H32
	13-H32#	13-H36	13-H36#	13-H36#	13-H38	13-H38#	13-H38#	13-H40	13-H40#	13-H40#	13-H53	13-H53#	13-H69	13-H69#
	13-H72#	13-H74#	13-H77	13-H77#	13-H81	13-H81#	13-H81#	13-H83	13-H83#	13-H83#	13-H85	13-H85#	13-H85#	13-H87
	13-H87#	13-I00	13-I00#	13-I12	13-I12#	13-I30	13-I30#	13-I33#	13-I35#	13-I38	13-I38#	13-I42	13-I42#	13-I42#
	13-I55	13-I55#	13-I72	13-I72#	13-I75#	13-I77#	13-I80	13-I80#	13-I84	13-I84#	13-I84#	13-I86	13-I86#	13-I86#
	13-I88	13-I88#	13-I88#	13-J01	13-J01#	13-J17	13-J17#	13-J20#	13-J22#	13-J25	13-J25#	13-J29	13-J29#	13-J29#
	13-J31	13-J31#	13-J31#	13-J33	13-J33#	13-J33#	13-J35	13-J35#	13-J50	13-J50#	13-J61	13-J61#	13-J77	13-J77#
	13-J80#	13-J82#	13-J85	13-J85#	13-J89	13-J89#	13-J89#	13-K02	13-K02#	13-K14	13-K14#	13-K20	13-K20#	13-K23#
	13-K30	13-K30#	13-K33#	13-K35#	13-K38	13-K38#	13-K42	13-K42#	13-K42#	13-K44	13-K44#	13-K44#	13-K46	13-K46#
	13-K46#	13-K58	13-K58#	13-K71	13-K71#	13-K80	13-K80#	13-K83#	13-K91	13-K91#	13-K94#	13-K96#	13-K99	13-K99#
	13-L03	13-L03#	13-L03#	13-L05	13-L05#	13-L05#	13-L07	13-L07#	13-L07#	13-L26	13-L26#	13-L40	13-L40#	13-L54
	13-L54#	13-L57#	13-L59#	13-L61	13-L61#	13-L65	13-L65#	13-L65#	13-L67	13-L67#	13-L67#			
BADSCT	13-52	13-199	13-341	13-515	13-663	13-808	13-988	13-;43	13-017	13-C08	13-E11	13-G66	13-I12	13-J61
	13-L26	16-33#												
BADTMO	10-3#	10-25												
BAI	4-749#	25-556												
BB00	4-678#													
BB01	4-678#													
BB02	4-678#													
BB03	4-678#													
BB04	4-678#													
BB05	4-678#													
BB06	4-678#													
BB07	4-678#													
BB08	4-678#													
BB09	4-678#													
BIT0	4-492#	23-43												
BIT00	4-492	4-492#	4-504	4-551	4-569	4-588	4-625	4-643	4-678	4-752	4-770	41-1	41-1	42-1
	42-1													
BIT01	4-492	4-492#	4-503	4-550	4-587	4-624	4-642	4-678	4-752	4-769				
BIT02	4-492	4-492#	4-502	4-549	4-586	4-623	4-641	4-678	4-752	4-768				
BIT03	4-492	4-492#	4-501	4-548	4-585	4-622	4-640	4-678	4-690	4-749	4-767			
BIT04	4-492	4-492#	4-500	4-547	4-584	4-639	4-678	4-748						
BIT05	4-492	4-492#	4-499	4-583	4-621	4-638	4-678	4-689	4-747					
BIT06	4-492	4-492#	4-568	4-582	4-604	4-620	4-637	4-678	4-733	4-746	4-766			
BIT07	4-492	4-492#	4-567	4-581	4-603	4-619	4-636	4-661	4-678	4-688	4-732	4-745		
BIT08	4-492	4-492#	4-544	4-566	4-580	4-602	4-618	4-635	4-678	4-731	4-744	41-1		
BIT09	4-492	4-492#	4-543	4-565	4-579	4-601	4-617	4-634	4-660	4-678	4-730	4-743	41-1	41-1
	42-1	42-1												
BIT11	4-492#	16-312	17-26	25-576										
BIT10	4-492#	4-542	4-564	4-578	4-600	4-616	4-633	4-659	4-675	4-687	4-729	4-742	4-765	15-165
	42-1													
BIT11	4-492#	4-498	4-541	4-563	4-577	4-615	4-632	4-650	4-658	4-674	4-686	4-741	4-764	15-167
	41-1													
BIT12	4-492#	4-562	4-576	4-614	4-631	4-657	4-673	4-685	4-740	4-763	15-111			
BIT13	4-492#	4-561	4-575	4-613	4-630	4-649	4-672	4-684	4-696	4-728	4-739	4-762	42-1	
BIT14	4-492#	4-560	4-574	4-612	4-629	4-648	4-671	4-683	4-695	4-727	4-738	4-761	15-73	41-1
BIT15	4-492#	4-559	4-573	4-611	4-628	4-647	4-670	4-682	4-694	4-726	4-737	4-760	18-33	18-90

F
F
G
G
G
G
G

	8-278	8-281	8-284	8-287	8-290	8-293	8-296	8-299	8-302	8-305	8-308	8-311	8-314	8-317
	8-320	8-323	8-326	8-329	8-332	8-335	8-338	8-341	8-344	8-347	8-354	8-357	8-360	8-363
	8-366	8-369	8-372	8-375	8-378	8-381	8-384	8-390	8-393	8-396	8-399	8-402	8-405	8-408
	8-411	8-414	8-417	8-420	8-423	8-432	8-435	8-438	8-441	8-444	8-450	8-453	8-456	8-459
	8-504	8-507	8-510	8-519	8-522	8-525	8-531	8-534	8-537	8-540	8-543	8-546	8-549	8-552
	8-556	8-559	8-563	8-566	8-569	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600	8-603
	8-606	8-609	8-612	8-615	8-618	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642	8-645
	8-648	8-651	8-654	8-657	8-660	8-663	8-666	8-669	8-672	8-675	8-678	8-693	8-696	8-702
	8-705	8-708	8-711	8-714	8-723	54-1#								
EHT110	8-221	54-4#												
EHT111	8-224	8-227	54-5#											
EHT114	54-7#													
EHT2	8-426	8-429	8-513	8-516	54-2#									
EHT223	8-447	54-8#												
EHT256	8-528	54-9#												
EHT336	8-351	8-681	8-687	8-690	54-11#									
EHT337	8-684	54-12#												
EHT344	8-699	54-13#												
EHT353	8-720	54-15#												
EMS1	53-3	57-3#												
EMS10	53-7	53-11	57-10#											
EMS100	53-39	53-71	53-176	53-196	57-66#									
EMS101	53-40	53-59	53-174	53-197	57-67#									
EMS102	53-177	53-178	53-181	53-182	53-183	53-184	53-186	53-187	53-188	53-189	53-234	53-235	57-68#	
EMS103	53-56	53-57	53-135	53-141	53-196	53-197	53-198	53-203	53-204	53-205	53-206	53-207	53-208	53-209
	53-210	53-211	53-213	53-214	53-215	53-219	53-220	57-69#						
EMS104	53-136	53-180	53-203	57-70#										
EMS105	53-204	57-71#												
EMS106	53-212	53-232	57-72#											
EMS11	53-12	53-15	53-16	53-17	53-18	53-19	53-20	53-21	53-22	53-23	53-24	53-25	53-26	53-27
	53-28	53-29	53-30	53-31	53-32	53-33	53-43	53-152	53-177	57-11#				
EMS110	53-9	53-74	53-77	57-73#										
EMS111	53-74	53-226	57-74#											
EMS112	53-74	57-75#												
EMS113	53-75	53-76	53-127	57-77#										
EMS114	53-77	57-78#												
EMS115	53-44	53-45	53-48	53-50	53-51	53-52	53-54	53-55	53-60	53-61	53-63	53-66	53-67	53-70
	53-73	53-77	53-107	53-108	53-109	53-110	53-134	53-137	53-139	53-140	53-142	53-150	53-151	53-153
	53-179	53-182	53-183	53-185	53-186	53-187	53-188	53-189	53-190	53-192	53-193	53-194	53-195	53-201
	53-205	53-221	53-222	53-223	53-232	53-234	57-79#							
EMS116	53-74	53-77	53-191	53-225	57-80#									
EMS117	53-77	53-143	53-148	53-149	53-150	53-178	53-185	53-201	53-225	53-228	53-229	53-230	53-231	57-81#
EMS12	53-12	57-12#												
EMS120	57-82#													
EMS121	53-79	53-88	53-97	53-99	57-83#									
EMS122	53-80	53-89	57-84#											
EMS123	53-81	53-88	53-89	53-90	53-90	53-91	53-92	53-93	53-94	53-95	53-96	53-98	53-126	57-85#
EMS124	53-56	53-82	53-91	53-101	57-86#									
EMS125	53-83	53-92	53-102	53-169	57-87#									
EMS126	53-84	53-93	53-103	57-88#										
EMS127	57-89#													
EMS13	53-13	53-42	53-216	57-13#										
EMS130	53-86	53-95	53-106	57-90#										
EMS131	53-87	53-96	53-104	57-91#										
EMS132	53-79	53-80	53-81	53-82	53-83	53-84	53-85	53-86	53-87	53-88	53-89	53-90	53-91	53-92
	53-93	53-94	53-95	53-96	53-97	53-98	53-99	53-100	53-101	53-102	53-103	53-104	53-105	53-106

