

RM05/3/2

RM05/3/2 FCTNL TST2
CZRMNAO

AH-F925A-MC
FICHE 2 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



Table with multiple columns and rows of data, including headers and numerical values. The table is mostly illegible due to low contrast and blurring. A handwritten number '5' is visible in the lower right quadrant of the table area.

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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-F924A-MC
PRODUCT NAME: CZRMNAO RM05/3/2 FCTNL TST 2
DATE CREATED: APRIL 1980
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

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

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

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RM05/3/2 DISK SUBSYSTEM. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

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

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

TO VERIFY THE FUNCTIONALITY OF THE RM05/3/2 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

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

1.2 UNIT UNDER TEST

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

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

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

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

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

2.2 MEDIA REQUIREMENTS

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

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

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

RM05/3/2 FUNCTIONAL TEST, PART 1

3.0 OPERATING PROCEDURE

3.1 LOADING

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

.PAPER TAPE, USING THE STANDARD PAPER TAPE LOADING PROCEDURE.

.XXDP MEDIA, USING THE APPROPRIATE LOADING DEVICE.

3.2 SWITCH OPTIONS

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

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

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

3.3 STARTING

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

3.4 HALTING

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

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

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

3.5 RESTARTING

THE PROGRAM CAN BE RESTARTED AT ADDRESS 200.

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

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

4.2 CONSOLE DIALOGUE

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

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

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

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

THE THIRD QUESTION TYPED IS, 'TYPE "A" TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S) AND TERMINATE INPUT WITH A CARRIAGE RETURN.' THEN, 'DRIVE(S):' IS TYPED AND WAITS FOR THE OPERATOR TO TYPE AN "A", TO TEST ALL POSSIBLE DRIVES OR THE NUMBER(S) OF THE DRIVE(S) HE WANTS TESTED AND TERMINATE HIS INPUT WITH A "CARRIAGE RETURN".

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

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

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

4.3 PROGRESS REPORTS

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

4.4 PERFORMANCE REPORT

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

4.5 PROGRAM HALTS

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

4.6 ERROR REPORTS

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

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

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

5.2 DUAL PORT CONFIGURATIONS

THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST

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

DUAL PORT LOGIC IN THE RM05/3/2 ADAPTER BUT IS EXECUTABLE ON RM05/3/2 SUBSYSTEMS HAVING THE DUAL PORT OPTION PROVIDING THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE IS NOT USED DURING THE EXECUTION OF THE RM05/3/2 SUBSYSTEM FUNCTIONAL TEST.

5.4 MEMORY MANAGEMENT HARDWARE

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

5.5 ACT11, APT11 COMPATIBILITY

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

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

6.0 TEST DESCRIPTION

CONTROLLER ACCESS TEST

PURPOSE:

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

PROCEDURE:

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

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

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

DEVICE AVAILABLE TEST

PURPOSE:

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

PROCEDURE:

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

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

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

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

.DEVICE AVAILABLE - THE DEVICE IS AVAILABLE FOR TESTING.

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

DRIVE TYPE TEST

PURPOSE:

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

PROCEDURE:

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

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

WRITE/READ HEADER AND DATA (FORMAT) TESTS

PURPOSE:

TO TEST WRITE HEADER AND DATA AND READ HEADER AND DATA FUNCTIONALITY OF THE RM05/3/2 SUBSYSTEM USING A SET OF VARIABLES WHICH INCLUDE WORD COUNT, HEAD MOTION, HEAD SWITCHING AND ERROR CONDITIONS.

PROCEDURE:

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

NOTE:

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

FORMAT ZEROS TEST

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

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

SEEK. THE TEST IS REPEATED FOR 16 BIT FORMAT.

ZERO FILL TEST TEST

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

FORMAT CHECK ZEROS TEST

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

FORMAT CHECK ZEROS W/ WCE ERROR TEST

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

FORMAT ONES TEST

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

FORMAT CHECK ONES TEST

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

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

FORMAT CHECK ONES W/ WCE ERROR TEST

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

FORMAT MULTIPLE SECTORS TEST

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE WRITE HEADER AND DATA COMMAND FOLLOWS, WITH THE WORD COUNT EQUAL TO MULTIPLE SECTORS. THE SAME SECTORS ARE VERIFIED WITH A WRITE CHECK HEADER AND DATA COMMAND.

FORMAT WITH HEAD SWITCHING TEST

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

FORMAT WITH MID-TRANSFER SEEK TEST

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

FORMAT WITH IMPLIED SEEK TEST

THIS TEST SEEKS TO THE LAST CYLINDER PRIOR TO WRITING HEADER AND DATA ON CYLINDER 0. THE EXPLICIT SEEK INSURES THAT THERE WILL BE MAXIMUM HEAD MOTION DURING THE IMPLIED SEEK OF THE WRITE

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

COMMAND. THE SAME OPERATION, INCLUDING THE EXPLICIT SEEK IS REPEATED FOR READ HEADER AND DATA.

FORMAT EACH SECTOR ADDRESS TEST

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

FORMAT EACH TRACK ADDRESS TEST

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

FORMAT PRIME CYLINDERS TEST

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

READ HEADER & DATA IN LAST SECTOR TEST

THIS TEST READS HEADER AND DATA ON THE LAST SECTOR OF THE DISK, I.E., AND VERIFIES THAT "LBT" STATUS SETS.

READ HEADER & DATA W/ AOE ERROR TEST

THIS TEST READS MULTIPLE SECTORS STARTING WITH THE LAST SECTOR 31. AND VERIFIES THAT AOE STATUS SETS.

READ INVALID SECTOR ADDRESS TEST

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

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

READ INVALID TRACK ADDRESS TEST

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

READ INVALID CYLINDER ADDRESS TEST

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

FORMAT AT OFFSET TEST

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

IVC FORMAT TEST

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

FORMAT ERROR TEST

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

FORMAT HCE TEST (1ST AND 2ND HEADER WORDS)

628
629
630
631
632
633
634
635
636
637
638

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

1
486
487

:PROGRAM REVISION #001

.TITLE CZRMNAO RM05/3/2 FCTNL TST 2

:*COPYRIGHT (C) 1980

:*DIGITAL EQUIPMENT CORP.

:*MAYNARD, MASS. 01754

:*

:*PROGRAM BY MIKE LEAVITT

:*

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

:*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.

:*

488

.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

489

490
491

.SBTTL BASIC DEFINITIONS

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

001100
104000
000004

STACK = 1100

ERROR = EMT

SCOPE = IOT

::BASIC DEFINITION OF ERROR CALL

::BASIC DEFINITION OF SCOPE CALL

:*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

HT = 11

LF = 12

CR = 15

CRLF = 200

PS = 177776

PSW=PS

STKLMT = 177774

PIRQ = 177772

DSWR = 177570

DDISP = 177570

::CODE FOR HORIZONTAL TAB

::CODE FOR LINE FEED

::CODE FOR CARRIAGE RETURN

::CODE FOR CARRIAGE RETURN-LINE FEED

::PROCESSOR STATUS WORD

::STACK LIMIT REGISTER

::PROGRAM INTERRUPT REQUEST REGISTER

::HARDWARE SWITCH REGISTER

::HARDWARE DISPLAY REGISTER

:*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003

R0 = %0

R1 = %1

R2 = %2

R3 = %3

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

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

;*PRIORITY LEVEL DEFINITIONS

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

;'SWITCH REGISTER' SWITCH DEFINITIONS

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

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

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

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

.SBTTL RM REGISTER BIT DEFINITIONS

;*RMCS1 CONTROL STATUS REGISTER

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

;FUNCTION CODES (BITS 01-05 OF RMCS1)

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

```

520      000030      SEARCH = 000030      ;SEARCH COMMAND
523      000030      ILF30  = 000030      ;ILLEGAL COMMAND
          000032      ILF32  = 000032      ;ILLEGAL COMMAND
          000034      ILF34  = 000034      ;ILLEGAL COMMAND
          000036      ILF36  = 000036      ;ILLEGAL COMMAND
          000040      ILF40  = 000040      ;ILLEGAL COMMAND
          000042      ILF42  = 000042      ;ILLEGAL COMMAND
          000044      ILF44  = 000044      ;ILLEGAL COMMAND
          000046      ILF46  = 000046      ;ILLEGAL COMMAND
524      000050      WCD     = 000050      ;WRITE CHECK DATA COMMAND
525      000052      WCH     = 000052      ;WRITE CHECK HEADER AND DATA
526      000054      ILF54  = 000054      ;ILLEGAL COMMAND
527      000056      ILF56  = 000056      ;ILLEGAL COMMAND
528      000060      WD      = 000060      ;WRITE DATA COMMAND
529      000062      WH      = 000062      ;WRITE HEADER AND DATA COMMAND
530      000064      ILF64  = 000064      ;ILLEGAL COMMAND
531      000066      ILF66  = 000066      ;ILLEGAL COMMAND
532      000070      RD      = 000070      ;READ DATA COMMAND
533      000072      RH      = 000072      ;READ HEADER AND DATA COMMAND
534      000074      ILF74  = 000074      ;ILLEGAL COMMAND
535      000076      ILF76  = 000076      ;ILLEGAL COMMAND
536
537      ;*RMDA DISK ADDRESS REGISTER
538
539      ;TRACK ADDRESS DEFINITIONS
540      010000      TA16   = BIT12      ;TRACK ADDRESS 16.
541      004000      TA8    = BIT11      ;TRACK ADDRESS 8.
542      002000      TA4    = BIT10      ;TRACK ADDRESS 4
543      001000      TA2    = BIT09      ;TRACK ADDRESS 2
544      000400      TA1    = BIT08      ;TRACK ADDRESS 1
545
546      ;SECTOR ADDRESS DEFINITIONS
547      000020      SA16   = BIT04      ;SECTOR ADDRESS 16.
548      000010      SA8    = BIT03      ;SECTOR ADDRESS 8.
549      000004      SA4    = BIT02      ;SECTOR ADDRESS 4
550      000002      SA2    = BIT01      ;SECTOR ADDRESS 2
551      000001      SA1    = BIT00      ;SECTOR ADDRESS 1
552
553      ;TRACK & SECTOR MASKS
554      177400      TADMSK = 177400      ;TRACK ADDRESS MASK
555      000377      SADMSK = 000377      ;SECTOR ADDRESS MASK
556
557      ;*RMDS DRIVE STATUS REGISTER
558
559      100000      ATA     = BIT15      ;ATTENTION ACTIVE
560      040000      ERR     = BIT14      ;COMPOSITE ERROR
561      020000      PIP     = BIT13      ;POSITIONING IN PROGRESS
562      010000      MOL     = BIT12      ;MEDIUM ON LINE
563      004000      WRL     = BIT11      ;WRITE LOCK
564      002000      LBT     = BIT10      ;LAST BLOCK TRANSFERRED
565      001000      PGM     = BIT09      ;PROGRAMMABLE
566      000400      DPR     = BIT08      ;DRIVE PRESENT
567      000200      DRY     = BIT07      ;DRIVE READY
568      000100      VV      = BIT06      ;VOLUME VALID
569      000001      OM      = BIT00      ;OFFSET MODE ACTIVE
570
571      ;*RMER1 ERROR REGISTER #1

```

```

572
573      100000      DCK      = BIT15      ;DATA CHECK ERROR
574      040000      UNS      = BIT14      ;DRIVE UNSAFE
575      020000      OPI      = BIT13      ;OPERATION INCOMPLETE
576      010000      DTE      = BIT12      ;DRIVE TIMING ERROR
577      004000      WLE      = BIT11      ;WRITE LOCK ERROR
578      002000      IAE      = BIT10      ;INVALID ADDRESS ERROR
579      001000      AOE      = BIT09      ;ADDRESS OVERFLOW ERROR
580      000400      HCRC     = BIT08      ;HEADER CRC ERROR
581      000200      HCE      = BIT07      ;HEADER COMPARE ERROR
582      000100      ECH      = BIT06      ;ECC "HARD" ERROR
583      000040      WCF      = BIT05      ;WRITE CLOCK FAILURE
584      000020      FER      = BIT04      ;FORMAT ERROR
585      000010      PAR      = BIT03      ;PARITY ERROR
586      000004      RMR      = BIT02      ;REGISTER MODIFICATION REFUSED
587      000002      ILR      = BIT01      ;ILLEGAL REGISTER
588      000001      ILF      = BIT00      ;ILLEGAL FUNCTION
589
590      115760      NDTMSK   = DCK!DTE!WLE!AOE!HCRC!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	020024	SNGPRT	= 020024	:SINGLE PORT DRIVE TYPE (RM02)
653	024024	DULPRT	= 024024	:DUAL PORT DRIVE TYPE (RM02)
654				
655		:*RMOF OFFSET REGISTER		
656				
657	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
658	004000	ECI	= BIT11	:ECC INHIBIT
659	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
660	000200	OFD	= BIT07	:OFFSET FORWARD
661				
662		:*RMDC DESIRED CYLINDER ADDRESS REGISTER		
663				
664	001777	CYLMSK	= 001777	:MASK FOR CYLINDER ADDRESS
665				
666		:*RMMR2 MAINTENANCE REGISTER #2		
667				
668		:READ ONLY BITS		
669	100000	RQA	= BIT15	:PORT A REQUEST
670	040000	RQB	= BIT14	:PORT B REQUEST
671	020000	TAG	= BIT13	:TAG CONTROL
672	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
673	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
674	002000	CH	= BIT10	:CONTROL OR HEAD TAG
675	001000	BB09	= BIT09	:TAG BUS
676	000400	BB08	= BIT08	:TAG BUS
677	000200	BB07	= BIT07	:TAG BUS
	000100	BB06	= BIT06	:TAG BUS
	000040	BB05	= BIT05	:TAG BUS
	000020	BB04	= BIT04	:TAG BUS
	000010	BB03	= BIT03	:TAG BUS
	000004	BB02	= BIT02	:TAG BUS
	000002	BB01	= BIT01	:TAG BUS
	000001	BB00	= BIT00	:TAG BUS

678

```
679          ;*RMR2 ERROR REGISTER 2
680
681          100000      BSE      = BIT15          ;BAD SECTOR ERROR
682          040000      SKI      = BIT14          ;SEEK INCOMPLETE
683          020000      OPE      = BIT13          ;OPERATOR PLUG ERROR
684          010000      IVC      = BIT12          ;INVALID COMMAND ERROR
685          004000      LSC      = BIT11          ;LOSS OF SYSTEM CLOCK
686          002000      LBC      = BIT10          ;LOSS OF BIT CLOCK
687          000200      DVC      = BIT07          ;DEVICE CHECK
688          000010      DPE      = BIT03          ;DATA PARITY ERROR
689
690          .SBTTL      PROGRAM MNEMONICS
691
692          100000      MSE      = BIT15          ;MANUFACTURING DETECTED SECTOR ERROR
693          040000      USE      = BIT14          ;USER DETECTED SECTOR ERROR
694
695          .SBTTL      RM REGISTER INDEX VALUES
696
697          000000      RMCS1    = 00            ;CONTROL STATUS REGISTER #1
698          000006      RMDA     = 06            ;DISK ADDRESS REGISTER
699          000012      RMDS     = 12            ;DRIVE STATUS REGISTER
700          000014      RMR1     = 14            ;ERROR REGISTER #1
701          000016      RMAS     = 16            ;ATTENTION SUMMARY REGISTER
702          000020      RMLA     = 20            ;LOOK AHEAD REGISTER
703          000024      RMMR1    = 24            ;MAINTENANCE REGISTER
704          000026      RMDT     = 26            ;DRIVE TYPE REGISTER
705          000030      RMSN     = 30            ;SERIAL NUMBER REGISTER
706          000032      RMOF     = 32            ;OFFSET REGISTER
707          000034      RMDC     = 34            ;DESIRED CYLINDER REGISTER
708          000036      RMHR     = 36            ;HOLDING REGISTER
709          000040      RMMR2    = 40            ;MAINTENANCE REGISTER #2
710          000042      RMR2     = 42            ;ERROR REGISTER #2
711          000044      RMEC1    = 44            ;ECC POSITION REGISTER
712          000046      RMEC2    = 46            ;ECC PATTERN REGISTER
713          000050      ILRG50    = 50            ;ILLEGAL REGISTER 50
714          000052      ILRG52    = 52            ;ILLEGAL REGISTER 52
715          000054      ILRG54    = 54            ;ILLEGAL REGISTER 54
716          000056      ILRG56    = 56            ;ILLEGAL REGISTER 56
717          000060      ILRG60    = 60            ;ILLEGAL REGISTER 60
718          000062      ILRG62    = 62            ;ILLEGAL REGISTER 62
719          000064      ILRG64    = 64            ;ILLEGAL REGISTER 64
720          000066      ILRG66    = 66            ;ILLEGAL REGISTER 66
721          000070      ILRG70    = 70            ;ILLEGAL REGISTER 70
722          000072      ILRG72    = 72            ;ILLEGAL REGISTER 72
723          000074      ILRG74    = 74            ;ILLEGAL REGISTER 74
724          000076      ILRG76    = 76            ;ILLEGAL REGISTER 76
725
726          000077      IDXMSK   = 77            ;MASK FOR REGISTER INDEX NUMBER
727
728          .SBTTL      RH CONTROLLER REGISTER BIT DEFINITIONS
729
730          ;*RMCS1 CONTROL STATUS REGISTER #1
731
732          100000      SC        = BIT15          ;SPECIAL CONDITION-READ ONLY
733          040000      TRE        = BIT14          ;TRANSFER ERROR
734          020000      MCPE       = BIT13          ;MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
735          002000      PSEL       = BIT10          ;PORT B SELECT
```

```

727      001000      A17      = BIT09      ;ADDRESS EXTENSION
728      000400      A16      = BIT08      ;ADDRESS EXTENSION
729      000200      RDY      = BIT07      ;READY-READ ONLY
730      000100      IE       = BIT06      ;INTERRUPT ENABLE
731
732      ;*RMCS2 RH CONTROL STATUS REGISTER #2
733
734      100000      DLT      = BIT15      ;DATA LATE-READ ONLY
735      040000      WCE      = BIT14      ;WRITE CHECK ERROR-READ ONLY
736      020000      UPE      = BIT13      ;UNIBUS PARITY ERROR
737      010000      NED      = BIT12      ;NONEXISTANT DRIVE-READ ONLY
738      004000      NEM      = BIT11      ;NONEXISTANT MEMORY-READ ONLY
739      002000      PGE      = BIT10      ;PROGRAM ERROR-READ ONLY
740      001000      MXF      = BIT09      ;MISSED TRANSFER
741      000400      MDPE     = BIT08      ;MASSBUS DATA BUS PARITY ERROR-READ ONLY
742      000200      OR       = BIT07      ;OUTPUT READY-READ ONLY
743      000100      IR       = BIT06      ;INPUT READY-READ ONLY
744      000040      CLR      = BIT05      ;CONTROLLER CLEAR
745      000020      PAT      = BIT04      ;PARITY TEST
746      000010      BAI      = BIT03      ;UNIBUS ADDRESS INCREMENT INHIBIT
749      000004      U2       = BIT02      ;UNIT SELECT
          000002      U1       = BIT01      ;UNIT SELECT
          000001      U0       = BIT00      ;UNIT SELECT

750
751      ;UNIT SELECT MASK
752
753      000007      UNTMSK   = 7          ;UNIT SELECT MASK
754
755      ;*RMCS3 RH70 CONTROL STATUS REGISTER #3
756
757      100000      APE      = BIT15      ;ADDRESS PARITY ERROR
758      040000      DPEHI    = BIT14      ;DATA PARITY ERROR HIGH WORD
759      020000      DPELO    = BIT13      ;DATA PARITY ERROR LOW WORD
760      010000      WCEHI    = BIT12      ;WRITE CHECK ERROR HIGH WORD
761      004000      WCELO    = BIT11      ;WRITE CHECK ERROR LOW WORD
762      002000      DBL      = BIT10      ;DOUBLE WORD TRANSFER
763      000100      IE       = BIT06      ;INTERRUPT ENABLE
764      000010      IPCK3    = BIT03      ;INVERT PARITY CHECK
765      000004      IPCK2    = BIT02      ;INVERT PARITY CHECK
766      000002      IPCK1    = BIT01      ;INVERT PARITY CHECK
767      000001      IPCK0    = BIT00      ;INVERT PARITY CHECK
768
769      .SBTTL RH11/RH70 CONTROLLER REGISTER INDEX VALUES
770
771      000000      RMCS1     = 00          ;CONTROL, STATUS REGISTER #1
772      000002      RMWC      = 02          ;WORD COUNT REGISTER
773      000004      RMBA      = 04          ;BUS ADDRESS REGISTER
774      000010      RMCS2     = 10          ;CONTROL, STATUS REGISTER #2
775      000022      RMDB      = 22          ;DATA BUFFER
776      000050      RMBAE     = 50          ;BUS ADDRESS EXTENSION
777      000052      RMCS3     = 52          ;CONTROL, STATUS REGISTER #3
778
779      176700      ABASE      = 176700     ;UNIBUS ADDRESS
780      120254      AVECT1    = 120254     ;UNIBUS VECTOR ADDRESS AND PRIORITY
781
783
784      .SBTTL TRAP CATCHER
  
```