EMT213	8-423	53-141#
EMT214	8-426	53-142#
EMT215	8-429	53-143#
EMT216	8-432	53-144#
EMT217	8-435	53-145#
EMT22	8-54	53-20#
EMT220	8-438	53-146#
EMT221	8-441	53-147#
EMT222	8-444	53-148#
EMT223	8-447	53-149#
EMT224	8-450	53-150#
EMT225	8-453	53-151#
EMT226	8-456	53-152#
EMT227	8-459	53-153#
EMT23	8-57	53-21#
EMT230	8-462	53-154#
EMT231	8-465	53-155#
EMT232	8-468	53-156#
EMT233	8-471	53-157#
EMT234	53-158#	
EMT235	53-159#	
EMT236	53-160#	
EMT237	53-161#	
EMT24	8-60	53-22#
EMT240	53-162#	
EMT241	53-163#	
EMT242	53-164#	
EMT243	53-165#	
EMT244	53-166#	
EMT245	53-167#	
EMT246	8-504	53-168#
EMT247	8-507	53-169#
EMT25	8-63	53-23#
EMT250	8-510	53-170#
EMT251	8-513	53-171#
EMT252	8-516	53-172#
EMT253	8-519	53-173#
EMT254	8-522	53-174#
EMT255	8-525	53-175#
EMT256	8-528	53-176#
EMT257	8-531	53-177#
EMT26	8-66	53-24#
EMT260	8-534	53-178#
EMT261	8-537	53-179#
EMT262	8-540	53-180#
EMT263	8-543	53-181#
EMT264	8-546	53-182#
EMT265	8-549	53-183#
EMT266	8-552	53-184#
EMT267	8-556	53-185#
EMT27	8-69	53-25#
EMT270	8-559	53-186#
EMT271	8-563	53-187#
EMT272	8-566	53-188#
EMT273	8-569	53-189#
EMT274	8-573	53-190#

TY
TY
TY
UO
U1
U2
UN
UN
UN
UN
UP
US
US
VV
WC
WC
WC
WC
WC
WC
WD
WH
WL
WR
XS
XX
Y
ZE

EMT37	8-93	53-33#																		
EMT4	8-12	53-6#																		
EMT40	8-96	53-34#																		
EMT41	8-99	53-35#																		
EMT42	8-102	53-36#																		
EMT43	8-105	53-37#																		
EMT44	8-108	53-38#																		
EMT45	8-111	53-39#																		
EMT46	8-114	53-40#																		
EMT47	8-117	53-41#																		
EMT5	8-15	53-7#																		
EMT50	8-120	53-42#																		
EMT51	8-123	53-43#																		
EMT52	8-126	53-44#																		
EMT53	8-130	53-45#																		
EMT54	8-133	53-46#																		
EMT55	8-136	53-47#																		
EMT56	8-139	53-48#																		
EMT57	8-142	53-49#																		
EMT6	8-18	53-8#																		
EMT60	8-146	53-50#																		
EMT61	8-150	53-51#																		
EMT62	8-154	53-52#																		
EMT63	8-157	53-53#																		
EMT64	8-160	53-54#																		
EMT65	8-163	53-55#																		
EMT66	8-166	53-56#																		
EMT67	8-169	53-57#																		
EMT7	8-21	53-9#																		
EMT70	8-172	53-58#																		
EMT71	8-175	53-59#																		
EMT72	8-178	53-60#																		
EMT73	8-182	53-61#																		
EMT74	8-185	53-62#																		
EMT75	8-188	53-63#																		
EMT76	8-191	53-64#																		
EMT77	8-194	53-65#																		
EMTVEC	4-492#	10-23*	10-23*																	
ENRGDT	52-144#																			
EQUALS	49-4#																			
ERR	4-560#	25-80	25-82	25-98	25-132															
ERRNMB	43-40*	43-41*	43-44	43-49	43-56	43-152#														
ERROR	4-492#																			
ERRTYP	42-1	43-13#																		
ERRVEC	4-492#	10-23	10-23*	10-23*	10-25*	10-26*	13-5	13-5	13-6*	13-7*	13-20*	13-20*	13-24*	13-24*						
	20-35	20-35	20-39*	20-40*	20-70*	20-70*	21-29	21-29	21-33*	21-34*	21-63*	21-63*	22-3	22-3						
	22-4*	22-5*	22-11*	22-18*	22-18*	23-9	23-9	23-10*	23-11*	23-31*	23-31*	27-20	27-20	27-21*						
	27-22*	27-51*	27-51*	29-12	29-12	29-13*	29-14*	29-25*	29-25*	41-1	41-1	41-1*	41-1*	41-1*						
	41-1*	41-1*	42-1	42-1*	42-1*															
ERTY00	43-21	43-157#																		
ERTY01	43-38	43-158#																		
ERTY02	43-43	43-159#																		
ERTY03	43-45	43-160#																		
ERTY04	43-144	43-161#																		
ERTY05	43-31	43-162#																		
ESRC	4-632#																			

CO
EM
ER

ES
GE
GE
MS
ML
NE
NW
PC
PL
PL
RE
RC
SE
SE
SI

FO	4-503#	25-28												
F1	4-502#	25-28												
F2	4-501#	25-28	25-675											
F3	4-500#	25-28												
F4	4-499#	25-28												
FER	4-584#	4-590	13-A40	13-A45	13-848	13-853	13-C92	13-C95	13-D58	13-D61	16-221	25-412	34-352	34-369
FIND	34-372	34-395	34-398											
	13-41	13-41#	13-41#	13-52	13-52#	13-52#	13-68	13-68#	13-68#	13-71	13-71#	13-73	13-73#	13-76
	13-76#	13-76#	13-80	13-80#	13-80#	13-93	13-93#	13-93#	13-110	13-110#	13-110#	13-113	13-113#	13-115
	13-115#	13-118	13-118#	13-118#	13-122	13-122#	13-122#	13-124	13-124#	13-124#	13-126	13-126#	13-126#	13-139
	13-139#	13-139#	13-155	13-155#	13-155#	13-158	13-158#	13-160	13-160#	13-163	13-163#	13-163#	13-167	13-167#
	13-167#	13-169	13-169#	13-169#	13-171	13-171#	13-171#	13-175	13-175#	13-175#	13-188	13-188#	13-188#	13-199
	13-199#	13-199#	13-215	13-215#	13-215#	13-218	13-218#	13-220	13-220#	13-223	13-223#	13-223#	13-227	13-227#
	13-227#	13-240	13-240#	13-240#	13-257	13-257#	13-257#	13-260	13-260#	13-262	13-262#	13-265	13-265#	13-265#
	13-269	13-269#	13-269#	13-271	13-271#	13-271#	13-273	13-273#	13-273#	13-286	13-286#	13-286#	13-301	13-301#
	13-301#	13-304	13-304#	13-306	13-306#	13-309	13-309#	13-309#	13-313	13-313#	13-313#	13-315	13-315#	13-315#
	13-317	13-317#	13-317#	13-330	13-330#	13-330#	13-341	13-341#	13-341#	13-356	13-356#	13-356#	13-359	13-359#
	13-361	13-361#	13-364	13-364#	13-364#	13-368	13-368#	13-368#	13-381	13-381#	13-381#	13-398	13-398#	13-398#
	13-401	13-401#	13-403	13-403#	13-406	13-406#	13-406#	13-410	13-410#	13-410#	13-412	13-412#	13-412#	13-414
	13-414#	13-414#	13-428	13-428#	13-428#	13-443	13-443#	13-443#	13-446	13-446#	13-448	13-448#	13-451	13-451#
	13-451#	13-455	13-455#	13-455#	13-460	13-460#	13-460#	13-488	13-488#	13-488#	13-504	13-504#	13-504#	13-515
	13-515#	13-515#	13-533	13-533#	13-533#	13-536	13-536#	13-538	13-538#	13-541	13-541#	13-541#	13-545	13-545#
	13-545#	13-559	13-559#	13-559#	13-576	13-576#	13-576#	13-579	13-579#	13-581	13-581#	13-584	13-584#	13-584#
	13-588	13-588#	13-588#	13-590	13-590#	13-590#	13-592	13-592#	13-592#	13-605	13-605#	13-605#	13-621	13-621#
	13-621#	13-624	13-624#	13-626	13-626#	13-629	13-629#	13-629#	13-633	13-633#	13-633#	13-635	13-635#	13-635#
	13-637	13-637#	13-637#	13-639	13-639#	13-639#	13-652	13-652#	13-652#	13-663	13-663#	13-663#	13-681	13-681#
	13-681#	13-684	13-684#	13-686	13-686#	13-689	13-689#	13-689#	13-693	13-693#	13-693#	13-707	13-707#	13-707#
	13-724	13-724#	13-724#	13-727	13-727#	13-729	13-729#	13-732	13-732#	13-732#	13-736	13-736#	13-736#	13-738
	13-738#	13-738#	13-740	13-740#	13-740#	13-753	13-753#	13-753#	13-768	13-768#	13-768#	13-771	13-771#	13-773
	13-773#	13-776	13-776#	13-776#	13-780	13-780#	13-780#	13-782	13-782#	13-782#	13-784	13-784#	13-784#	13-797
	13-797#	13-797#	13-808	13-808#	13-808#	13-826	13-826#	13-826#	13-829	13-829#	13-831	13-831#	13-834	13-834#
	13-834#	13-838	13-838#	13-838#	13-852	13-852#	13-852#	13-869	13-869#	13-869#	13-872	13-872#	13-874	13-874#
	13-877	13-877#	13-877#	13-881	13-881#	13-881#	13-883	13-883#	13-883#	13-885	13-885#	13-885#	13-899	13-899#
	13-899#	13-914	13-914#	13-914#	13-917	13-917#	13-919	13-919#	13-922	13-922#	13-922#	13-926	13-926#	13-926#
	13-930	13-930#	13-930#	13-960	13-960#	13-960#	13-985	13-985#	13-985#	13-988	13-988#	13-988#	13-988#	13-988#
	13-:06#	13-:09	13-:09#	13-:11	13-:11#	13-:14	13-:14#	13-:14#	13-:18	13-:18#	13-:18#	13-:40	13-:40#	13-:40#
	13-:55	13-:55#	13-:55#	13-:58	13-:58#	13-:60	13-:60#	13-:60#	13-:63	13-:63#	13-:63#	13-:67	13-:67#	13-:67#
	13-:69#	13-:69#	13-:71	13-:71#	13-:71#	13-:84	13-:84#	13-:84#	13-:99	13-:99#	13-:99#	13-:02	13-:02#	13-:02#
	13-:04#	13-:07	13-:07#	13-:07#	13-:11	13-:11#	13-:11#	13-:13	13-:13#	13-:13#	13-:15	13-:15#	13-:15#	13-:31
	13-:31#	13-:31#	13-:43	13-:43#	13-:43#	13-:61	13-:61#	13-:61#	13-:64	13-:64#	13-:64#	13-:66	13-:66#	13-:69
	13-:69#	13-:73	13-:73#	13-:73#	13-:86	13-:86#	13-:86#	13-:91	13-:91#	13-:91#	13-:94	13-:94#	13-:96	13-:96#
	13-:99	13-:99#	13-:99#	13-<03	13-<03#	13-<03#	13-<21	13-<21#	13-<21#	13-<21#	13-<24	13-<24#	13-<27	13-<27#
	13-<31	13-<31#	13-<31#	13-<33	13-<33#	13-<33#	13-<35	13-<35#	13-<35#	13-<48	13-<48#	13-<48#	13-<53	13-<53#
	13-<53#	13-<56	13-<56#	13-<58	13-<58#	13-<61	13-<61#	13-<61#	13-<65	13-<65#	13-<65#	13-<83	13-<83#	13-<83#
	13-<86	13-<86#	13-<89	13-<89#	13-<89#	13-<93	13-<93#	13-<93#	13-<95	13-<95#	13-<95#	13-<97	13-<97#	13-<97#
	13-<99	13-<99#	13-<99#	13-=33	13-=33#	13-=33#	13-=48	13-=48#	13-=48#	13-=51	13-=51#	13-=53	13-=53#	13-=56
	13-=56#	13-=56#	13-=60	13-=60#	13-=60#	13-=78	13-=78#	13-=78#	13-=81	13-=81#	13-=83	13-=83#	13-=86	13-=86#
	13-=86#	13-=95	13-=95#	13-=95#	13-=97	13-=97#	13-=97#	13-=99	13-=99#	13-=99#	13->12	13->12#	13->12#	13->16
	13->16#	13->16#	13->19	13->19#	13->21	13->21#	13->24	13->24#	13->24#	13->24#	13->28	13->28#	13->28#	13->43
	13->43#	13->46	13->46#	13->48	13->48#	13->51	13->51#	13->51#	13->60	13->60#	13->60#	13->62	13->62#	13->62#
	13->64	13->64#	13->64#	13->98	13->98#	13->98#	13-?16	13-?16#	13-?16#	13-?19	13-?19#	13-?21	13-?21#	13-?24
	13-?24#	13-?24#	13-?33	13-?33#	13-?33#	13-?35	13-?35#	13-?35#	13-?37	13-?37#	13-?37#	13-?50	13-?50#	13-?50#
	13-?65	13-?65#	13-?65#	13-?68	13-?68#	13-?70	13-?70#	13-?73	13-?73#	13-?73#	13-?82	13-?82#	13-?82#	13-?84
	13-?84#	13-?84#	13-?86	13-?86#	13-?86#	13-?06	13-?06#	13-?06#	13-?17	13-?17#	13-?17#	13-?41	13-?41#	13-?41#
	13-?44	13-?44#	13-?46	13-?46#	13-?49	13-?49#	13-?49#	13-?53	13-?53#	13-?53#	13-?66	13-?66#	13-?66#	13-?83
	13-?83#	13-?83#	13-?86	13-?86#	13-?88	13-?88#	13-?91	13-?91#	13-?91#	13-?95	13-?95#	13-?95#	13-A14	13-A14#