```
000000      .=0
              ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
              ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
              ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
000174      000174
000176      000000      .=174
DISPREG:    .WORD 0      ;;SOFTWARE DISPLAY REGISTER
SWREG:      .WORD 0      ;;SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)
000200      000137 005420      JMP @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
785
786
.SBTTL ACT11 HOOKS
*****
;HOOKS REQUIRED BY ACT11
000204      000204      $$VPC=.      ;SAVE PC
000046      000046      .=46
032530      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
000052      000052      .=52
000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
000204      .=$VPC      ;; RESTORE PC
787
788      001100      .=1100
789      .SBTTL APT PARAMETER BLOCK
*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
*****
001100      .SX=.      ;;SAVE CURRENT LOCATION
000024      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000200      200      ;;FOR APT START UP
000044      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044      $APTHDR    ;;POINT TO APT HEADER BLOCK
001100      .=$X      ;;RESET LOCATION COUNTER
*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.
001100      $APTHD:
001100      000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102      001222      $MBADR: .WORD $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104      000001      $STMT: .WORD 1      ;;RUN TIM OF LONGEST TEST
001106      000002      $PASTM: .WORD 2      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110      000002      $UNITM: .WORD 2      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112      000042      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
790      001114      TAGADR=.
```

0

.SBTTL COMMON TAGS

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

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

.SBTTL APT MAILBOX-ETABLE

001222				..EVEN	
001222	000000			\$MAIL: .WORD	::APT MAILBOX
001224	000000			\$MSGTY: .WORD	AMSGTY ::MESSAGE TYPE CODE
001226	000000			\$FATAL: .WORD	AFATAL ::FATAL ERROR NUMBER
				\$TESTN: .WORD	ATESTN ::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	\$DEVN:	.WORD	ADEVN	::DEVICE MAP
001302	000000	\$CDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001304	000000	\$CDW2:	.WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001306	000000	\$DDW0:	.WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
001310	000000	\$DDW1:	.WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
001312	000000	\$DDW2:	.WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
001314	000000	\$DDW3:	.WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
001316	000000	\$DDW4:	.WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
001320	000000	\$DDW5:	.WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
001322	000000	\$DDW6:	.WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
001324	000000	\$DDW7:	.WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
001326		\$ETEND:			
		.MEXIT			

0

.SBTTL USER DEFINED TAGS

001326 000000
 001330 000000

CTLFG: .WORD 0 ;CONTAINS CONTROL-C FLAG
 XXDP: .WORD 0 ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
 ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
 ;'XXDP' DEVICE CODE FOR THE RM05/3/2.
 LSTRK: .BYTE 0 ;LO BYTE = 0
 ;HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT
 ;UNDER TEST. RM02/3 = 4., RM05 = 18.

001332 000
 001333 000

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

001334

GETBUF:

001334 000000
 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

;REGISTER INPUT BUFFER
 RMCS11: .WORD 0 ;CONTROL, STATUS REGISTER #1
 RMWCI: .WORD 0 ;WORD COUNT REGISTER
 RMBAI: .WORD 0 ;BUS ADDRESS REGISTER
 RMDAI: .WORD 0 ;DISK ADDRESS REGISTER
 RMCS21: .WORD 0 ;CONTROL, STATUS REGISTER #2
 RMDSI: .WORD 0 ;DRIVE STATUS REGISTER
 RMER11: .WORD 0 ;ERROR REGISTER #1
 RMASI: .WORD 0 ;ATTENTION SUMMARY REGISTER
 RMLA1: .WORD 0 ;LOOK AHEAD REGISTER
 RMDBI: .WORD 0 ;DATA BUFFER
 RMMR11: .WORD 0 ;MAINTENANCE REGISTER #1
 RMDTI: .WORD 0 ;DRIVE TYPE REGISTER
 RMSNI: .WORD 0 ;SERIAL NUMBER REGISTER
 RMOF1: .WORD 0 ;OFFSET REGISTER
 RMDC1: .WORD 0 ;DESIRED CYLINDER REGISTER
 RMHRI: .WORD 0 ;HOLDING REGISTER
 RMMR21: .WORD 0 ;MAINTENANCE REGISTER #2
 RMER21: .WORD 0 ;ERROR REGISTER #2
 RMEC11: .WORD 0 ;ECC POSITION REGISTER
 RMEC21: .WORD 0 ;ECC PATTERN REGISTER
 RMBAE1: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER
 RMCS31: .WORD 0 ;CONTROL, STATUS REGISTER #3

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

001410

PUTBUF:

001410 000000
 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
 RMCS10: .WORD 0 ;CONTROL, STATUS REGISTER #1
 RMWCO: .WORD 0 ;WORD COUNT REGISTER
 RMBAO: .WORD 0 ;BUS ADDRESS REGISTER
 RMDAO: .WORD 0 ;DISK ADDRESS REGISTER
 RMCS20: .WORD 0 ;CONTROL, STATUS REGISTER #2
 RMDSO: .WORD 0 ;DRIVE STATUS REGISTER
 RMER10: .WORD 0 ;ERROR REGISTER #1
 RMASO: .WORD 0 ;ATTENTION SUMMARY REGISTER
 RMLAO: .WORD 0 ;LOOK AHEAD REGISTER
 RMDBO: .WORD 0 ;DATA BUFFER
 RMMR10: .WORD 0 ;MAINTENANCE REGISTER #1
 RMDTO: .WORD 0 ;DRIVE TYPE REGISTER

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

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

001464	000000	TSTQUE: .WORD	0	;CONTAINS DEVICE POINTER
		.BLKW	8.	;TEST QUE FOR DEVICES UNDER TEST
001506	000000	.WORD	0	;TABLE TERMINATOR GOES HERE WHEN ;ALL 8. DEVICES ARE UNDER TEST.

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

001510	000000	MEDENB: .WORD	0	;MEDIA ENABLE
--------	--------	---------------	---	---------------

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

001512	000000	ASNDC: .WORD	0	;ASSIGNED DESIRED CYLINDER
001514	000000	ASNDA: .WORD	0	;ASSIGNED TRACK, AND SECTOR
001516	000000	CLKADR: .WORD	0	;UNIBUS ADDRESS OF KW11 CLOCK
001520	000000	CLKVCT: .WORD	0	;VECTOR ADDRESS OF KW11 CLOCK

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

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

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

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

;PUT TAGS HERE

0

.SBTTL ERROR POINTER TABLE

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

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

001600

\$ERRTB:

1
2
3
4

;ERROR 1 WRONG UNIT SELECTED

001600 065144
001602 071236
001604 071362
001606 071452

EMT1
EHT1
EDT1
EFT1

5
6
7

;ERROR 2 DEVICE WENT UNAVAILABLE

001610 065150
001612 071236
001614 071362
001616 071452

EMT2
EHT1
EDT1
EFT1

8
9
10

;ERROR 3 DEVICE WENT NONEXISTENT

001620 065156
001622 071236
001624 071362
001626 071452

EMT3
EHT1
EDT1
EFT1

11
12
13

;ERROR 4 CONTROLLER NOT READY

001630 065164
001632 071236
001634 071362
001636 071452

EMT4
EHT1
EDT1
EFT1

14
15
16

;ERROR 5 DRIVE NOT READY AND GO NOT RESET

001640 065172
001642 071236
001644 071362
001646 071452

EMT5
EHT1
EDT1
EFT1

17

18
19
001650 065200
001652 071236
001654 071362
001656 071452

:ERROR 6 UNEXPECTED VALUE FOR 'ATA' STATUS
EMT6
EHT1
EDT1
EFT1

20
21
22
001660 065206
001662 000000
001664 000000
001666 000000

:ERROR 7 BUS TIMEOUT TRYING TO READ OR WRITE REGISTER
EMT7
0
0
0

23
24
25
001670 065214
001672 071236
001674 071362
001676 071452

:ERROR 10 DRIVE NOT READY BUT GO IS RESET
EMT10
EHT1
EDT1
EFT1

26
27
28
001700 065220
001702 071236
001704 071362
001706 071452

:ERROR 11 GO NOT RESET BUT DRIVE IS READY
EMT11
EHT1
EDT1
EFT1

29
30
31
001710 065224
001712 071236
001714 071362
001716 071452

:ERROR 12 INCORRECT FUNCTION CODE
EMT12
EHT1
EDT1
EFT1

32
33
34
001720 065232
001722 071236
001724 071362
001726 071452

:ERROR 13 PARITY ERROR READING REMOTE REGISTERS
EMT13
EHT1
EDT1
EFT1

35
36
37
001730 065244
001732 071236
001734 071362
001736 071452

:ERROR 14 TRANSFER ERROR IS INCORRECT
EMT14
EHT1
EDT1
EFT1

38
39
:ERROR 15 INCORRECT WORD COUNT

40	001740 065252	EMT15	
	001742 071236	EHT1	
	001744 071362	EDT1	
	001746 071452	EFT1	
41			
42		:ERROR 16	INCORRECT BUS ADDRESS
43			
	001750 065260	EMT16	
	001752 071236	EHT1	
	001754 071362	EDT1	
	001756 071452	EFT1	
44			
45		:ERROR 17	INCORRECT LBT STATUS
46			
	001760 065270	EMT17	
	001762 071236	EHT1	
	001764 071362	EDT1	
	001766 071452	EFT1	
47			
48		:ERROR 20	INCORRECT AOE
49			
	001770 065300	EMT20	
	001772 071236	EHT1	
	001774 071362	EDT1	
	001776 071452	EFT1	
50			
51		:ERROR 21	INCORRECT DISK ADDRESS
52			
	002000 065310	EMT21	
	002002 071236	EHT1	
	002004 071362	EDT1	
	002006 071452	EFT1	
53			
54		:ERROR 22	INCORRECT CYLINDER ADDRESS
55			
	002010 065320	EMT22	
	002012 071236	EHT1	
	002014 071362	EDT1	
	002016 071452	EFT1	
56			
57		:ERROR 23	INCORRECT WLE STATUS
58			
	002020 065330	EMT23	
	002022 071236	EHT1	
	002024 071362	EDT1	
	002026 071452	EFT1	
59			
60		:ERROR 24	INCORRECT UPE STATUS
61			

	002030	065340		EMT24
	002032	071236		EHT1
	002034	071362		EDT1
	002036	071452		EFT1
62				
63			:ERROR	25 INCORRECT WCF STATUS
64				
	002040	065350		EMT25
	002042	071236		EHT1
	002044	071362		EDT1
	002046	071452		EFT1
65				
66			:ERROR	26 INCORRECT WCE STATUS
67				
	002050	065360		EMT26
	002052	071236		EHT1
	002054	071362		EDT1
	002056	071452		EFT1
68				
69			:ERROR	27 INCORRECT MDPE STATUS
70				
	002060	065370		EMT27
	002062	071236		EHT1
	002064	071362		EDT1
	002066	071452		EFT1
71				
72			:ERROR	30 INCORRECT DCK STATUS
73				
	002070	065400		EMT30
	002072	071236		EHT1
	002074	071362		EDT1
	002076	071452		EFT1
74				
75			:ERROR	31 INCORRECT ECH STATUS
76				
	002100	065410		EMT31
	002102	071236		EHT1
	002104	071362		EDT1
	002106	071452		EFT1
77				
78			:ERROR	32 DLT SHOULD NOT BE SET
79				
	002110	065420		EMT32
	002112	071236		EHT1
	002114	071362		EDT1
	002116	071452		EFT1
80				
81			:ERROR	33 MXF SHOULD NOT BE SET
82				
	002120	065430		EMT33

ERROR POINTER TABLE				
	002122 071236		EHT1	
	002124 071362		EDT1	
	002126 071452		EFT1	
83				
84		:ERROR 34		DTE SHOULD NOT BE SET
85				
	002130 065440		EMT34	
	002132 071236		EHT1	
	002134 071362		EDT1	
	002136 071452		EFT1	
86				
87		:ERROR 35		INCORRECT HCRC STATUS
88				
	002140 065450		EMT35	
	002142 071236		EHT1	
	002144 071362		EDT1	
	002146 071452		EFT1	
89				
90		:ERROR 36		INCORRECT HCE STATUS
91				
	002150 065460		EMT36	
	002152 071236		EHT1	
	002154 071362		EDT1	
	002156 071452		EFT1	
92				
93		:ERROR 37		INCORRECT FER STATUS
94				
	002160 065470		EMT37	
	002162 071236		FHT1	
	002164 071362		EDT1	
	002166 071452		EFT1	
95				
96		:ERROR 40		DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
97				
	002170 065500		EMT40	
	002172 071236		EHT1	
	002174 071362		EDT1	
	002176 071452		EFT1	
98				
99		:ERROR 41		LOST "MOL" DURING PACK ACKNOWLEDGE
100				
	002200 065506		EMT41	
	002202 071236		EHT1	
	002204 071362		EDT1	
	002206 071452		EFT1	
101				
102		:ERROR 42		UNSAFE ERROR DURING PACK ACKNOWLEDGE
103				
	002210 065516		EMT42	
	002212 071236		EHT1	

	002214	071362	EDT1	
	002216	071452	EFT1	
104				
105				:ERROR 43 "OPI" ERROR DURING PACK ACKNOWLEDGE
106				
	002220	065530	EMT43	
	002222	071236	EHT1	
	002224	071362	EDT1	
	002226	071452	EFT1	
107				
108				:ERROR 44 "RMR" ERROR DURING PACK ACKNOWLEDGE
109				
	002230	065540	EMT44	
	002232	071236	EHT1	
	002234	071362	EDT1	
	002236	071452	EFT1	
110				
111				:ERROR 45 "ILR" ERROR DURING PACK ACKNOWLEDGE
112				
	002240	065550	EMT45	
	002242	071236	EHT1	
	002244	071362	EDT1	
	002246	071452	EFT1	
113				
114				:ERROR 46 "ILF" ERROR DURING PACK ACKNOWLEDGE
115				
	002250	065560	EMT46	
	002252	071236	EHT1	
	002254	071362	EDT1	
	002256	071452	EFT1	
116				
117				:ERROR 47 COMPOSITE ERROR STATUS IS INCORRECT
118				
	002260	065570	EMT47	
	002262	071236	EHT1	
	002264	071362	EDT1	
	002266	071452	EFT1	
119				
120				:ERROR 50 PARITY ERROR WRITING REMOTE REGISTERS
121				
	002270	065576	EMT50	
	002272	071236	EHT1	
	002274	071362	EDT1	
	002276	071452	EFT1	
122				
123				:ERROR 51 INCORRECT IAE STATUS DURING SEEK COMMAND
124				
	002300	065606	EMT51	
	002302	071236	EHT1	
	002304	071362	EDT1	

	002306	071452	EFT1	
125				
126				
127				
	002310	065620	EMT52	
	002312	071236	EHT1	
	002314	071362	EDT1	
	002316	071452	EFT1	
128				
129				
130				
131				
	002320	065636	EMT53	
	002322	071236	EHT1	
	002324	071362	EDT1	
	002326	071452	EFT1	
132				
133				
134				
	002330	065654	EMT54	
	002332	071236	EHT1	
	002334	071362	EDT1	
	002336	071452	EFT1	
135				
136				
137				
	002340	065664	EMT55	
	002342	071236	EHT1	
	002344	071362	EDT1	
	002346	071452	EFT1	
138				
139				
140				
	002350	065676	EMT56	
	002352	071236	EHT1	
	002354	071362	EDT1	
	002356	071452	EFT1	
141				
142				
143				
	002360	065714	EMT57	
	002362	071236	EHT1	
	002364	071362	EDT1	
	002366	071452	EFT1	
144				
145				
146				
147				
	002370	065724	EMT60	
	002372	071236	EHT1	

Line	Address	Code	Label	Description
	002374	071362	EDT1	
	002376	071452	EFT1	
148				
149			:ERROR 61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
150			:	VOLUME VALID IS STIL SET
151				
	002400	065742	EMT61	
	002402	071236	EHT1	
	002404	071362	EDT1	
	002406	071452	EFT1	
152				
153			:ERROR 62	MOL IS ZERO, BUT OPI WAS NOT
154			:	REPORTED DURING SEEK COMMAND
155				
	002410	065762	EMT62	
	002412	071236	EHT1	
	002414	071362	EDT1	
	002416	071452	EFT1	
156				
157			:ERROR 63	UNUSED
158				
	002420	000000	0	
	002422	000000	0	
	002424	000000	0	
	002426	000000	0	
159				
160			:ERROR 64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