	13-A14#	13-A30	13-A30#	13-A30#	13-A43	13-A43#	13-A43#	13-A55	13-A55#	13-A55#	13-A64	13-A64#	13-A64#	13-A76
	13-A76#	13-A76#	13-A93	13-A93#	13-A93#	13-A96	13-A96#	13-A98	13-A98#	13-B01	13-B01#	13-B01#	13-B05	13-B05#
	13-B05#	13-B22	13-B22#	13-B22#	13-B38	13-B38#	13-B38#	13-B51	13-B51#	13-B51#	13-B63	13-B63#	13-B63#	13-B72
	13-B72#	13-B72#	13-B97	13-B97#	13-B97#	13-C08	13-C08#	13-C08#	13-C34	13-C34#	13-C34#	13-C37	13-C37#	13-C39
	13-C39#	13-C42	13-C42#	13-C42#	13-C46	13-C46#	13-C46#	13-C59	13-C59#	13-C59#	13-C77	13-C77#	13-C77#	13-C80
	13-C80#	13-C82	13-C82#	13-C85	13-C85#	13-C85#	13-C89	13-C89#	13-C89#	13-D12	13-D12#	13-D12#	13-D14	13-D14#
	13-D14#	13-D27	13-D27#	13-D27#	13-D43	13-D43#	13-D43#	13-D46	13-D46#	13-D48	13-D48#	13-D51	13-D51#	13-D51#
	13-D55	13-D55#	13-D55#	13-D78	13-D78#	13-D78#	13-D80	13-D80#	13-D80#	13-D82	13-D82#	13-D82#	13-E11	13-E11#
	13-E11#	13-E18	13-E18#	13-E18#	13-E39	13-E39#	13-E39#	13-E42	13-E42#	13-E44	13-E44#	13-E47	13-E47#	13-E47#
	13-E51	13-E51#	13-E51#	13-E56	13-E56#	13-E56#	13-E64	13-E64#	13-E64#	13-E77	13-E77#	13-E77#	13-E99	13-E99#
	13-E99#	13-F02	13-F02#	13-F04	13-F04#	13-F04#	13-F07	13-F07#	13-F07#	13-F11	13-F11#	13-F11#	13-F16	13-F16#
	13-F24	13-F24#	13-F24#	13-F44	13-F44#	13-F44#	13-F63	13-F63#	13-F63#	13-F66	13-F66#	13-F69	13-F69#	13-F69#
	13-F79	13-F79#	13-F82	13-F82#	13-F82#	13-F84	13-F84#	13-F84#	13-F86	13-F86#	13-F86#	13-F99	13-F99#	13-F99#
	13-G18	13-G18#	13-G18#	13-G21	13-G21#	13-G24	13-G24#	13-G24#	13-G35	13-G35#	13-G38	13-G38#	13-G38#	13-G40
	13-G40#	13-G40#	13-G42	13-G42#	13-G42#	13-G55	13-G55#	13-G55#	13-G66	13-G66#	13-G66#	13-G82	13-G82#	13-G82#
	13-G85	13-G85#	13-G87	13-G87#	13-G90	13-G90#	13-G90#	13-G94	13-G94#	13-G94#	13-H07	13-H07#	13-H07#	13-H24
	13-H24#	13-H24#	13-H27	13-H27#	13-H29	13-H29#	13-H32	13-H32#	13-H32#	13-H36	13-H36#	13-H36#	13-H38	13-H38#
	13-H38#	13-H40	13-H40#	13-H40#	13-H53	13-H53#	13-H53#	13-H69	13-H69#	13-H69#	13-H72	13-H72#	13-H74	13-H74#
	13-H77	13-H77#	13-H77#	13-H81	13-H81#	13-H81#	13-H83	13-H83#	13-H83#	13-H85	13-H85#	13-H85#	13-H87	13-H87#
	13-H87#	13-I00	13-I00#	13-I00#	13-I12	13-I12#	13-I12#	13-I30	13-I30#	13-I30#	13-I33	13-I33#	13-I35	13-I35#
	13-I38	13-I38#	13-I38#	13-I42	13-I42#	13-I42#	13-I55	13-I55#	13-I55#	13-I72	13-I72#	13-I72#	13-I75	13-I75#
	13-I77	13-I77#	13-I80	13-I80#	13-I80#	13-I84	13-I84#	13-I84#	13-I86	13-I86#	13-I86#	13-I88	13-I88#	13-I88#
	13-J01	13-J01#	13-J01#	13-J17	13-J17#	13-J17#	13-J20	13-J20#	13-J22	13-J22#	13-J25	13-J25#	13-J25#	13-J29
	13-J29#	13-J29#	13-J31	13-J31#	13-J31#	13-J33	13-J33#	13-J33#	13-J35	13-J35#	13-J35#	13-J50	13-J50#	13-J50#
	13-J61	13-J61#	13-J61#	13-J77	13-J77#	13-J77#	13-J80	13-J80#	13-J82	13-J82#	13-J85	13-J85#	13-J85#	13-J89
	13-J89#	13-J89#	13-K02	13-K02#	13-K02#	13-K14	13-K14#	13-K14#	13-K20	13-K20#	13-K20#	13-K23	13-K23#	13-K30
	13-K30#	13-K30#	13-K33	13-K33#	13-K35	13-K35#	13-K38	13-K38#	13-K38#	13-K42	13-K42#	13-K42#	13-K44	13-K44#
	13-K44#	13-K46	13-K46#	13-K46#	13-K58	13-K58#	13-K58#	13-K71	13-K71#	13-K71#	13-K80	13-K80#	13-K80#	13-K83
	13-K83#	13-K91	13-K91#	13-K91#	13-K94	13-K94#	13-K96	13-K96#	13-K99	13-K99#	13-K99#	13-L03	13-L03#	13-L03#
	13-L05	13-L05#	13-L05#	13-L07	13-L07#	13-L07#	13-L26	13-L26#	13-L26#	13-L40	13-L40#	13-L40#	13-L54	13-L54#
	13-L54#	13-L57	13-L57#	13-L59	13-L59#	13-L61	13-L61#	13-L61#	13-L65	13-L65#	13-L65#	13-L67	13-L67#	13-L67#
FMT16	4-657#	13-49	13-196	13-338	13-512	13-660	13-805	13-974	13-40	13-17	13->82	13-a14	13-A04	13-B12
	13-C05	13-C64	13-E08	13-F39	13-G63	13-I09	13-J58	13-L21	16-59	16-274	16-283	17-32	17-34	18-27
	25-447	28-59	34-183											
FNCDTB	25-126	25-442	50-55#											
FNCMSK	4-505#	33-17	34-345											
GENBUF	13-56	13-203	13-345	13-519	13-667	13-812	13-992	13-:30	13-:47	13-a21	13-C12	13-E15	13-G70	13-I16
	13-J65	13-L32	17-20#											
GET	13-76	13-118	13-163	13-223	13-265	13-309	13-364	13-406	13-451	13-470	13-541	13-584	13-629	13-689
	13-732	13-776	13-834	13-877	13-922	13-941	13-:14	13-:63	13-:07	13-:69	13-:99	13-<27	13-<61	13-<89
	13-=56	13-=86	13->24	13->51	13-?24	13-?73	13-a49	13-a91	13-B01	13-C42	13-C85	13-D51	13-E47	13-F07
	13-F69	13-F82	13-G24	13-G38	13-G90	13-H32	13-H77	13-I38	13-I80	13-J25	13-J85	13-K38	13-K99	13-L61
	15-47	15-90	15-114	15-143	15-170	15-201	16-75	16-96	16-128	16-162	16-191	20-31#		
GETBUF	7-0#	20-37												
GETINX	7-0#	13-470*	13-470*	13-941*	13-941*	19-13	20-38							
GETSSF	16-270	16-444#												
GETSTS	13-71	13-113	13-158	13-218	13-260	13-304	13-359	13-401	13-446	13-536	13-579	13-624	13-684	13-727
	13-771	13-829	13-872	13-917	13-:09	13-:58	13-:02	13-:64	13-:94	13-<56	13-=51	13-=81	13->19	13->46
	13-?19	13-?68	13-a44	13-a86	13-A96	13-C37	13-C80	13-D46	13-E42	13-F02	13-F66	13-G21	13-G85	13-H27
	13-H72	13-I33	13-I75	13-J20	13-J80	13-K33	13-K94	13-L57	15-46	16-74	19-10#			
GNS	5-1	5-1	10-6	10-28	10-42	10-53	10-59	12-30	14-20	14-20	46-1	46-1	46-1	46-1
	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1
	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1
GO	4-504#	13-50	13-100	13-145	13-197	13-247	13-291	13-339	13-388	13-433	13-513	13-566	13-611	13-661
	13-714	13-758	13-806	13-859	13-904	13-982	13-:45	13-:89	13-:41	13-:89	13-<11	13-<51	13-<72	13-=37
	13-=65	13->14	13->33	13-?03	13-?55	13-a15	13-a73	13-A83	13-C06	13-C66	13-D32	13-E27	13-E86	13-F40

	13-F53	13-G08	13-G64	13-H14	13-H58	13-I10	13-I62	13-J07	13-J59	13-K15	13-K25	13-K75	13-K86	13-L44
	15-125	15-183	16-86	16-118	16-152	16-181	23-17	24-90	24-102	24-107	24-123	25-47	25-50	34-346
GTSWR	10-28	46-1#												
HCE	4-581#	4-590	13-A11	13-A17	13-B19	13-B25	13-C92	13-C95	13-D58	13-D61	16-221	25-375	25-380	25-395
	25-397	25-410	25-425	25-451	25-456	34-352	34-382	34-385	34-395	34-398	50-77	50-78	50-81	50-85
	50-86													
HCI	4-659#	13-C64	25-373	34-348										
HCRC	4-580#	4-590	16-221	25-382	34-352	34-356	34-359	34-395	34-398					
HELP	11-17	61-23#												
HT	4-492#	40-1	40-1	43-82										
IAE	4-578#	25-279	25-651	25-667	25-694	28-48	28-72	32-24	32-78	32-81	34-172	34-194	50-59	50-69
	50-77	50-78	50-81	50-82	50-85	50-86								
IBSAVE	42-1	42-1	42-1	42-1	42-1	42-1#	42-1*	42-1*	42-1*					
IDXMSK	4-720#	20-51	20-54	21-45	21-48									
IE	4-733#	4-766#												
ILF	4-588#	25-151	25-153	31-51	31-111	31-115	32-24	32-45	32-48	34-128	34-145	34-148	50-58	50-67
	50-68	50-70	50-71	50-72	50-73	50-74	50-75	50-76	50-79	50-80	50-83	50-84	50-87	50-88
ILF02	4-509#													
ILF24	4-519#													
ILF26	4-520#													
ILF30	4-524#													
ILF32	4-524#													
ILF34	4-524#													
ILF36	4-524#													
ILF40	4-524#													
ILF42	4-524#													
ILF44	4-524#													
ILF46	4-524#													
ILF54	4-527#													
ILF56	4-528#													
ILF64	4-531#													
ILF66	4-532#													
ILF74	4-535#													
ILF76	4-536#													
ILR	4-587#	31-51	31-97	31-101	32-232	32-236	32-239	34-128	34-131	34-134				
ILRG50	4-718#													
ILRG52	4-718#													
ILRG54	4-718#													
ILRG56	4-718#													
ILRG60	4-718#													
ILRG62	4-718#													
ILRG64	4-718#													
ILRG66	4-718#													
ILRG70	4-718#													
ILRG72	4-718#													
ILRG74	4-718#													
ILRG76	4-718#													
IOTVEC	4-492#	10-23*	10-23*											
IPCK0	4-770#													
IPCK1	4-769#													
IPCK2	4-768#													
IPCK3	4-767#													
IR	4-746#	30-56												
IVC	4-685#	13-E53	13-E60	13-F13	13-F19	25-194	25-195	28-90	28-213	32-92	32-99	32-102	32-171	34-111
	34-115	34-550	35-58	50-58	50-59	50-60	50-62	50-63	50-64	50-67	50-68	50-69	50-70	50-71
	50-72	50-73	50-74	50-75	50-76	50-77	50-78	50-79	50-80	50-81	50-82	50-83	50-84	50-85