161				
	002430	066000	EMT64	
	002432	071236	EHT1	
	002434	071362	EDT1	
	002436	071452	EFT1	
162				
163			:ERROR 65	DRIVE EXECUTED A SEEK WITH ERROR SET
164				
	002440	066020	EMT65	
	002442	071236	EHT1	
	002444	071362	EDT1	
	002446	071452	EFT1	
165				
166			:ERROR 66	UNEXPECTED ERROR SET IN RMER1
167				
	002450	066040	EMT66	
	002452	071236	EHT1	
	002454	071362	EDT1	
	002456	071452	EFT1	
168				
169			:ERROR 67	UNEXPECTED ERROR SET IN RMER2
170				
	002460	066052	EMT67	

ERROR POINTER TABLE			
	002462 071236		EHT1
	002464 071362		EDT1
	002466 071452		EFT1
171			
172		;ERROR 70	ERRONEOUS "IAE" ERROR DURING RECALIBRATE
173			
	002470 066064		EMT70
	002472 071236		EHT1
	002474 071362		EDT1
	002476 071452		EFT1
174			
175		;ERROR 71	"ILF" ERROR DURING RECALIBRATE
176			
	002500 066074		EMT71
	002502 071236		EHT1
	002504 071362		EDT1
	002506 071452		EFT1
177			
178		;ERROR 72	"OPI" ERROR DURING RECALIBRATE DUE TO "MOL" = 0
179			
	002510 066104		EMT72
	002512 071236		EHT1
	002514 071362		EDT1
	002516 071452		EFT1
180			
181		;ERROR 73	"OPI" ERROR DURING RECALIBRATE BECAUSE ON
182		;	CYLINDER DIDNT DROP
183			
	002520 066122		EMT73
	002522 071236		EHT1
	002524 071362		EDT1
	002526 071452		EFT1
184			
185		;ERROR 74	"IVC" ERROR DURING RECALIBRATE - "VV" = 0
186			
	002530 066140		EMT74
	002532 071236		EHT1
	002534 071362		EDT1
	002536 071452		EFT1
187			
188		;ERROR 75	ERRONEOUS "IVC" ERROR DURING RECALIBRATE - "VV" = 1
189			
	002540 066150		EMT75
	002542 071236		EHT1
	002544 071362		EDT1
	002546 071452		EFT1
190			
191		;ERROR 76	"SKI" ERROR DURING RECALIBRATE
192			
	002550 066170		EMT76

ERROR POINTER TABLE			
	002552 071236		EHT1
	002554 071362		EDT1
	002556 071452		EFT1
193			
194		:ERROR 77	"DVC" OCCURRED DURING RECALIBRATE
195			
	002560 066200		EMT77
	002562 071236		EHT1
	002564 071362		EDT1
	002566 071452		EFT1
196			
197		:ERROR 100	LOST "MOL" DURING RECALIBRATE - "OPI" = 0
198			
	002570 066212		EMT100
	002572 071236		EHT1
	002574 071362		EDT1
	002576 071452		EFT1
199			
200		:ERROR 101	LOST "VV" DURING RECALIBRATE - "IVC" = 0
201			
	002600 066230		EMT101
	002602 071236		EHT1
	002604 071362		EDT1
	002606 071452		EFT1
202			
203		:ERROR 102	"ATA" DID NOT SET DURING RECALIBRATE
204			
	002610 066246		EMT102
	002612 071236		EHT1
	002614 071362		EDT1
	002616 071452		EFT1
205			
206		:ERROR 103	"OM" DID NOT RESET DURING RECALIBRATE
207			
	002620 066256		EMT103
	002622 071236		EHT1
	002624 071362		EDT1
	002626 071452		EFT1
208			
209		:ERROR 104	"PIP" IS STIL SET AFTER RECALIBRATE
210			
	002630 066270		EMT104
	002632 071236		EHT1
	002634 071362		EDT1
	002636 071452		EFT1
211			
212		:ERROR 105	UNEXPECTED "ILR" ERROR DURING RECALIBRATE
213			
	002640 066306		EMT105
	002642 071236		EHT1

ERROR POINTER TABLE				
	002644	071362		EDT1
	002646	071452		EFT1
214				
215			:ERROR 106	UNEXPECTED 'RMR' ERROR DURING RECALIBRATE
216				
	002650	066316		EMT106
	002652	071236		EHT1
	002654	071362		EDT1
	002656	071452		EFT1
217				
218			:ERROR 107	'UNS' ERROR DURING RECALIBRATE - AC POWER IS LOW
219				
	002660	066326		EMT107
	002662	071236		EHT1
	002664	071362		EDT1
	002666	071452		EFT1
220				
221			:ERROR 110	CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
222				
	002670	066346		EMT110
	002672	071260		EHT110
	002674	071400		EDT110
	002676	071470		EFT110
223				
224			:ERROR 111	NONEXISTENT DEVICE
225				
	002700	066360		EMT111
	002702	071264		EHT111
	002704	071402		EDT111
	002706	071472		EFT111
226				
227			:ERROR 112	DEVICE NOT AVAILABLE
228				
	002710	066366		EMT112
	002712	071264		EHT111
	002714	071402		EDT111
	002716	071472		EFT111
229				
230			:ERROR 113	BUS TIMEOUT-NED STATUS FAILURE
231				
	002720	066374		EMT113
	002722	000000		0
	002724	000000		0
	002726	000000		0
232				
233			:ERROR 114	DEVICE NOT AN RM05/3/2
234				
	002730	066410		EMT114
	002732	071270		EHT114
	002734	071404		EDT114

Line	Address	Code	Message
	002736 071474	EFT114	
235			
236		:ERROR 115	RMCS1 NOT INITIALIZED BY UNIBUS
237	002740 066416	EMT115	
	002742 071236	EHT1	
	002744 071362	EDT1	
	002746 071452	EFT1	
238			
239		:ERROR 116	RMBA NOT INITIALIZED BY UNIBUS
240	002750 066426	EMT116	
	002752 071236	EHT1	
	002754 071362	EDT1	
	002756 071452	EFT1	
241			
242		:ERROR 117	RMCS2 NOT INITIALIZED BY UNIBUS
243	002760 066436	EMT117	
	002762 071236	EHT1	
	002764 071362	EDT1	
	002766 071452	EFT1	
244			
245		:ERROR 120	RMER1 NOT INITIALIZED BY UNIBUS
246	002770 066446	EMT120	
	002772 071236	EHT1	
	002774 071362	EDT1	
	002776 071452	EFT1	
247			
248		:ERROR 121	RMAS NOT INITIALIZED BY UNIBUS
249	003000 066456	EMT121	
	003002 071236	EHT1	
	003004 071362	EDT1	
	003006 071452	EFT1	
250			
251		:ERROR 122	RMMR1 NOT INITIALIZED BY UNIBUS
252	003010 066466	EMT122	
	003012 071236	EHT1	
	003014 071362	EDT1	
	003016 071452	EFT1	
253			
254		:ERROR 123	RMDS NOT INITIALIZED BY UNIBUS
255	003020 066476	EMT123	
	003022 071236	EHT1	
	003024 071362	EDT1	
	003026 071452	EFT1	

256			
257			
258		:ERROR 124	RMEC2 NOT INITIALIZED BY UNIBUS
	003030	066506	EMT124
	003032	071236	EHT1
	003034	071362	EDT1
	003036	071452	EFT1
259			
260			
261		:ERROR 125	RMMR2 NOT INITIALIZED BY UNIBUS
	003040	066516	EMT125
	003042	071236	EHT1
	003044	071362	EDT1
	003046	071452	EFT1
262			
263			
264		:ERROR 126	RMCS1 NOT CLEARED BY CONTROLLER CLEAR
	003050	066526	EMT126
	003052	071236	EHT1
	003054	071362	EDT1
	003056	071452	EFT1
265			
266			
267		:ERROR 127	RMBA NOT CLEARED BY CONTROLLER CLEAR
	003060	066540	EMT127
	003062	071236	EHT1
	003064	071362	EDT1
	003066	071452	EFT1
268			
269			
270		:ERROR 130	RMCS2 NOT CLEARED BY CONTROLLER CLEAR
	003070	066552	EMT130
	003072	071236	EHT1
	003074	071362	EDT1
	003076	071452	EFT1
271			
272			
273		:ERROR 131	RMER1 NOT CLEARED BY CONTROLLER CLEAR
	003100	066564	EMT131
	003102	071236	EHT1
	003104	071362	EDT1
	003106	071452	EFT1
274			
275			
276		:ERROR 132	RMAS NOT CLEARED BY CONTROLLER CLEAR
	003110	066576	EMT132
	003112	071236	EHT1
	003114	071362	EDT1
	003116	071452	EFT1

277			
278			
279		;ERROR 133	RMMR1 NOT CLEARED BY CONTROLLER CLEAR
	003120	066610	EMT133
	003122	071236	EHT1
	003124	071362	EDT1
	003126	071452	EFT1
280			
281			
282		;ERROR 134	RMDS NOT CLEARED BY CONTROLLER CLEAR
	003130	066622	EMT134
	003132	071236	EHT1
	003134	071362	EDT1
	003136	071452	EFT1
283			
284			
285		;ERROR 135	RMEC2 NOT CLEARED BY CONTROLLER CLEAR
	003140	066634	EMT135
	003142	071236	EHT1
	003144	071362	EDT1
	003146	071452	EFT1
286			
287			
288		;ERROR 136	RMMR2 NOT CLEARED BY CONTROLLER CLEAR
	003150	066646	EMT136
	003152	071236	EHT1
	003154	071362	EDT1
	003156	071452	EFT1
289			
290			
291		;ERROR 137	RMCS1 NOT CLEARED BY ERROR CLEAR
	003160	066660	EMT137
	003162	071236	EHT1
	003164	071362	EDT1
	003166	071452	EFT1
292			
293			
294		;ERROR 140	RMCS2 NOT CLEARED BY ERROR CLEAR
	003170	066670	EMT140
	003172	071236	EHT1
	003174	071362	EDT1
	003176	071452	EFT1
295			
296			
297		;ERROR 141	RMCS1 NOT CLEARED BY DRIVE CLEAR
	003200	066700	EMT141
	003202	071236	EHT1
	003204	071362	EDT1
	003206	071452	EFT1
298			

299				
300			:ERROR 142	RMDS NOT CLEARED BY DRIVE CLEAR
	003210	066710		EMT142
	003212	071236		EHT1
	003214	071362		EDT1
	003216	071452		EFT1
301				
302			:ERROR 143	RMER1 NOT CLEARED BY DRIVE CLEAR
303				
	003220	066720		EMT143
	003222	071236		EHT1
	003224	071362		EDT1
	003226	071452		EFT1
304				
305			:ERROR 144	RMAS NOT CLEARED BY DRIVE CLEAR
306				
	003230	066730		EMT144
	003232	071236		EHT1
	003234	071362		EDT1
	003236	071452		EFT1
307				
308			:ERROR 145	RMMR1 NOT CLEARED BY DRIVE CLEAR
309				
	003240	066740		EMT145
	003242	071236		EHT1
	003244	071362		EDT1
	003246	071452		EFT1
310				
311			:ERROR 146	RMMR2 NOT CLEARED BY DRIVE CLEAR
312				
	003250	066750		EMT146
	003252	071236		EHT1
	003254	071362		EDT1
	003256	071452		EFT1
313				
314			:ERROR 147	RMER2 NOT CLEARED BY DRIVE CLEAR
315				
	003260	066760		EMT147
	003262	071236		EHT1
	003264	071362		EDT1
	003266	071452		EFT1
316				
317			:ERROR 150	RMEC2 NOT CLEARED BY DRIVE CLEAR
318				
	003270	066770		EMT150
	003272	071236		EHT1
	003274	071362		EDT1
	003276	071452		EFT1
319				
320			:ERROR 151	MEDIUM NOT ON LINE

321	003300 067000	EMT151	
	003302 071236	EHT1	
	003304 071362	EDT1	
	003306 071452	EFT1	
322			
323		;ERROR 152	DRIVE FAULT
324	003310 067012	EMT152	
	003312 071236	EHT1	
	003314 071362	EDT1	
	003316 071452	EFT1	
325			
326		;ERROR 153	UNSAFE SHOULD BE SET BECAUSE DVC IS SET
327	003320 067024	EMT153	
	003322 071236	EHT1	
	003324 071362	EDT1	
	003326 071452	EFT1	
328			
329		;ERROR 154	UNSAFE SHOULD NOT BE SET, AC IS LOW
330	003330 067042	EMT154	
	003332 071236	EHT1	
	003334 071362	EDT1	
	003336 071452	EFT1	
331			
332		;ERROR 155	VOLUME VALID NOT SET BY PACK ACK
333	003340 067060	EMT155	
	003342 071236	EHT1	
	003344 071362	EDT1	
	003346 071452	EFT1	
334			
335		;ERROR 156	OFFSET MODE NOT SET BY OFFSET COMMAND
336	003350 067072	EMT156	
	003352 071236	EHT1	
	003354 071362	EDT1	
	003356 071452	EFT1	
337			
338		;ERROR 157	OFFSET MODE NOT RESET BY RTC COMMAND
339	003360 067104	EMT157	
	003362 071236	EHT1	
	003364 071362	EDT1	
	003366 071452	EFT1	
340			
341		;ERROR 160	RMOF NOT RESET BY RIP COMMAND
342			

003370	067116	EMT160
003372	071236	EHT1
003374	071362	EDT1
003376	071452	EFT1
343		
344		
345		:ERROR 161 RMDA NOT RESET BY RIP COMMAND
003400	067126	EMT161
003402	071236	EHT1
003404	071362	EDT1
003406	071452	EFT1
346		
347		
348		:ERROR 162 RMDC NOT RESET BY RIP COMMAND
003410	067140	EMT162
003412	071236	EHT1
003414	071362	EDT1
003416	071452	EFT1
349		
350		
351		:ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
352		: WRITE BUFFER
003420	071032	EMT336
003422	071320	EHT336
003424	071416	EDT336
003426	071506	EFT336
353		
354		
355		:ERROR 164 OPI SHOULD NOT BE SET
003430	067162	EMT164
003432	071236	EHT1
003434	071362	EDT1
003436	071452	EFT1
356		
357		
358		:ERROR 165 IVC SHOULD NOT BE SET
003440	067170	EMT165
003442	071236	EHT1
003444	071362	EDT1
003446	071452	EFT1
359		
360		
361		:ERROR 166 IAE SHOULD NOT BE SET
003450	067176	EMT166
003452	071236	EHT1
003454	071362	EDT1
003456	071452	EFT1
362		
363		
364		:ERROR 167 NEM SHOULD NOT BE SET

	003460	067204		EMT167
	003462	071236		EHT1
	003464	071362		EDT1
	003466	071452		EFT1
365				
366			:ERROR 170	UNUSED
367				
	003470	000000		0
	003472	000000		0
	003474	000000		0
	003476	000000		0
368				
369			:ERROR 171	"ATA" NOT SET DURING RETURN TO CENTERLINE
370				
	003500	067222		EMT171
	003502	071236		EHT1
	003504	071362		EDT1
	003506	071452		EFT1
371				
372			:ERROR 172	"ATA" NOT SET BY OFFSET COMMAND
373				
	003510	067232		EMT172
	003512	071236		EHT1
	003514	071362		EDT1
	003516	071452		EFT1
374				
375			:ERROR 173	RMER2 NOT INITIALIZED BY UNIBUS INIT
376				
	003520	067242		EMT173
	003522	071236		EHT1
	003524	071362		EDT1
	003526	071452		EFT1
377				
378			:ERROR 174	RMER2 NOT INITIALIZED BY CONTROLLER CLEAR
379				
	003530	067252		EMT174
	003532	071236		EHT1
	003534	071362		EDT1
	003536	071452		EFT1
380				
381			:ERROR 175	SELECTED DEVICE IS IN WRITE PROTECT
382				
	003540	067264		EMT175
	003542	071236		EHT1
	003544	071362		EDT1
	003546	071452		EFT1
383				
384			:ERROR 176	CANNOT SET DIAGNOSTIC MODE
385				
	003550	067272		EMT176

003552	071236	EMT1	
003554	071362	EDT1	
003556	071452	EFT1	
386			
387			
388		:ERROR	177 INCORRECT "MOL" STATUS DURING DIAGNOSTIC MODE
003560	067300	EMT177	
003562	071236	EMT1	
003564	071362	EDT1	
003566	071452	EFT1	
389			
390		:ERROR	200 INCORRECT "PIP" STATUS DURING DIAGNOSTIC MODE
391			
003570	067312	EMT200	
003572	071236	EMT1	
003574	071362	EDT1	
003576	071452	EFT1	
392			
393		:ERROR	201 INCORRECT "WRL" STATUS DURING DIAGNOSTIC MODE
394			
003600	067324	EMT201	
003602	071236	EMT1	
003604	071362	EDT1	
003606	071452	EFT1	
395			
396		:ERROR	202 INCORRECT "SKI" STATUS DURING DIAGNOSTIC MODE
397			
003610	067336	EMT202	
003612	071236	EMT1	
003614	071362	EDT1	
003616	071452	EFT1	
398			
399		:ERROR	203 INCORRECT "DVC" STATUS DURING DIAGNOSTIC MODE
400			
003620	067350	EMT203	
003622	071236	EMT1	
003624	071362	EDT1	
003626	071452	EFT1	
401			
402		:ERROR	204 "VV" WAS NOT RESET BY MAINTENANCE UNIT READY
403			
003630	067362	EMT204	
003632	071236	EMT1	
003634	071362	EDT1	
003636	071452	EFT1	
404			
405		:ERROR	205 SELECTED DEVICE HAS A PERSISTENT "SKI" ERROR
406			
003640	067400	EMT205	
003642	071236	EMT1	

	003644 071362		EDT1	
	003646 071452		EFT1	
407				
408		:ERROR 206		'LBC' DID NOT SET DURING DIAGNOSTIC MODE
409				
	003650 067410		EMT206	
	003652 071236		EHT1	
	003654 071362		EDT1	
	003656 071452		EFT1	
410				
411		:ERROR 207		UNEXPECTED LOSS OF 'MOL' - MEDIUM IS OFF LINE
412				
	003660 067420		EMT207	
	003662 071236		EHT1	
	003664 071362		EDT1	
	003666 071452		EFT1	
413				
414		:ERROR 210		UNEXPECTED LOSS OF VOLUME VALID - 'VV' = 0
415				
	003670 067432		EMT210	
	003672 071236		EHT1	
	003674 071362		EDT1	
	003676 071452		EFT1	
416				
417		:ERROR 211		UNEXPECTED MECHANICAL MOTION - 'PIP' = 1
418				
	003700 067440		EMT211	
	003702 071236		EHT1	
	003704 071362		EDT1	
	003706 071452		EFT1	
419				
420		:ERROR 212		UNEXPECTED DEVICE FAULT - 'DVC' = 1
421				
	003710 067454		EMT212	
	003712 071236		EHT1	
	003714 071362		EDT1	
	003716 071452		EFT1	
422				
423		:ERROR 213		UNEXPECTED SEEK INCOMPLETE ERROR - 'SKI' = 1
424				
	003720 067470		EMT213	
	003722 071236		EHT1	
	003724 071362		EDT1	
	003726 071452		EFT1	
425				