REX	4-631#													
RG	4-629#													
RGDTPT	52-3#													
RH	4-534#	16-181												
RIP	4-516#													
RELEASE	4-513#													
RMAS	4-704#													
RMASI	7-0#	33-50	59-13											
RMASO	7-0#													
RMBA	4-776#	13-12*	13-13	13-62	13-105	13-150	13-209	13-252	13-296	13-350	13-393	13-438	13-525	13-571
	13-616	13-673	13-719	13-763	13-818	13-864	13-909	13-998	13-:50	13-:94	13-:53	13-<16	13-<78	13-:41
	13-:73	13->38	13-?11	13-?60	13-?35	13-?78	13-A88	13-C26	13-C72	13-D38	13-E35	13-E94	13-F58	13-G13
	13-G76	13-H19	13-H64	13-I22	13-I67	13-J12	13-J71	13-K10	13-K67	13-L50	16-67			
RMBAE	4-779#													
RMBAEI	7-0#													
RMBAEO	7-0#													
RMBAI	7-0#	13-476	13-948	25-560	25-562	30-42	59-14							
RMBAO	7-0#	13-47*	13-98*	13-144*	13-194*	13-245*	13-336*	13-387*	13-510*	13-564*	13-610*	13-658*	13-712*	13-803*
	13-858*	13-980*	13-:26*	13-:38*	13-<10*	13-<73*	13-:20*	13->86*	13-?12*	13-?71*	13-A81*	13-C03*	13-C67*	13-D33*
	13-E06*	13-E87*	13-F37*	13-G61*	13-H13*	13-H59*	13-I07*	13-I60*	13-J06*	13-J56*	13-J97*	13-L23*	16-60*	16-246
	16-275*	16-279	16-282*	17-21	18-31	25-554	25-558							
RMCS1	4-700#	4-774#	10-89	13-9*	13-65	13-107	13-152	13-212	13-254	13-298	13-353	13-395	13-440	13-528
	13-573	13-618	13-676	13-721	13-765	13-821	13-866	13-911	13-:01	13-:52	13-:96	13-:56	13-<18	13-<80
	13-:44	13-:75	13->40	13-?13	13-?62	13-?38	13-?80	13-A90	13-C29	13-C74	13-D40	13-E36	13-E96	13-F60
	13-G15	13-G79	13-H21	13-H66	13-I25	13-I69	13-J14	13-J74	13-K17	13-K27	13-K77	13-K88	13-L51	15-126
	15-184	16-68	20-42	21-36	23-16	27-28								
RMCS1I	7-0#	16-208	24-65	24-67	24-69	24-86	24-88	24-91	24-102	24-106	24-108	24-120	24-122	24-124
	25-47	25-49	25-60	25-103	25-540	30-29	33-16	34-27	34-29	34-31	59-13			
RMCS1O	7-0#	13-50*	13-100*	13-145*	13-197*	13-247*	13-291*	13-339*	13-388*	13-433*	13-513*	13-566*	13-611*	13-661*
	13-714*	13-758*	13-806*	13-859*	13-904*	13-982*	13-:27*	13-:45*	13-:89*	13-:41*	13-:89*	13-<11*	13-<51*	13-<72*
	13-:22*	13-:37*	13-:65*	13->14*	13->33*	13->87*	13-?03*	13-?55*	13-?15*	13-?73*	13-A83*	13-C06*	13-C66*	13-D32*
	13-E09*	13-E27*	13-E86*	13-F40*	13-F53*	13-G08*	13-G64*	13-H14*	13-H58*	13-I10*	13-I62*	13-J07*	13-J59*	13-K15*
	13-K25*	13-K75*	13-K86*	13-L24*	13-L44*	15-125*	15-183*	16-86*	16-118*	16-152*	16-181*	16-312	17-26	25-27
	25-576	34-324	34-344											
RMCS2	4-777#	10-84*	10-85*	10-87	12-55*	12-56*	13-14	13-15*	13-16	13-17*	20-41	21-35	27-27*	27-29
	29-16*	29-18*												
RMCS2I	7-0#	13-457	13-462	13-464	13-928	13-932	13-934	24-47	24-72	24-74	24-75	25-95	25-166	25-267
	25-310	25-325	25-340	25-482	25-556	30-51	34-50	34-63	34-65	34-67	34-442	34-444	34-446	34-496
	34-498	34-500	59-13											
RMCS2O	7-0#													
RMCS3	4-780#													
RMCS3I	7-0#													
RMCS3O	7-0#													
RMDA	4-701#	13-61	13-102	13-147	13-208	13-249	13-293	13-349	13-390	13-435	13-524	13-568	13-613	13-672
	13-716	13-760	13-817	13-861	13-906	13-997	13-:47	13-:91	13-:52	13-<13	13-<75	13-:40	13-:70	13->35
	13-?08	13-?57	13-?34	13-?75	13-A85	13-C25	13-C69	13-D35	13-E31	13-E91	13-F55	13-G10	13-G75	13-H16
	13-H61	13-I21	13-I64	13-J09	13-J70	13-K07	13-K64	13-L46	16-63					
RMDAI	7-0#	25-628	25-699	59-14	59-16									
RMDAO	7-0#	13-46*	13-193*	13-335*	13-509*	13-657*	13-802*	13-979*	13-:37*	13-:16*	13->68*	13->80*	13->81*	13-?90*
	13-?91*	13-?11*	13-C02*	13-E05*	13-E13	13-F36*	13-G60*	13-I05*	13-J37*	13-J38	13-J55*	13-L19*	13-L71*	13-L72
	13-L75*	13-L76*	13-L77	13-L80*	16-57*	16-233*	16-238*	16-241*	16-242	16-248	16-250	16-273*	16-281*	16-307
	16-425*	17-24	25-588	25-589	25-628	28-54	28-57	28-61	28-65	34-178	34-181	34-185	34-189	
RMDB	4-778#	13-18*	13-19	13-470	13-941	19-20								
RMDBI	7-0#	13-472	13-941*	13-944										
RMDBO	7-0#													
RMDC	4-710#	13-60	13-103	13-149	13-207	13-250	13-295	13-348	13-391	13-437	13-523	13-569	13-615	13-671

SNGPRT	4-652#													
SSE	4-689#	13-=91	13->56	13-?29	13-?78	13-A27	13-A34	13-B35	13-B42	13-D06	13-D08	13-D72	13-D74	16-224
	16-231	25-458												
SSEI	4-660#	13-;17	13-;19	13->66	13->69	13-?88	13-?92	13-C64	13-L21	16-235	16-375	16-401	17-37	25-449
	25-599	25-624	25-626	25-637	28-63	34-187								
SSF	4-696#	13-A02	13-B10	13-L36										
SSFENB	7-0#	13-L34	16-338*	16-365*										
SSFIL	16-336	16-446	16-451	16-454	16-470	61-17#								
STACK	4-492#	10-23	13-2	13-33	13-180	13-322	13-496	13-644	13-789	13-968	13-;23	13-=09	13->73	13-?96
	13-887	13-D97	13-F29	13-G47	13-H92	13-J43	13-L15							
STANDA	10-66	11-3#												
START	5-1	10-17#	12-33	12-42	14-27									
START1	5-3	10-14#	13-28											
START2	10-15	10-18#												
STCDRV	35-27#													
STIMER	23-14	23-38#												
STKMT	4-492#													
STOP	41-1	41-5#												
STSD1	55-1	55-2	55-8	55-10	55-11	55-12	59-13#							
STSD2	55-1	55-2	55-8	55-10	55-11	55-12	59-14#							
STSD3	59-16#													
STSD4	55-1	55-2	55-8	55-10	55-11	55-12	59-17#							
STSF	56-1	56-1	56-1	56-2	56-2	56-2	56-8	56-8	56-8	56-10	56-10	56-10	56-11	56-11
	56-11	56-12	56-12	56-12	60-7#									
STSH1	54-1	54-2	54-8	54-9	54-11	54-12	54-13	58-21#						
STSH2	54-1	54-2	54-8	54-9	54-11	54-12	54-13	58-23#						
STSH3	58-25#													
STSH4	54-1	54-2	54-8	54-9	54-11	54-12	54-13	58-26#						
SW0	4-492#													
SW00	4-492	4-492#												
SW01	4-492	4-492#												
SW02	4-492	4-492#												
SW03	4-492	4-492#												
SW04	4-492	4-492#												
SW05	4-492	4-492#												
SW06	4-492	4-492#												
SW07	4-492	4-492#												
SW08	4-492	4-492#												
SW09	4-492	4-492#												
SW1	4-492#													
SW10	4-492#													
SW11	4-492#													
SW12	4-492#													
SW13	4-492#	16-260	43-14											
SW14	4-492#													
SW15	4-492#													
SW2	4-492#													
SW3	4-492#													
SW4	4-492#													
SW5	4-492#													
SW6	4-492#													
SW7	4-492#													
SW8	4-492#													
SW9	4-492#													
SWR	6-0#	10-23	10-23	10-23*	10-23*	10-23*	10-28	16-260	41-1	41-1	41-1	41-1	41-1	41-1
	41-1*	41-1*	41-1*	41-1*	42-1	42-1	42-1	42-1	42-1	43-14	44-1	44-1	44-1*	47-1