426		:ERROR 214		DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
427				
	003730 067500		EMT214	
	003732 071250		EHT2	
	003734 071372		EDT2	

003736	071462	EFT2	
428			
429		:ERROR 215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
430			
003740	067520	EMT215	
003742	071250	EHT2	
003744	071372	EDT2	
003746	071462	EFT2	
431			
432		:ERROR 216	INCORRECT "IVC" STATUS
433			
003750	067532	EMT216	
003752	071236	EHT1	
003754	071362	EDT1	
003756	071452	EFT1	
434			
435		:ERROR 217	INCORRECT "IAE" STATUS
436			
003760	067542	EMT217	
003762	071236	EHT1	
003764	071362	EDT1	
003766	071452	EFT1	
437			
438		:ERROR 220	INCORRECT "WLE" STATUS
439			
003770	067552	EMT220	
003772	071236	EHT1	
003774	071362	EDT1	
003776	071452	EFT1	
440			
441		:ERROR 221	INCORRECT "OPI" STATUS
442			
004000	067562	EMT221	
004002	071236	EHT1	
004004	071362	EDT1	
004006	071452	EFT1	
443			
444		:ERROR 222	RM DID NOT DETECT RMR ERROR
445			
004010	067572	EMT222	
004012	071236	EHT1	
004014	071362	EDT1	
004016	071452	EFT1	
446			
447		:ERROR 223	RM DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
448			
004020	067602	EMT223	
004022	071274	EHT223	
004024	071406	EDT223	
004026	071476	EFT223	

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

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

491

Line	Address	Value	Error Code	Description
492			:ERROR 242	UNUSED
493	004210	000000	0	
	004212	000000	0	
	004214	000000	0	
	004216	000000	0	
494			:ERROR 243	UNUSED
495				
496	004220	000000	0	
	004222	000000	0	
	004224	000000	0	
	004226	000000	0	
497			:ERROR 244	UNUSED
498				
499	004230	000000	0	
	004232	000000	0	
	004234	000000	0	
	004236	000000	0	
500			:ERROR 245	UNUSED
501				
502	004240	000000	0	
	004242	000000	0	
	004244	000000	0	
	004246	000000	0	
503			:ERROR 246	"ATA" NOT RESET BY GO WHEN "ERR" = 0
504				
505	004250	067656	EMT246	
	004252	071236	EHT1	
	004254	071362	EDT1	
	004256	071452	EFT1	
506			:ERROR 247	"ATA" NOT RESET BY WRITING RMAS
507				
508	004260	067666	EMT247	
	004262	071236	EHT1	
	004264	071362	EDT1	
	004266	071452	EFT1	
509			:ERROR 250	"ATA" WAS RESET BY GO WHEN "ERR" = 1
510				
511	004270	067700	EMT250	
	004272	071236	EHT1	
	004274	071362	EDT1	
	004276	071452	EFT1	
512			:ERROR 251	PROGRAM INTERRUPT WAS NOT GENERATED
513				

514	004300 067714	EMT251	
	004302 071250	EHT2	
	004304 071372	EDT2	
	004306 071462	EFT2	
515			
516		;ERROR 252	PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
517			
	004310 067722	EMT252	
	004312 071250	EHT2	
	004314 071372	EDT2	
	004316 071462	EFT2	
518			
519		;ERROR 253	OFFSET MODE WAS NOT RESET BY WRITING RMDC
520			
	004320 067730	EMT253	
	004322 071236	EHT1	
	004324 071362	EDT1	
	004326 071452	EFT1	
521			
522		;ERROR 254	INCORRECT "ILF" STATUS
523			
	004330 067746	EMT254	
	004332 071236	EHT1	
	004334 071362	EDT1	
	004336 071452	EFT1	
524			
525		;ERROR 255	INCORRECT "ATA" STATUS
526			
	004340 067756	EMT255	
	004342 071236	EHT1	
	004344 071362	EDT1	
	004346 071452	EFT1	
527			
528		;ERROR 256	INCORRECT "ILR" STATUS
529			
	004350 067766	EMT256	
	004352 071306	EHT256	
	004354 071406	EDT223	
	004356 071476	EFT223	
530			
531		;ERROR 257	INVALID IAE STATUS DURING SEARCH COMMAND
532			
	004360 067776	EMT257	
	004362 071236	EHT1	
	004364 071362	EDT1	
	004366 071452	EFT1	
533			
534		;ERROR 260	"IVC" WAS NOT DETECTED DURING SEARCH COMMAND
535			

004370	070010	EMT260
004372	071236	EHT1
004374	071362	EDT1
004376	071452	EFT1
536		
537		
538		:ERROR 261 DRIVE EXECUTED SEARCH WITH ERROR SET
004400	070022	EMT261
004402	071236	EHT1
004404	071362	EDT1
004406	071452	EFT1
539		
540		
541		:ERROR 262 "LBC" ERROR NOT SET DURING DIAGNOSTIC MODE
004410	070042	EMT262
004412	071236	EHT1
004414	071362	EDT1
004416	071452	EFT1
542		
543		
544		:ERROR 263 "SKI" ERROR DURING SEARCH COMMAND
004420	070052	EMT263
004422	071236	EHT1
004424	071362	EDT1
004426	071452	EFT1
545		
546		
547		:ERROR 264 "IVC" ERROR DURING SEARCH - LOST VOLUME VALID
004430	070062	EMT264
004432	071236	EHT1
004434	071362	EDT1
004436	071452	EFT1
548		
549		
550		:ERROR 265 ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
004440	070102	EMT265
004442	071236	EHT1
004444	071362	EDT1
004446	071452	EFT1
551		
552		
553		:ERROR 266 DEVICE FAULT (DVC) DURING SEARCH
004450	070122	EMT266
004452	071236	EHT1
004454	071362	EDT1
004456	071452	EFT1
554		
555		
556		:ERROR 267 SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER
557		: ADDRESS IS TOO LARGE

004460	070134	EMT267
004462	071236	EHT1
004464	071362	EDT1
004466	071452	EFT1
558		
559		
560		:ERROR 270 OPI ERROR DURING SEARCH BECAUSE MOL = 0
004470	070152	EMT270
004472	071236	EHT1
004474	071362	EDT1
004476	071452	EFT1
561		
562		:ERROR 271 OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
563		: DIDN'T DROP
564		
004500	070166	EMT271
004502	071236	EHT1
004504	071362	EDT1
004506	071452	EFT1
565		
566		:ERROR 272 LOST MOL DURING SEARCH, OPI IS NOT SET
567		
004510	070204	EMT272
004512	071236	EHT1
004514	071362	EDT1
004516	071452	EFT1
568		
569		:ERROR 273 PIP STIL SET AFTER SEARCH
570		
004520	070222	EMT273
004522	071236	EHT1
004524	071362	EDT1
004526	071452	EFT1
571		
572		:ERROR 274 PARITY ERROR OCCURRED WHILE WRITING REMOTE
573		: REGISTERS BUT MXF DID NOT SET
574		
004530	070240	EMT274
004532	071236	EHT1
004534	071362	EDT1
004536	071452	EFT1
575		
576		:ERROR 275 MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
577		: COMMAND STARTED
578		
004540	070256	EMT275
004542	071236	EHT1
004544	071362	EDT1
004546	071452	EFT1
579		

580		:ERROR 276	'OPI' ERROR DURING DATA TRANSFER BECAUSE 'MOL' WAS
581		:	ZERO
582			
	004550 070270	EMT276	
	004552 071236	EHT1	
	004554 071362	EDT1	
	004556 071452	EFT1	
583			
584		:ERROR 277	'OPI' ERROR DURING DATA TRANSFER BECAUSE 1) ON
585		:	CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
586		:	3) RUN TIMED OUT
587			
	004560 070304	EMT277	
	004562 071236	EHT1	
	004564 071362	EDT1	
	004566 071452	EFT1	
588			
589		:ERROR 300	'IVC' ERROR DURING DATA TRANSFER BECAUSE VOLUME
590		:	WAS NOT VALID
591			
	004570 070322	EMT300	
	004572 071236	EHT1	
	004574 071362	EDT1	
	004576 071452	EFT1	
592			
593		:ERROR 301	ERRONEOUS 'IVC' ERROR DURING DATA TRANSFER - VOLUME
594		:	IS VALID
595			
	004600 070342	EMT301	
	004602 071236	EHT1	
	004604 071362	EDT1	
	004606 071452	EFT1	
596			
597		:ERROR 302	'ILR' ERROR DURING DATA TRANSFER
598			
	004610 070364	EMT302	
	004612 071236	EHT1	
	004614 071362	EDT1	
	004616 071452	EFT1	
599			
600		:ERROR 303	'ILF' ERROR DURING DATA TRANSFER
601			
	004620 070376	EMT303	
	004622 071236	EHT1	
	004624 071362	EDT1	
	004626 071452	EFT1	
602			
603		:ERROR 304	'RMR' ERROR DURING DATA TRANSFER
604			
	004630 070410	EMT304	
	004632 071236	EHT1	

ERROR POINTER TABLE			
	004634 071362		EDT1
	004636 071452		EFT1
605			
606		:ERROR 305	INCORRECT "IAE" STATUS DURING DATA TRANSFER
607			
	004640 070422		EMT305
	004642 071236		EHT1
	004644 071362		EDT1
	004646 071452		EFT1
608			
609		:ERROR 306	"SKI" ERROR DURING DATA TRANSFER
610			
	004650 070434		EMT306
	004652 071236		EHT1
	004654 071362		EDT1
	004656 071452		EFT1
611			
612		:ERROR 307	DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
613			
	004660 070444		EMT307
	004662 071236		EHT1
	004664 071362		EDT1
	004666 071452		EFT1
614			
615		:ERROR 310	DEVICE FAULT DURING DATA TRANSFER
616			
	004670 070464		EMT310
	004672 071236		EHT1
	004674 071362		EDT1
	004676 071452		EFT1
617			
618		:ERROR 311	LOSS OF BIT CLOCK DURING DATA TRANSFER
619			
	004700 070476		EMT311
	004702 071236		EHT1
	004704 071362		EDT1
	004706 071452		EFT1
620			
621		:ERROR 312	LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
622			
	004710 070510		EMT312
	004712 071236		EHT1
	004714 071362		EDT1
	004716 071452		EFT1
623			
624		:ERROR 313	UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
625			
	004720 070522		EMT313
	004722 071236		EHT1
	004724 071362		EDT1

Line	Code	Module	Code	Module	Description
	004726	071452		EFT1	
626					
627			:ERROR	314	DRIVE TIMING ERROR DURING DATA TRANSFER
628					
	004730	070542		EMT314	
	004732	071236		EHT1	
	004734	071362		EDT1	
	004736	071452		EFT1	
629					
630			:ERROR	315	WRITE LOCK ERROR
631					
	004740	070554		EMT315	
	004742	071236		EHT1	
	004744	071362		EDT1	
	004746	071452		EFT1	
632					
633			:ERROR	316	ERRONEOUS WRITE LOCK ERROR
634					
	004750	070566		EMT316	
	004752	071236		EHT1	
	004754	071362		EDT1	
	004756	071452		EFT1	
635					
636			:ERROR	317	HEADER CRC ERROR DURING DATA TRANSFER
637					
	004760	070600		EMT317	
	004762	071236		EHT1	
	004764	071362		EDT1	
	004766	071452		EFT1	
638					
639			:ERROR	320	FORMAT ERROR DURING DATA TRANSFER
640					
	004770	070610		EMT320	
	004772	071236		EHT1	
	004774	071362		EDT1	
	004776	071452		EFT1	
641					
642			:ERROR	321	HEADER COMPARE ERROR DURING DATA TRANSFER
643					
	005000	070620		EMT321	
	005002	071236		EHT1	
	005004	071362		EDT1	
	005006	071452		EFT1	
644					
645			:ERROR	322	HEADER ERRORS SHOULD NOT BE SET
646					
	005010	070630		EMT322	
	005012	071236		EHT1	
	005014	071362		EDT1	
	005016	071452		EFT1	

647			
648		:ERROR 323	DATA CHECK ERROR DURING DATA TRANSFER
649	005020 070636	EMT323	
	005022 071236	EHT1	
	005024 071362	EDT1	
	005026 071452	EFT1	
650			
651		:ERROR 324	CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
652	005030 070646	EMT324	
	005032 071236	EHT1	
	005034 071362	EDT1	
	005036 071452	EFT1	
653			
654		:ERROR 325	UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
655	005040 070660	EMT325	
	005042 071236	EHT1	
	005044 071362	EDT1	
	005046 071452	EFT1	
656			
657		:ERROR 326	DATA PARITY ERROR DURING READ COMMAND
658	005050 070672	EMT326	
	005052 071236	EHT1	
	005054 071362	EDT1	
	005056 071452	EFT1	
659			
660		:ERROR 327	OFFSET MODE NOT RESET BY WRITE COMMAND
661	005060 070710	EMT327	
	005062 071236	EHT1	
	005064 071362	EDT1	
	005066 071452	EFT1	
662			
663		:ERROR 330	DATA PARITY ERROR DURING WRITE COMMAND
664	005070 070722	EMT330	
	005072 071236	EHT1	
	005074 071362	EDT1	
	005076 071452	EFT1	
665			
666		:ERROR 331	WRITE CLOCK FAILURE DURING WRITE COMMAND
667	005100 070732	EMT331	
	005102 071236	EHT1	
	005104 071362	EDT1	
	005106 071452	EFT1	

668			
669		:ERROR 332	DATA LATE ERROR DURING DATA TRANSFER
670			
	005110	070744	EMT332
	005112	071236	EHT1
	005114	071362	EDT1
	005116	071452	EFT1
671			
672		:ERROR 333	PIP STIL SET AFTER DATA TRANSFER - SKI = 0
673			
	005120	070756	EMT333
	005122	071236	EHT1
	005124	071362	EDT1
	005126	071452	EFT1
674			
675		:ERROR 334	LOST MOL DURING DATA TRANSFER - OPI = 0
676			
	005130	070774	EMT334
	005132	071236	EHT1
	005134	071362	EDT1
	005136	071452	EFT1
677			
678		:ERROR 335	LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
679			
	005140	071012	EMT335
	005142	071236	EHT1
	005144	071362	EDT1
	005146	071452	EFT1
680			
681		:ERROR 336	DATA READ DOES NOT COMPARE WITH DATA WRITTEN
682			
	005150	071032	EMT336
	005152	071320	EHT336
	005154	071416	EDT336
	005156	071506	EFT336
683			
684		:ERROR 337	WRITE CHECK ERROR NOT DETECTED
685			
	005160	071042	EMT337
	005162	071332	EHT337
	005164	071426	EDT337
	005166	071516	EFT337
686			
687		:ERROR 340	WRITE CHECK ERROR AT UNEXPECTED ADDRESS
688			
	005170	071052	EMT340
	005172	071320	EHT336
	005174	071416	EDT336
	005176	071506	EFT336

689

690			:ERROR 341	INCORRECT DATA DURING WRITE CHECK ERROR
691	005200	071064		EMT341
	005202	071320		EHT336
	005204	071416		EDT336
	005206	071506		EFT336
692			:ERROR 342	"IVC" ERROR NOT DETECTED DURING DATA TRANSFER
693				
694	005210	071072		EMT342
	005212	071236		EHT1
	005214	071362		EDT1
	005216	071452		EFT1
695			:ERROR 343	"FER" NOT DETECTED DURING DATA TRANSFER
696				
697	005220	071104		EMT343
	005222	071236		EHT1
	005224	071362		EDT1
	005226	071452		EFT1
698			:ERROR 344	"HCE" NOT DETECTED DURING DATA TRANSFER
699				
700	005230	071116		EMT344
	005232	071344		EHT344
	005234	071436		EDT344
	005236	071526		EFT344
701			:ERROR 345	"BSE" NOT DETECTED DURING DATA TRANSFER
702				
703	005240	071130		EMT345
	005242	071236		EHT1
	005244	071362		EDT1
	005246	071452		EFT1
704			:ERROR 346	HEADER ERROR WAS DETECTED W/ HCI SET
705				
706	005250	071140		EMT346
	005252	071236		EHT1
	005254	071362		EDT1
	005256	071452		EFT1
707			:ERROR 347	DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
708				
709	005260	071154		EMT347
	005262	071236		EHT1
	005264	071362		EDT1
	005266	071452		EFT1
710			:ERROR 350	LOST VOLUME VALID DURING SEARCH - "IVC" = 0
711				

712
005270 071166 EMT350
005272 071236 EHT1
005274 071362 EDT1
005276 071452 EFT1

713
714 ;ERROR 351 "ATA" DID NOT SET DURING SEARCH
715
005300 071204 EMT351
005302 071236 EHT1
005304 071362 EDT1
005306 071452 EFT1

716
717 ;ERROR 352 PROGRAM TIMEOUT WHILE TESTING RMLA
718
005310 071214 EMT352
005312 000000 0
005314 000000 0
005316 000000 0

719
720 ;ERROR 353 LOOK AHEAD TEST FAILS
721
005320 071220 EMT353
005322 071356 EHT353
005324 071450 EDT353
005326 071536 EFT353

722
723 ;ERROR 354 BSE SHOULD NOT BE SET
724
005330 071230 EMT354
005332 071236 EHT1
005334 071362 EDT1
005336 071452 EFT1

725
726 ;PUT ERROR TABLE HERE
727

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

.SBTTL ERROR TABLE USAGE

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

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

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

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

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