SWREG	47-1*													
SYSTAT	5-1#	10-23	10-28	44-1	44-1	44-1								
TA1	10-76	49-25#												
TA2	4-544#	12-57												
TA4	4-543#	13-46	13-193	13-335	13-509	13-657	13-802	13-979	13-;37	13-=16	13->68	13-a11	13-C02	13-E05
TAB	13-F36	13-G60	13-J55	13-L19										
TADMSK	4-542#	12-57												
TAG	4-541#	12-57												
TAGADR	4-554#	25-594												
TAP	4-672#													
TBITVE	5-9#	6-0												
TIMOUT	4-648#													
	4-492#													
	13-73	13-115	13-160	13-220	13-262	13-306	13-361	13-403	13-448	13-538	13-581	13-626	13-686	13-729
	13-773	13-831	13-874	13-919	13-:11	13-:60	13-:04	13-:66	13-:96	13-<24	13-<58	13-<86	13-=53	13-=83
	13->21	13->48	13-?21	13-?70	13-a46	13-a88	13-A98	13-C39	13-C82	13-D48	13-E44	13-F04	13-F79	13-G35
	13-G87	13-H29	13-H74	13-I35	13-I77	13-J22	13-J82	13-K23	13-K35	13-K83	13-K96	13-L59	15-136	15-194
	16-95	16-127	16-161	16-190	23-9#									
TKVEC	4-492#	44-1*	44-1*											
TPVEC	4-492#													
TRAPVE	4-492#	10-23*	10-23*											
TRE	4-727#	16-208	24-90	24-107	24-123	25-102	25-104	25-540						
TRTVEC	4-492#													
TST	4-673#	30-103	33-83											
TST1	13-2#	41-1												
TST10	13-968#	41-1												
TST11	13-;23#	41-1												
TST12	13-=09#	41-1												
TST13	13->73#	41-1												
TST14	13-?96#	41-1												
TST15	13-887#	41-1												
TST16	13-D97#	41-1												
TST17	13-F29#	41-1												
TST2	13-33#	41-1												
TST20	13-G47#	41-1												
TST21	13-H92#	41-1												
TST22	13-J43#	41-1												
TST23	13-L15#	41-1												
TST3	13-180#	41-1												
TST4	13-322#	41-1												
TST5	13-496#	41-1												
TST6	13-644#	41-1												
TST7	13-789#	41-1												
TSTNMB	43-36*	43-37*	43-39	43-151#										
TSTPRP	13-41	13-93	13-139	13-188	13-240	13-286	13-330	13-381	13-428	13-504	13-559	13-605	13-652	13-707
	13-753	13-797	13-852	13-899	13-985	13-:40	13-:84	13-:31	13-:86	13-<48	13-=33	13->12	13->98	13-?50
	13-806	13-866	13-A76	13-B97	13-C59	13-D27	13-E18	13-E77	13-F44	13-F99	13-G55	13-H07	13-H53	13-I00
	13-155	13-J01	13-J50	13-K02	13-K58	13-L40	15-38#							
TSTQUE	7-0#	12-8	12-9*	12-51	13-2	13-33	13-180	13-322	13-496	13-644	13-789	13-968	13-;23	13-=09
	13->73	13-?96	13-B87	13-D97	13-F29	13-G47	13-H92	13-J43	13-L15	14-11	14-13*	14-17	14-17*	27-24
	29-17	30-54	33-53											
TYPBN	43-141	46-1#												
TYPDS	14-20	14-20	43-135	46-1#										
TYPE	10-6	10-28	10-42	10-48	10-53	10-59	10-76	10-80	10-82	10-96	10-102	10-105	10-109	10-117
	10-120	10-121	10-122	10-129	11-11	11-16	11-17	11-19	11-20	11-28	11-32	11-34	11-40	11-44
	11-48	11-54	11-61	11-62	11-64	11-69	11-75	11-86	11-87	11-92	11-100	12-3	12-6	12-7

	12-14	12-25	12-30	12-60	12-61	13-52	13-199	13-341	13-515	13-663	13-808	13-988	13-:43	13-a17
	13-C08	13-E11	13-G66	13-I12	13-J61	13-L26	14-20	14-20	14-20	37-1	38-1	39-1	40-1	42-1
	42-1	43-20	43-21	43-31	43-32	43-38	43-43	43-45	43-53	43-96	43-102	43-106	43-118	43-124
	43-126	43-144	43-146	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1	44-1
	44-1	44-1	44-1	44-1	44-1	44-1	44-1	46-1#	47-1					
TYPOC	10-8	11-33	11-47	43-138	44-1	46-1#								
TYPON	46-1#													
TYPOS	10-45	10-56	10-81	12-16	12-62	43-22	43-39	43-44	43-46	46-1#				
UO	4-752#													
U1	4-752#													
U2	4-752#													
UNS	4-574#	13-F48	13-G03	30-70	31-51	31-55	31-59	32-232	32-264	32-269	34-283	34-286	34-291	
UNTMSK	4-756#	24-48	24-50											
UNTOFF	10-109	49-31#												
UNTON	10-120	49-32#												
UPE	4-739#	25-266												
USE	4-695#	13-A06	13-B14	17-29										
USRF IL	16-279	16-282	16-333	61-12#										
VV	4-568#	15-122	16-115	25-192	28-96	28-161	28-162	28-211	28-216	31-23	31-26	32-106	32-145	32-146
	32-169	32-175	33-29	34-90	34-119	34-509	34-510	34-548	34-553	35-35	35-36	35-56	35-61	
WC	4-638#	30-80	33-70											
WCD	4-525#	13-291	13-433	13-758	13-904	13-:89	13->33	13-?55	13-K86					
WCE	4-738#	13-457	13-463	13-928	13-933	25-165	50-77	50-78						
WCEHI	4-763#													
WCELO	4-764#													
WCF	4-583#	4-590	25-236	34-483	34-486									
WCH	4-526#													
WD	4-529#	13-100	13-247	13-388	13-566	13-714	13-859	13-:27	13-:45	13-<11	13-a73	13-C66	13-E09	13-E27
	13-F40	13-F53	13-H14	13-162										
WH	4-530#	13-50	13-197	13-339	13-513	13-661	13-806	13-982	13-:41	13-a15	13-C06	13-G64	13-I10	13-J59
	13-L24	13-L44	34-346											
WLE	4-577#	4-590	25-217	25-221	25-234	25-249	25-264	34-283	34-315	34-319	34-327	50-81	50-82	
WRL	4-563#	25-219	34-322											
XSIZ	10-70#	11-71	11-101											
XXDP	7-0#	10-33*	10-36*	10-37	10-39*	10-44	10-55	10-113	10-115					
Y	49-36#													
ZEROS	13-54	13-201	13-343	13-990	13-a19	13-J63	52-38#							