```

1           ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 005340   011600           BADTMO: MOV     (SP),R0             ;SAVE PC WHERE THE TIME OUT OCCURED
4 005342   005740           TST      -(R0)             ;ADJUST PC -2
5 005344   022626           CMP     (SP)+,(SP)+       ;RESTORE STACK POINTER
6 005346   104401   005354   TYPE    ,65$             ;:TYPE ASCIZ STRING
   005352   000417           BR      64$             ;:GET OVER THE ASCIZ
   ;:65$: .ASCIZ  <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 005412   010046           MOV     R0,-(SP)        ;SETUP FOR TYPING OUT PC
8 005414   104402           TYPOC
9 005416   000240           NOP                    ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;TO STOP ON UNEXPECTED TIMEOUT.

10
11
12          .SBTTL  START OF PROGRAM
13
14 005420   000240           START:  NOP
15 005422   005227   000000   INC     #0             ;TTY LOOP, WAIT FOR INCREMENT
16 005426   001375           BNE    -4             ;OF WORD
17 005430   000005           RESET   ;RESET THE WORLD

18
19          .SBTTL  INITIALIZE THE COMMON TAGS
   ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
   MOV     #$CMTAG,R6    ;:FIRST LOCATION TO BE CLEARED
   CLR    (R6)+         ;:CLEAR MEMORY LOCATION
   CMP    #SWR,R6       ;:DONE?
   BNE   -6             ;:LOOP BACK IF NO
   MOV    #STACK,SP     ;:SETUP THE STACK POINTER

   ;:INITIALIZE A FEW VECTORS
   MOV    #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
   MOV    #340,@IOTVEC+2 ;:LEVEL 7
   MOV    #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
   MOV    #340,@EMTVEC+2 ;:LEVEL 7
   MOV    #STRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
   MOV    #340,@TRAPVEC+2 ;:LEVEL 7
   MOV    #SPWRDN,@PWRVEC ;:POWER FAILURE VECTOR
   MOV    #340,@PWRVEC+2 ;:LEVEL 7
   MOV    $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
   CLR    $TIMES        ;:INITIALIZE NUMBER OF ITERATIONS
   CLR    $ESCAPE       ;:CLEAR THE ESCAPE ON ERROR ADDRESS
   MOVB  #1,$ERMAX      ;:ALLOW ONE ERROR PER TEST
   MOV   #.,$LPADR      ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
   MOV   #.,$LPERR      ;:SETUP THE ERROR LOOP ADDRESS

   ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
   MOV   @ERRVEC, -(SP) ;:SAVE ERROR VECTOR
   MOV   #64$,@ERRVEC  ;:SET UP ERROR VECTOR
   MOV   #DSWR,SWR     ;:SETUP FOR A HARDWARE SWICH REGISTER
   MOV   #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
   CMP   #-1,@SWR      ;:TRY TO REFERENCE HARDWARE SWR
   BNE   66$           ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
   ;:AND THE HARDWARE SWR IS NOT = -1
   BR    65$           ;:BRANCH IF NO TIMEOUT
   64$:  MOV    #65$, (SP) ;:SET UP FOR TRAP RETURN
   65$:  MOV    #SWREG,SWR ;:POINT TO SOFTWARE SWR
   MOV    #DISPREG,DISPLAY

```

```

005654 012637 000004      66$: MOV      (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR
005660 005037 001230      CLR      $PASS          ;;CLEAR PASS COUNT
005664 132737 000200 001243  BITB     #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
005672 001403           BEQ      67$           ;;YES,USE NON-APT SWITCH
005674 012737 001244 001154  MOV      #$$SWREG,$SWR   ;;NO,USE APT SWITCH REGISTER
005702           67$:
20 005702 012737 005340 000004  :SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
21 005710 012737 000300 000006  MOV      #BADTMO,ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
22 005716 012746 000300           MOV      #PR6,ERRVEC+2  ;;LEVEL 6
23 005722 012746 005730           MOV      #PR6,-(SP)     ;;PUT NEW PS ON STACK
005726 000002           MOV      #68$,-(SP)    ;;PUT NEW PC ON STACK
005730           RTI           ;;POP NEW PC AND PS
24 68$:
25 .SBTTL  TYPE PROGRAM NAME
   ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005730 005227 177777      INC      #-1           ;;FIRST TIME?
005734 001056           BNE      69$           ;;BRANCH IF NO
005736 022737 032530 000042  CMP      #SENDAD,@#42   ;;ACT-11?
005744 001452           BEQ      69$           ;;BRANCH IF YES
005746 104401 006014      TYPE     ,70$         ;;TYPE ASCIZ STRING
   .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
005752 005737 000042      TST     @#42          ;;ARE WE RUNNING UNDER XXDP/ACT?
005756 001012           BNE      71$           ;;BRANCH IF YES
005760 123727 001242 000001  CMPB    $ENV,#1        ;;ARE WE RUNNING UNDER APT?
005766 001406           BEQ      71$           ;;BRANCH IF YES
005770 023727 001154 000176  CMP     $SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
005776 001005           BNE      72$           ;;BRANCH IF NO
006000 104407           GTSWR          ;;GET SOFT-SWR SETTINGS
006002 000403           BR       72$
006004 112737 000001 001150 71$:  MOVB    #1,$AUTOB     ;;SET AUTO-MODE INDICATOR
006012           72$:
006012 000427      BR       69$         ;;GET OVER THE ASCIZ
006072           ;;70$: .ASCIZ <CRLF>@CZRMNAO - RM05/3/2 FUNCTIONAL TEST, PART 2@<CRLF>
           69$:
26
27
28   ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
29   ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
30 006072 005037 001330      CLR     XXDP          ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
31 006076 122737 000016 000041  CMPB    #16,@#41      ;;LOADED FROM AN RM05/3/2 ?
32 006104 001160           BNE     3$           ;;BRANCH IF NOT
33 006106 013737 000040 001330  MOV     @#40,XXDP     ;;GET DEVICE INDICATOR AND NUMBER
34 006114 122737 000007 001330  CMPB    #7,XXDP      ;;IS IT A VALID NUMBER ?
35 006122 103002           BHS     1$           ;;YES
36 006124 105037 001330      CLRB   XXDP          ;;NO, DEFAULT TO DRIVE 0
37 006130 005737 000042      TST    @#42          ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
38 006134 001425           BEQ     2$           ;;BR IF NEITHER
39 006136 104401 006144      TYPE   ,74$         ;;TYPE ASCIZ STRING
           006142 000412      BR     73$         ;;GET OVER THE ASCIZ
           ;;74$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
           73$:
40 006170           CLR     -(SP)        ;;CLEAR WORD ON STACK
41 006172 005046 001330      MOVB   XXDP,(SP)    ;;GET DRIVE ADDRESS
42 006176 104403           TYPOS          ;;TYPE THE ADDRESS
43 006200 001           .BYTE  1          ;;ONLY 1 CHARACTER

```



```

95
96 006624 146137 064506 001300 3$: BICB ATNTBL(R1), $DEV ;CLEAR DEVICE FROM BIT MAP
97 006632 104401 001217 4$: TYPE , $CRLF ;CR-LF
98 006636 104401 064277 TYPE ,MSGDRV ;TYPE 'DRIVE'
99 006642 010146 MOV R1, -(SP) ;SAVE R1 FOR TYPEOUT
006644 104403 TYPOS ;GO TYPE--OCTAL ASCII
006646 002 .BYTE 2 ;TYPE 2 DIGIT(S)
006647 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
100 006650 104401 TYPE ;TYPE ERROR MESSAGE
101 006652 000000 5$: .WORD 0 ;ADDRESS OF MESSAGE GOES HERE
102
103 006654 005201 6$: INC R1 ;INCREMENT THE DRIVE ADDRESS
104 006656 020127 000007 CMP R1, #7 ;ALL DRIVES ARE CHECKED ?
105 006662 003706 BLE 1$ ;BRANCH IF NOT
106
107 006664 104401 001217 7$: TYPE , $CRLF ;CR-LF
108 006670 005004 CLR R4 ;THESE TWO LOOPS ARE ADDED TO
109 006672 005304 DEC R4 ;WAIT FOR TTY TO FINISH TYPING.
110 006674 001376 BNE .-2
111 006676 005304 DEC R4
112 006700 001376 BNE .-2
113
114 006702 000137 007612 JMP CMNSTART ;JUMP TO COMMON START
115
  
```

```
1          .SBTTL  STANDALONE INPUT ROUTINES
2
3          STANDALONE:
4          JSR      PC,$TKINT      ;INITIALIZE CONSOLE
5
6          INC      #-1            ;FIRST TIME THRU HERE ?
7          BEQ      3$             ;YES !!
8
9          ;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
10         1$:
11         TYPE     ,CNSLOO        ;MAINTAIN PREVIOUS PARAMETERS ?
12         RDCHR    (SP)+,$TMP1    ;GET RESPONSE
13         MOV      $TMP1          ;ECHO RESPONSE
14         TYPE     ,STMP1
15         CMPB     $TMP1,#'Y      ;YES RESPONSE ?
16         BNE     2$             ;NO!!
17         TYPE     ,SCRLF         ;CR-LF
18         JMP      CMNSTART      ;KEEP PREVIOUS PARAMETERS
19
20         2$:  CMPB     $TMP1,#'N   ;NO RESPONSE ?
21         BEQ     5$             ;YES, GET NEW PARAMETERS
22         TYPE     ,CNSLO8        ;NO, TYPE ' ILLEGAL INPUT '
23         BR      1$            ;TRY AGAIN
24
25         ;SEE IF OPERATOR WANTS HELP TEXT
26         3$:
27         TYPE     ,MSHELP        ;WANT HELP ?
28         RDCHR    (SP)+,$TMP1    ;GET RESPONSE
29         MOV      $TMP1          ;SAVE AND ECHO RESPONSE
30         TYPE     ,STMP1
31         CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ?
32         BEQ     4$             ;YES
33         CMPB     $TMP1,#'N      ;WAS IT A NO RESPONSE ?
34         BEQ     5$             ;YES
35         TYPE     ,CNSLO8        ;NO, TYPE ' ILLEGAL INPUT '
36         BR      3$            ;TRY AGAIN
37         4$:  TYPE     ,HELP      ;YES - TYPE HELP TEXT
38
39         ;SEE IF USER WANTS TO CHANGE ADDRESSES
40         5$:
41         TYPE     ,SCRLF         ;CR-LF
42         TYPE     ,UBUSQST      ;WANT TO CHANGE ADDRESS ?
43         RDCHR    (SP)+,$TMP1    ;GET RESPONSE
44         MOV      $TMP1          ;SAVE AND ECHO RESPONSE
45         TYPE     ,STMP1
46         CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ?
47         BEQ     6$             ;YES
48         CMPB     $TMP1,#'N      ;WAS IT A NO RESPONSE ?
49         BEQ     12$            ;YES
50         TYPE     ,CNSLO8        ;NO, TYPE ' ILLEGAL INPUT '
51         BR      5$+4          ;TRY AGAIN
52
53         ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
54         6$:
55         TYPE     ,CNSLO1        ;TYPE CURRENT BUS ADDRESS
56         MOV      $BASE,-(SP)    ;:SAVE $BASE FOR TYPEOUT
          TYPOC                   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```

57 007126 104401 063225          TYPE      ,QUES          ;TYPE " ? "
58 007132 104413          RDOCT           ;GET NEW BUS ADDRESS
59 007134 012637 001176        MOV      (SP)+,$TMP1    ;CARRIAGE RETURN ?
60 007140 001412          BEQ      8$           ;YES-SKIP TO NEXT ENTRY
61 007142 022737 160000 001176  CMP      #160000,$TMP1 ;BASE ADDRESS IN I/O PAGE ?
62 007150 101403          BLOS     7$           ;YES
63 007152 104401 063715        TYPE      ,CNSL02      ;TYPE WARNING MESSAGE
64 007156 000760          BR       6$+4         ;TRY AGAIN
65 007160 013737 001176 001276 7$: MOV      $TMP1,$BASE   ;STORE NEW BUS ADDRESS
66
67 007166 104401 063757          8$:  TYPE      ,CNSL03
68 007172 005046          CLR      -(SP)
69 007174 113716 001272        MOVVB   $VECT1,(SP)   ;GET CURRENT VECTOR ADDRESS
70 007200 104403          TYPOS
71 007202      003          .BYTE    3           ;TYPE 3 DIGITS
72 007203      000          .BYTE    0           ;SUPPRESS LEADING ZEROS
73 007204 104401 063225          TYPE      ,QUES          ;TYPE " ? "
74 007210 104413          RDOCT           ;GET NEW VECTOR ADDRESS
75 007212 012637 001176        MOV      (SP)+,$TMP1    ;CARRIAGE RETURN?
76 007216 001412          BEQ     10$          ;YES-SKIP TO NEXT ENTRY
77 007220 022737 001000 001176  CMP      #1000,$TMP1   ;VECTOR ADDRESS < 1000 ?
78 007226 101003          BHI     9$           ;YES!!
79 007230 104401 063777        TYPE      ,CNSL04      ;TYPE WARNING MESSAGE
80 007234 000754          BR       8$           ;RETRY
81 007236 113737 001176 001272 9$: MOVVB   $TMP1,$VECT1 ;STORE NEW VECTOR ADDRESS
82
83 007244 104401 064033          10$: TYPE      ,CNSL05
84 007250 005046          CLR      -(SP)
85 007252 113716 001273        MOVVB   $VECT1+1,(SP) ;GET CURRENT BR LEVEL
86 007256 006216          ASR     (SP)
87 007260 006216          ASR     (SP)
88 007262 006216          ASR     (SP)
89 007264 006216          ASR     (SP)
90 007266 006216          ASR     (SP)
91 007270 104403          TYPOS
92 007272      001          .BYTE    1           ;ONLY 1 DIGIT
93 007273      000          .BYTE    0           ;SUPPRESS LEADING ZEROS
94 007274 104401 063225          TYPE      ,QUES          ;TYPE " ? "
95 007300 104413          RDOCT           ;GET NEW PRIORITY
96 007302 012637 001176        MOV      (SP)+,$TMP1    ;CARRIAGE RETURN ?
97 007306 001424          BEQ     12$          ;YES-SKIP TO NEXT ENTRY
98 007310 023727 001176 000007  CMP      $TMP1,#7      ;LEGAL PRIORITY ?
99 007316 002403          BLT     11$          ;YES!!
100 007320 104401 064045        TYPE      ,CNSL06      ;TYPE WARNING MESSAGE
101 007324 000747          BR       10$         ;TRY AGAIN
102 007326 006337 001176          11$: ASL     $TMP1        ;STORE NEW PRIORITY
103 007332 006337 001176          ASL     $TMP1
104 007336 006337 001176          ASL     $TMP1
105 007342 006337 001176          ASL     $TMP1
106 007346 006337 001176          ASL     $TMP1
107 007352 113737 001176 001273  MOVVB   $TMP1,$VECT1+1
108
109
110 007360          ;DIALOGUE TO INPUT DEVICE NUMBERS
111 007360 005227 177777          12$: INC      #-1          ;FIRST TIME THRU ?
112 007364 001002          BNE     13$          ;BR IF NO
113 007366 104401 064076        TYPE      ,CNSL07      ;TYPE INPUT INSTRUCTIONS

```

```
114 007372 104401 001217        13$:    TYPE      ,SCRLF           :CR-LF
115 007376 005037 001300        14$:    CLR        $DEVM           :CLEAR DEVICE MAP
116 007402 104401 064263                TYPE      ,MSDRVS           :TYPE 'DRIVE(S): '
117 007406 104411                RDCHR
118 007410 012637 001176                MOV      (SP)+,$TMP1      :GET RESPONSE
119 007414 023727 001176 000101      CMP      $TMP1,#'A         :IS INPUT 'A' ?
120 007422 001007                BNE      15$              :NO
121 007424 104401 063220                TYPE      ,ALL            :YES, TYPE "ALL" AND GO
122 007430 012737 000377 001300      MOV      #377,$DEVM       :SET DEVICE MAP FOR ALL DRIVES
123 007436 000137 006462                JMP      XSIZ             :AUTO SIZE.
124
125 007442 023727 001176 000015  15$:    CMP      $TMP1,#CR        :CARRIAGE RETURN ?
126 007450 001436                BEQ      17$              :YES
127 007452 104401 001176                TYPE      , $TMP1         :ECHO RESPONSE
128 007456 023727 001176 000060      CMP      $TMP1,#'0        :NUMBER < 0 ?
129 007464 002430                BLT      17$              :YES
130 007466 023727 001176 000067      CMP      $TMP1,#'7        :NUMBER > 7 ?
131 007474 003427                BLE      18$              :NO
132 007476 000423                BR       17$              :ILLEGAL INPUT
133
134 007500 104411                16$:    RDCHR
135 007502 012637 001176                MOV      (SP)+,$TMP1      :GET RESPONSE
136 007506 023727 001176 000015      CMP      $TMP1,#CR        :CARRIAGE RETURN ?
137 007514 001432                BEQ      19$              :YES
138 007516 104401 063231                TYPE      ,COMMA          :TYPE ','
139 007522 104401 001176                TYPE      , $TMP1         :ECHO RESPONSE
140 007526 023727 001176 000060      CMP      $TMP1,#'0        :NUMBER < 0 ?
141 007534 002404                BLT      17$              :YES
142 007536 023727 001176 000067      CMP      $TMP1,#'7        :NUMBER > 7 ?
143 007544 003403                BLE      18$              :NO
144 007546 104401 064241                17$:    TYPE      ,CNSLO8        :TYPE "?ILLEGAL INPUT"
145 007552 000711                BR       14$              :RETRY
146
147 007554 013701 001176                18$:    MOV      $TMP1,R1         :R1 = DRIVE NUMBER
148 007560 042701 177770                BIC      #'C7,R1
149 007564 156137 064506 001300      BISB    ATNTBL(R1),$DEVM  :SET DEVICE IN MAP
150 007572 122737 000377 001300      CMPB    #377,$DEVM        :DONE ?
151 007600 101337                BHI     16$              :NO
152 007602 104401 001217                19$:    TYPE      ,SCRLF           :CR-LF
153 007606 000137 006462                JMP      XSIZ             :GO SIZE DEVICES
154
```

```

1          ;ASSEMBLE TEST QUE FROM DEVICE MAP
2 007612    CMNSTART:
3 007612    013700 001300    MOV    $DEVN,R0          ;R0 = DEVICE MAP
4 007616    012701 001466    MOV    #TSTQUE+2,R1     ;R1 = ADDRESS OF FIRST ENTRY IN QUE
5 007622    010137 001464    MOV    R1,TSTQUE       ;INITIALIZE ENTRY POINTER
6 007626    012702 000001    MOV    #1,R2           ;R2 = DEVICE POINTER
7 007632    005003          CLR    R3              ;R3 = DEVICE NUMBER
8 007634    030200          1$:  BIT    R2,R0       ;IS THIS DEVICE IN MAP ?
9 007636    001406          BEQ    2$              ;NO !!
10 007640   010311          MOV    R3,(R1)         ;YES - ENTER DEVICE NUMBER IN QUE
11 007642   116361 064506 000001  MOVB   ATNTBL(R3),1(R1) ;ENTER ATTENTION BIT IN QUE
12 007650   062701 000002          ADD    #2,R1           ;ADVANCE ENTRY POINTER
13 007654   006302          2$:  ASL    R2           ;ADVANCE DEVICE POINTER
14 007656   105702          TSTB   R2              ;DONE ALL DEVICES ?
15 007660   001402          BEQ    3$              ;YES
16 007662   005203          INC    R3              ;ADVANCE DEVICE NUMBER
17 007664   000763          BR     1$              ;ENTER NEXT DEVICE
18 007666   005011          3$:  CLR    (R1)         ;TERMINATE TEST QUE
19
20          ;SIZE FOR CLOCK
21 007670   004737 037600          JSR    PC,SIZCLK       ;SEE IF CLOCK PRESENT
22 007674   000413          BR     5$              ;YES - CLOCK IS PRESENT
23          4$:
24 007676   104000          EMT
25 007700   104401 007706          TYPE  ,65$             ;;TYPE ASCIZ STRING
26 007704   000405          BR     64$             ;;GET OVER THE ASCIZ
27          ;;65$: .ASCIZ <CRLF>/PROG HLT/
28          64$:
29          HALT          ;PROGRAM HALT !!
30          BR     4$
31          5$:
32          .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
33          TST    @#42    ;;ARE WE RUNNING UNDER XXDP/ACT?
34          BNE    66$     ;;BRANCH IF YES
35          007724 005737 000042  CMPB   $ENV,#1        ;;ARE WE RUNNING UNDER APT?
36          007730 001012          BEQ    66$             ;;BRANCH IF YES
37          007732 123727 001242 000001  CMP    SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
38          007740 001406          BNE    67$             ;;BRANCH IF NO
39          007742 023727 001154 000176  GTSWR          ;;GET SOFT-SWR SETTINGS
40          007750 001005          BR     67$
41          007752 104407          MOVB   #1,$AUTOB     ;;SET AUTO-MODE INDICATOR
42          007754 000403          66$:
43          007756 112737 000001 001150 67$:
44          28          READY: NOP              ;READY TO START TEST
45          29 007764 000240          TSTB   $DEVN          ;ANY DRIVES IN MAP ?
46          30 007766 105737 001300          BNE    2$             ;BR IF YES
47          31 007772 001007          TST    @#42          ;ANY MONITOR PRESENT ?
48          32 007774 005737 000042          BNE    1$             ;BR IF YES
49          33 010000 001002          JMP    START         ;JUMP TO START
50          34 010002 000137 005420          1$:  JMP    $EOP          ;RETURN CONTROL TO MONITOR
51          35 010006 000137 032340          2$:  CLRB   $TSTNM        ;RESET TEST NUMBER
52          36          CLR    $TIMES       ;INITIALIZE NUMBER OF ITERATIONS
53          37 010012 105037 001116          CLR    CTFLG         ;CLEAR CONTROL-C FLAG
54          38 010016 005037 001206          JSR    PC,$TKINT     ;INITIALIZE TTY
55          39 010022 005037 001326          MOV    #PR6,-(SP)    ;;PUT NEW PS ON STACK
56          40 010026 004737 061036          MOV    #64$,-(SP)   ;;PUT NEW PC ON STACK
57          41 010032 012746 000300
58          010036 012746 010044

```


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

1
2

```

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

```

010130
010130 000004
010132 000240
010134 012706 001100
010140 013700 001276
010144 013701 001464
010150 012737 000001 001226
3
4 010156 005001
5 010160 013746 000004
6 010164 013746 000006
7 010170 012737 010262 000004
8 010176 012737 000300 000006
9
10 010204 110160 000001
11 010210 010160 000002
12 010214 016002 000002
13 010220 010160 000004
14 010224 016002 000004
15 010230 010160 000010
16 010234 016002 000010
17 010240 010160 000022
18 010244 016002 000022
19 010250 012637 000006
20 010254 012637 000004
21 010260 000415
22
23 010262 022626 3$:
24 010264 012637 000006
25 010270 012637 000004
26 010274 104110
27 010276 005737 000042
28 010302 001002
29 010304 000137 005420
30
31 010310 000137 032340 5$:
32 010314 7$:
33
34
    
```

```

SCOPE
NOP
MOV #STACK,SP
MOV $BASE,R0
MOV TSTQUE,R1
MOV #1,$TESTN
CLR R1
MOV ERRVEC,-(SP)
MOV ERRVEC+2,-(SP)
MOV #3$,ERRVEC
MOV #PR6,ERRVEC+2
MOVB R1,RMCS1+1(R0)
MOV R1,RMWC(R0)
MOV RMWC(R0),R2
MOV R1,RMBA(R0)
MOV RMBA(R0),R2
MOV R1,RMCS2(R0)
MOV RMCS2(R0),R2
MOV R1,RMDB(R0)
MOV RMDB(R0),R2
MOV (SP)+,ERRVEC+2
MOV (SP)+,ERRVEC
BR 7$
3$: CMP (SP)+,(SP)+
MOV (SP)+,ERRVEC+2
MOV (SP)+,ERRVEC
EMT 110
TST @#42
BNE 5$
JMP START
5$: JMP $EOP
7$:
    
```

```

:SCOPE CALL
:START OF TEST
:INITIALIZE STACK POINTER
:R0 = UNIBUS ADDRESS
:(R1) = DEVICE BEING TESTED
::SET TEST NUMBER IN APT MAIL BOX
::PUSH ERRVEC ON STACK
::PUSH ERRVEC+2 ON STACK
:MOVE HI BYTE TO RMCS1
:MOVE WORD COUNT REGISTER
:MOVE BUS ADDRESS REGISTER
:MOVE CONTROL STATUS REGISTER
:MOVE DATA BUFFER
::POP STACK INTO ERRVEC+2
::POP STACK INTO ERRVEC
:NO BUS TIMEOUT OCCURRED
:ADJUST STACK
::POP STACK INTO ERRVEC+2
::POP STACK INTO ERRVEC
:STAND ALONE MODE ?
:NO!!
:YES-GO GET $BASE
:GO TO END OF PASS HANDLER
    
```

```

*****
:*TEST 2 DEVICE AVAILABLE TEST
*****
TST2:
    
```

```

010314
010314 000004
010316 000240
010320 012706 001100
010324 013700 001276
010330 013701 001464
010334 012737 000002 001226
35
36 010342 004737 046464
010346 000404
010350 000240
010352 104000
    
```

```

SCOPE
NOP
MOV #STACK,SP
MOV $BASE,R0
MOV TSTQUE,R1
MOV #2,$TESTN
JSR PC,CNTCLR
BR 2$
NOP
EMT
    
```

```

:SCOPE CALL
:START OF TEST
:INITIALIZE STACK POINTER
:R0 = UNIBUS ADDRESS
:(R1) = DEVICE BEING TESTED
::SET TEST NUMBER IN APT MAIL BOX
:GO ISSUE CONTROLLER CLEAR
:GO TO 2$ IF NO ERROR
:RETURN HERE IF ERROR
:ERROR NUMBER DEFINED BY SUBROUTINE
    
```

```

37 010354 000137 010474      JMP       7$           ;GO TO 7$ IF ERROR
38 010360      2$:      MOV       ERRVEC,-(SP)   ;;PUSH ERRVEC ON STACK
39 010364 013746 000004      MOV       ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
40 010370 012737 010460 000004  MOV       #5$,ERRVEC
41 010376 013737 000300 000006  MOV       PR6,ERRVEC+2
42
43 010404 016037 000000 001176  MOV       RMCS1(RO),$TMP1 ;GET DVA STATUS
44 010412 016037 000010 001174  MOV       RMCS2(RO),$TMP0 ;GET NED STATUS
45 010420 012637 000006      MOV       (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
46 010424 012637 000004      MOV       (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
47 010430 032737 010000 001174  BIT       #NED,$TMP0      ;NONEXISTENT DEVICE ?
48 010436 001402      BEQ       3$           ;NO!!
49 010440 104111      EMT
50 010442 000414      BR        7$
51 010444 032737 004000 001176  3$:      BIT       #DVA,$TMP1      ;DEVICE AVAILABLE ?
52 010452 001012      BNE       9$           ;YES!!
53 010454 104112      EMT
54 010456 000406      BR        7$
55
56 010460 022626      5$:      CMP       (SP)+,(SP)+    ;ADJUST STACK
57 010462 012637 000006      MOV       (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
58 010466 012637 000004      MOV       (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
59 010472 104113      EMT
60 010474 000137 032302      7$:      JMP       $EOSP
61 010500      9$:
62
63

```

```

.....
: *TEST 3 DRIVE TYPE TEST
: .....
TST3:

```

```

010500
010500 000004      SCOPE
010502 000240      NOP                ;SCOPE CALL
010504 012706 001100  MOV       #STACK,SP    ;START OF TEST
010510 013700 001276  MOV       $BASE,R0     ;INITIALIZE STACK POINTER
010514 013701 001464  MOV       TSTQUE,R1    ;R0 = UNIBUS ADDRESS
010520 012737 000003 001226  MOV       #3,$TESTIN   ;(R1) = DEVICE BEING TESTED
                                ;;SET TEST NUMBER IN APT MAIL BOX
64
65 010526 004737 046464  JSR       PC,CNTCLR    ;GO ISSUE CONTROLLER CLEAR
010532 000404      BR        2$          ;GO TO 2$ IF NO ERROR
010534 000240      NOP
010536 104000      EMT
010540 000137 010676  JMP       4$          ;ERROR NUMBER DEFINED BY SUBROUTINE
                                ;GO TO 4$ IF ERROR
66 010544      2$:
67 010544 112737 000026 001522  MOVVB    #RMDT,GETINX  ;SETUP GET INDEX TABLE
010552 112737 000200 001523  MOVVB    #200,GETINX+1 ;SETUP TERMINATOR BYTE
010560 012737 010702 001362  MOV       #5$,RMDTI    ;SET RMDT INPUT BUFFER = 5$
010566 004737 037110  JSR       PC,GET      ;GO READ RMDT VIA GET SUBROUTINE
010572 000402      BR        3$          ;GO TO 3$ IF NO ERROR
010574 000240      NOP
010576 104000      EMT                ;RETURN HERE IF ERROR
                                ;ERROR DEFINED BY GET SUBROUTINE
68
69 010600 022737 020024 001362  3$:      CMP       #SNGPRT,RMDTI ;SINGLE PORT RM03 ?
70 010606 001435      BEQ       5$          ;YES !!
71 010610 022737 024024 001362  CMP       #DULPRT,RMDTI ;DUAL PORT RM03 ?
72 010616 001431      BEQ       5$          ;YES !!
73