SSCMRE	5-10#													
SSCMTM	5-10#	6-0	6-0	6-0	6-0	6-0								
SSESCA	4-492#													
SSNEWT	4-492#	13-2	13-33	13-180	13-322	13-496	13-644	13-789	13-968	13-:23	13-=09	13->73	13-?96	13-887
	13-D97	13-F29	13-G47	13-H92	13-J43	13-L15								
SSSET	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1	46-1#
SSSETM	10-23	10-23#												
SSSKIP	4-492#													
.SACT1	4-484#	5-5												
.SAPT8	4-484#	6-0	6-0#											
.SAPTH	4-484#	5-8												
.SAPTY	4-484#	48-1												
.SCATC	4-480#	5-1												
.SCMTA	4-481#	5-10												
.SEOP	4-481#	14-20												
.SERRO	4-481#	42-1												
.SPOWE	4-483#	47-1												
.SRDDE	4-482#													
.SRDOC	4-482#	45-1												
.SREAD	4-482#	44-1												
.SSAVE	4-483#	36-1												
.SSCOP	4-481#	41-1												
.SSIZE	4-483#													
.STRAP	4-483#	46-1												
.STYPB	4-482#	37-1												
.STYPD	4-482#	38-1												
.STYPE	4-481#	40-1												
.STYPO	4-482#	39-1												
.EQUAT	4-480#	4-492												
.HEADE	4-480#	4-488												
.SETUP	4-480#	4-785												
.SWRHI	4-480#	4-489												
.SWRLO	4-480#	4-489#	4-490											
CALCLR	4-192#													
CALSUB	4-206#	13-41	13-52	13-68	13-71	13-73	13-76	13-80	13-93	13-110	13-113	13-115	13-118	13-122
	13-124	13-126	13-139	13-155	13-158	13-160	13-163	13-167	13-169	13-171	13-175	13-188	13-199	13-215
	13-218	13-220	13-223	13-227	13-240	13-257	13-260	13-262	13-265	13-269	13-271	13-273	13-286	13-301
	13-304	13-306	13-309	13-313	13-315	13-317	13-330	13-341	13-356	13-359	13-361	13-364	13-368	13-381
	13-398	13-401	13-403	13-406	13-410	13-412	13-414	13-428	13-443	13-446	13-448	13-451	13-455	13-460
	13-488	13-504	13-515	13-533	13-536	13-538	13-541	13-545	13-559	13-576	13-579	13-581	13-584	13-588
	13-590	13-592	13-605	13-621	13-624	13-626	13-629	13-633	13-635	13-637	13-639	13-652	13-663	13-681
	13-684	13-686	13-689	13-693	13-707	13-724	13-727	13-729	13-732	13-736	13-738	13-740	13-753	13-768
	13-771	13-773	13-776	13-780	13-782	13-784	13-797	13-808	13-826	13-829	13-831	13-834	13-838	13-852
	13-869	13-872	13-874	13-877	13-881	13-883	13-885	13-899	13-914	13-917	13-919	13-922	13-926	13-930
	13-960	13-985	13-988	13-:06	13-:09	13-:11	13-:14	13-:18	13-:40	13-:55	13-:58	13-:60	13-:63	13-:67
	13-:69	13-:71	13-:84	13-:99	13-:02	13-:04	13-:07	13-:11	13-:13	13-:15	13-:31	13-:43	13-:61	13-:64
	13-:66	13-:69	13-:73	13-:86	13-:91	13-:94	13-:96	13-:99	13-<03	13-<21	13-<24	13-<27	13-<31	13-<33
	13-<35	13-<48	13-<53	13-<56	13-<58	13-<61	13-<65	13-<83	13-<86	13-<89	13-<93	13-<95	13-<97	13-<99
	13-=33	13-=48	13-=51	13-=53	13-=56	13-=60	13-=78	13-=81	13-=83	13-=86	13-=95	13-=97	13-=99	13->12
	13->16	13->19	13->21	13->24	13->28	13->43	13->46	13->48	13->51	13->60	13->62	13->64	13->98	13-?16
	13-?19	13-?21	13-?24	13-?33	13-?35	13-?37	13-?50	13-?65	13-?68	13-?70	13-?73	13-?82	13-?84	13-?86
	13-@06	13-@17	13-@41	13-@44	13-@46	13-@49	13-@53	13-@66	13-@83	13-@86	13-@88	13-@91	13-@95	13-A14
	13-A30	13-A43	13-A55	13-A64	13-A76	13-A93	13-A96	13-A98	13-B01	13-B05	13-B22	13-B38	13-B51	13-B63
	13-B72	13-B97	13-C08	13-C34	13-C37	13-C39	13-C42	13-C46	13-C59	13-C77	13-C80	13-C82	13-C85	13-C89
	13-D12	13-D14	13-D27	13-D43	13-D46	13-D48	13-D51	13-D55	13-D78	13-D80	13-D82	13-E11	13-E18	13-E39
	13-E42	13-E44	13-E47	13-E51	13-E56	13-E64	13-E77	13-E99	13-F02	13-F04	13-F07	13-F11	13-F16	13-F24

SKIP	4-492#													
SLASH	4-492#													
STARS	4-492#	5-5	5-8	5-8	5-8	6-0	6-0	6-0	13-2	13-2	13-33	13-33	13-35	13-87
	13-133	13-180	13-180	13-182	13-234	13-280	13-322	13-322	13-324	13-375	13-420	13-496	13-496	13-498
	13-552	13-599	13-644	13-644	13-646	13-700	13-747	13-789	13-789	13-791	13-845	13-891	13-968	13-968
	13-970	13-:33	13-:78	13-:23	13-:23	13-:25	13-:80	13-<42	13-=09	13-=09	13-=11	13->06	13->73	13->73
	13->75	13-?44	13-?96	13-?96	13-?98	13-@60	13-A70	13-887	13-887	13-889	13-C53	13-D21	13-D97	13-D97
	13-D99	13-E71	13-F29	13-F29	13-F31	13-F93	13-G47	13-G47	13-G49	13-H01	13-H47	13-H92	13-H92	13-H94
	13-149	13-195	13-J43	13-J43	13-J45	13-J98	13-K53	13-L15	13-L15	14-20	15-57	15-70	15-84	15-107
	15-138	15-160	15-196	16-212	16-298	16-302	25-115	25-529	25-531	36-1	37-1	38-1	39-1	40-1
	41-1	42-1	44-1	44-1	44-1	44-1	44-1	45-1	46-1	47-1	47-1	48-1		
SWRSU	4-492#	10-23	10-23#											
TAGS	4-103#	6-0												
TRMTRP	46-1#													
TYPBIN	4-492#													
TYPDEC	4-492#	14-20	14-20											
TYPNAM	4-480#	4-492#	10-28											
TYPNUM	4-492#													
TYPOCS	4-492#	10-81	12-16	12-62	43-22	43-39	43-44	43-46						
TYPOCT	4-492#	11-33	44-1											
TYPTXT	4-492#	10-6	10-42	10-53	10-59	12-30	14-20	14-20						
XPER	4-12#	8-3	8-6	8-9	8-12	8-15	8-18	8-21	8-24	8-27	8-30	8-33	8-36	8-39
	8-42	8-45	8-48	8-51	8-54	8-57	8-60	8-63	8-66	8-69	8-72	8-75	8-78	8-81
	8-84	8-87	8-90	8-93	8-96	8-99	8-102	8-105	8-108	8-111	8-114	8-117	8-120	8-123
	8-126	8-130	8-133	8-136	8-139	8-142	8-146	8-150	8-154	8-157	8-160	8-163	8-166	8-169
	8-172	8-175	8-178	8-182	8-185	8-188	8-191	8-194	8-197	8-200	8-203	8-206	8-209	8-212
	8-215	8-218	8-221	8-224	8-227	8-230	8-233	8-236	8-239	8-242	8-245	8-248	8-251	8-254
	8-257	8-260	8-263	8-266	8-269	8-272	8-275	8-278	8-281	8-284	8-287	8-290	8-293	8-296
	8-299	8-302	8-305	8-308	8-311	8-314	8-317	8-320	8-323	8-326	8-329	8-332	8-335	8-338
	8-341	8-344	8-347	8-351	8-354	8-357	8-360	8-363	8-366	8-369	8-372	8-375	8-378	8-381
	8-384	8-387	8-390	8-393	8-396	8-399	8-402	8-405	8-408	8-411	8-414	8-417	8-420	8-423
	8-426	8-429	8-432	8-435	8-438	8-441	8-444	8-447	8-450	8-453	8-456	8-459	8-462	8-465
	8-468	8-471	8-474	8-477	8-480	8-483	8-486	8-489	8-492	8-495	8-498	8-501	8-504	8-507
	8-510	8-513	8-516	8-519	8-522	8-525	8-528	8-531	8-534	8-537	8-540	8-543	8-546	8-549
	8-552	8-556	8-559	8-563	8-566	8-569	8-573	8-577	8-581	8-586	8-590	8-594	8-597	8-600
	8-603	8-606	8-609	8-612	8-615	8-618	8-621	8-624	8-627	8-630	8-633	8-636	8-639	8-642
	8-645	8-648	8-651	8-654	8-657	8-660	8-663	8-666	8-669	8-672	8-675	8-678	8-681	8-684
	8-687	8-690	8-693	8-696	8-699	8-702	8-705	8-708	8-711	8-714	8-717	8-720	8-723	