```



```
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 30$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 190$ IF ERROR
011042 000404 BR 30$
011044 000240 NOP
011046 104000 EMT
011050 000137 011544 JMP 190$
111 011054 30$:
112
113 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
114 011054 012737 000005 001410 MOV #SEEK!GO, RMCS10 ;CHANGE COMMAND TO SEEK
115 011062 012702 001551 MOV #PUTINX, R2 ;WRITE REGISTER INDEX TABLE
116 011066 112722 000006 MOVB #RMDA, (R2)+
117 011072 112722 000034 MOVB #RMDC, (R2)+
118 011076 112722 000032 MOVB #RMOF, (R2)+
119 011102 112722 000000 MOVB #RMCS1, (R2)+
120 011106 112722 000200 MOVB #200, (R2)+
121
122 011112 004737 037360 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011116 000404 BR 40$ ;GO TO 40$ IF NO ERROR
011120 000240 NOP ;RETURN HERE IF ERROR
011122 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
011124 000137 011544 JMP 190$ ;GO TO 190$ IF ERROR
123 011130 40$:
124
125 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
126 011130 004737 037024 JSR PC, GETSTS ;SETUP FOR STATUS
127 011134 004737 037722 JSR PC, TIMEOUT ;WAIT FOR SEEK TO COMPLETE
128 011140 50$:
129
130 ;GO READ SEEK STATUS
131 011140 004737 037110 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
011144 000404 BR 60$ ;GO TO 60$ IF NO ERROR
011146 000240 NOP ;RETURN HERE IF ERROR
011150 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
011152 000137 011544 JMP 190$ ;GO TO 190$ IF ERROR
132 011156 60$:
133
134 ;VERIFY THE RESULTS OF THE SEEK COMMAND
135 011156 004737 045224 JSR PC, SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
011162 000405 BR 70$ ;GO TO 70$ IF NO ERROR
011164 000240 NOP ;RETURN HERE IF ERROR
011166 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
011170 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
011172 000137 011544 JMP 190$ ;GO TO 190$ IF ERROR
136 011176 70$:
137
138 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
139 011176 012737 000063 001410 MOV #WH!GO, RMCS10 ;WRITE HEADER AND DATA
140 011204 012702 001554 MOV #PUTINX+3, R2 ;EXTEND REGISTER INDEX TABLE
141 011210 112722 000002 MOVB #RMWC, (R2)+
142 011214 112722 000004 MOVB #RMBA, (R2)+
143 011220 112722 000000 MOVB #RMCS1, (R2)+
144 011224 112722 000200 MOVB #200, (R2)+
145
146 011230 004737 037360 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011234 000404 BR 80$ ;GO TO 80$ IF NO ERROR
011236 000240 NOP ;RETURN HERE IF ERROR
```

```

147 011240 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
      011242 000137 011544  JMP          190$          ;GO TO 190$ IF ERROR
148
149
150 011246 004737 037722  ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
      JSR          PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
151
152 011252 004737 037110  JSR          PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011256 000404          BR          90$          ;GO TO 90$ IF NO ERROR
      011260 000240          NOP          ;RETURN HERE IF ERROR
      011262 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
      011264 000137 011544  JMP          190$          ;GO TO 190$ IF ERROR
153 011270
154
155
156 011270 004737 040106  ;VERIFY RESULTS OF WRITE COMMAND
      JSR          PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      011274 000405          BR          100$         ;GO TO 100$ IF NO ERROR
      011276 000240          NOP          ;RETURN HERE IF ERROR
      011300 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
      011302 004736          JSR          PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      011304 000137 011544  JMP          190$         ;GO TO 190$ IF ERROR
157 011310
158 011310 004737 052622  ;GO VERIFY RESULTS OF DATA TRANSFER
      JSR          PC,DTASTS      ;GO TO 110$ IF NO ERROR
      011314 000405          BR          110$         ;RETURN HERE IF ERROR
      011316 000240          NOP          ;ERROR # DEFINED BY DTASTS SUBROUTINE
      011320 104000          EMT          ;GO BACK FOR MORE ERROR CHECKS
      011322 004736          JSR          PC,@(SP)+    ;GO TO 190$ IF ERROR
      011324 000137 011544  JMP          190$
159 011330
160 011330 004737 040740  ;GO CHECK FOR SECONDARY ERRORS
      JSR          PC,SECERR      ;GO TO 120$ IF NO ERROR
      011334 000405          BR          120$         ;RETURN HERE IF ERROR
      011336 000240          NOP          ;ERROR # DEFINED BY SECERR SUBROUTINE
      011340 104000          EMT          ;GO BACK FOR MORE ERROR CHECKS
      011342 004736          JSR          PC,@(SP)+    ;GO TO 190$ IF ERROR
      011344 000137 011544  JMP          190$
161 011350
162
163
164 011350 012737 000073 001410 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
      MOV          #RH!GO, RMCS10 ;READ HEADER & DATA COMMAND
165 011356 012737 102212 001414  MOV          #BUFTWO, RMBAO    ;CHANGE BUS ADDRESS
166
167 011364 004737 037360  JSR          PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      011370 000404          BR          130$         ;GO TO 130$ IF NO ERROR
      011372 000240          NOP          ;RETURN HERE IF ERROR
      011374 104000          EMT          ;ERROR # DEFINED BY PUT SUBROUTINE
      011376 000137 011544  JMP          190$         ;GO TO 190$ IF ERROR
168 011402
169
170
171 011402 004737 037722  ;WAIT FOR READ TO COMPLETE AND GET STATUS
      JSR          PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
172
173 011406 004737 037110  JSR          PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      011412 000404          BR          140$         ;GO TO 140$ IF NO ERROR
      011414 000240          NOP          ;RETURN HERE IF ERROR
      011416 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
      011420 000137 011544  JMP          190$         ;GO TO 190$ IF ERROR
174 011424

```

175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193

011424 004737 040106
011430 000405
011432 000240
011434 104000
011436 004736
011440 000137 011544
011444 004737 052622
011450 000405
011452 000240
011454 104000
011456 004736
011460 000137 011544
011464 004737 040740
011470 000405
011472 000240
011474 104000
011476 004736
011500 000137 011544
011504
011504 004737 036462
011510 101206
011512 102212
011514 000402
011516 000240
011520 104000
011522 032737 010000 001442
011530 001005
011532 012737 010000 001442
011540 000137 010736
011544
011544 000004
011546 000240
011550 012706 001100
011554 013700 001276
011560 013701 001464
011564 012737 000005 001226
011572 012737 000000 001444
011600 012737 000000 001416
011606 012737 010000 001442
011614 012737 177376 001412
011622 012737 101206 001414
011630 012737 000062 001410
202

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

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

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

170$:

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

180$:
BIT  #FMT16,RMOFO ;TEST 16 BIT MODE YET ?
BNE  190$ ;YES
MOV  #FMT16,RMOFO ;SET 16 BIT MODE AND
JMP  5$ ;TEST AGAIN.

190$:

;*****
;*TEST 5 ZERO FILL TEST
;*****
TST5:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV  #STACK,SP ;INITIALIZE STACK POINTER
MOV  $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV  TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV  #5,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

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

```

203          ;VERIFY THAT SECTOR IS NOT BAD
011636 004737 034276 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
011642 000405 BR 10$ ;GO TO 10$ IF NO ERROR
011644 104401 063120 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
011650 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
011652 000137 012400 JMP 180$ ;GO TO 180$ IF ERROR
204 011656 10$:
205 011656 012737 064620 001174 MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
206 011664 012737 000001 001176 MOV #1,$TMP1
207 011672 004737 036224 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
208 011676
209
210
211 011676 004737 033352 ;PREPARE DEVICE FOR DATA TRANSFER
011702 154130 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 30$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 180$ IF ERROR
011704 000404 BR 30$
011706 000240 NOP
011710 104000 EMT
011712 000137 012400 JMP 180$
212 011716 30$:
213
214 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
215 011716 012737 000005 001410 MOV #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
216 011724 012702 001551 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
217 011730 112722 000006 MOVB #RMDA,(R2)+
218 011734 112722 000034 MOVB #RMDC,(R2)+
219 011740 112722 000032 MOVB #RMOF,(R2)+
220 011744 112722 000000 MOVB #RMCS1,(R2)+
221 011750 112722 000200 MOVB #200,(R2)+
222
223 011754 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
011760 000404 BR 40$ ;GO TO 40$ IF NO ERROR
011762 000240 NOP ;RETURN HERE IF ERROR
011764 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
011766 000137 012400 JMP 180$ ;GO TO 180$ IF ERROR
224 011772 40$:
225
226 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
227 011772 004737 037024 JSR PC,GETSTS ;SETUP FOR STATUS
228 011776 004737 037722 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
229 012002 50$:
230
231 ;GO READ SEEK STATUS
232 012002 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
012006 000404 BR 60$ ;GO TO 60$ IF NO ERROR
012010 000240 NOP ;RETURN HERE IF ERROR
012012 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
012014 000137 012400 JMP 180$ ;GO TO 180$ IF ERROR
233 012020 60$:
234

```



```
235                ;VERIFY THE RESULTS OF THE SEEK COMMAND
236 012020 004737 045224      JSR      PC,SEKSTS      ;GO VERIFY RESULTS OF SEEK OPERATION
                               BR        70$           ;GO TO 70$ IF NO ERROR
                               NOP        ;RETURN HERE IF ERROR
                               EMT        ;ERROR # DEFINED BY SEKSTS SUBROUTINE
                               JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                               JMP      180$         ;GO TO 180$ IF ERROR
237 012040          70$:
238
239                ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
240 012040 012737 177776 001412  MOV      #-2,RMWC0      ;FORMAT PARTIAL SECTOR
241 012046 012737 000063 001410  MOV      #WH!GO,RMCS10  ;WRITE HEADER AND DATA
242 012054 012702 001554          MOV      #PUTINX+3,R2   ;EXTEND REGISTER INDEX TABLE
243 012060 112722 000002          MOVB     #RMWC,(R2)+
244 012064 112722 000004          MOVB     #RMBA,(R2)+
245 012070 112722 000000          MOVB     #RMCS1,(R2)+
246 012074 112722 000200          MOVB     #200,(R2)+
247
248 012100 004737 037360          JSR      PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                               BR        80$           ;GO TO 80$ IF NO ERROR
                               NOP        ;RETURN HERE IF ERROR
                               EMT        ;ERROR # DEFINED BY PUT SUBROUTINE
                               JMP      180$         ;GO TO 180$ IF ERROR
249 012116          80$:
250
251                ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
252 012116 004737 037722      JSR      PC,TIMOUT     ;WAIT FOR COMMAND TO COMPLETE
253
254 012122 004737 037110          JSR      PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
                               BR        90$           ;GO TO 90$ IF NO ERROR
                               NOP        ;RETURN HERE IF ERROR
                               EMT        ;ERROR # DEFINED BY GET SUBROUTINE
                               JMP      180$         ;GO TO 180$ IF ERROR
255 012140          90$:
256
257                ;VERIFY RESULTS OF WRITE COMMAND
258 012140 004737 040106      JSR      PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
                               BR        100$          ;GO TO 100$ IF NO ERROR
                               NOP        ;RETURN HERE IF ERROR
                               EMT        ;ERROR # DEFINED BY PRIERR SUBROUTINE
                               JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                               JMP      180$         ;GO TO 180$ IF ERROR
259 012160          100$:
260 012160 004737 052622      JSR      PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
                               BR        110$          ;GO TO 110$ IF NO ERROR
                               NOP        ;RETURN HERE IF ERROR
                               EMT        ;ERROR # DEFINED BY DTASTS SUBROUTINE
                               JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                               JMP      180$         ;GO TO 180$ IF ERROR
261 012200          110$:
262 012200 004737 040740      JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
                               BR        120$          ;GO TO 120$ IF NO ERROR
                               NOP        ;RETURN HERE IF ERROR
                               EMT        ;ERROR # DEFINED BY SECERR SUBROUTINE
                               JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                               JMP      180$         ;GO TO 180$ IF ERROR
263 012220          120$:
```

```

264
265
266 012220 012737 177376 001412 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
267 012226 012737 000073 001410     MOV    #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
268 012234 012737 102212 001414     MOV    #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
269                                     MOV    #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
270 012242 004737 037360                JSR    PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    012246 000404                BR     130$ ;GO TO 130$ IF NO ERROR
    012250 000240                NOP ;RETURN HERE IF ERROR
    012252 104000                EMT ;ERROR # DEFINED BY PUT SUBROUTINE
    012254 000137 012400                JMP    180$ ;GO TO 180$ IF ERROR
271 012260                130$:
272
273                                     ;WAIT FOR READ TO COMPLETE AND GET STATUS
274 012260 004737 037722                JSR    PC,TIMOUT ;WAIT FOR READ TO COMPLETE
275
276 012264 004737 037110                JSR    PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
    012270 000404                BR     140$ ;GO TO 140$ IF NO ERROR
    012272 000240                NOP ;RETURN HERE IF ERROR
    012274 104000                EMT ;ERROR # DEFINED BY GET SUBROUTINE
    012276 000137 012400                JMP    180$ ;GO TO 180$ IF ERROR
277 012302                140$:
278
279                                     ;VERIFY THE RESULTS OF READ OPERATION
280 012302 004737 040106                JSR    PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
    012306 000405                BR     150$ ;GO TO 150$ IF NO ERROR
    012310 000240                NOP ;RETURN HERE IF ERROR
    012312 104000                EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
    012314 004736                JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    012316 000137 012400                JMP    180$ ;GO TO 180$ IF ERROR
281 012322                150$:
282 012322 004737 052622                JSR    PC,DASTS ;GO VERIFY RESULTS OF DATA TRANSFER
    012326 000405                BR     160$ ;GO TO 160$ IF NO ERROR
    012330 000240                NOP ;RETURN HERE IF ERROR
    012332 104000                EMT ;ERROR # DEFINED BY DASTS SUBROUTINE
    012334 004736                JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    012336 000137 012400                JMP    180$ ;GO TO 180$ IF ERROR
283 012342                160$:
284 012342 004737 040740                JSR    PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
    012346 000405                BR     170$ ;GO TO 170$ IF NO ERROR
    012350 000240                NOP ;RETURN HERE IF ERROR
    012352 104000                EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
    012354 004736                JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
    012356 000137 012400                JMP    180$ ;GO TO 180$ IF ERROR
285 012362                170$:
286
287                                     ;VERIFY DATA
288 012362 004737 036462                JSR    PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
    012366 101206                .WORD BUFOE ;STARTING ADDRESS OF WRITE BUFFER
    012370 102212                .WORD BUFTWO ;STARTING ADDRESS OF READ BUFFER
    012372 000402                BR     180$ ;GO TO 180$ IF NO ERROR
    012374 000240                NOP ;RETURN HERE IF ERROR
    012376 104000                EMT ;ERROR # DEFINED BY CMPBUF SUBROUTINE
289 012400                180$:
290
291
    ;*****
    ;*TEST 6             FORMAT CHECK ZEROS
    ;*****
    
```



```

012620 104000      EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
012622 000137 013170 40$: JMP 260$                ;GO TO 260$ IF ERROR
322 012626
323
324 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
325 012626 004737 037024 JSR PC,GETSTS          ;SETUP FOR STATUS
326 012632 004737 037722 JSR PC,TIMOUT          ;WAIT FOR SEEK TO COMPLETE
327 012636 50$:
328
329 ;GO READ SEEK STATUS
330 012636 004737 037110 JSR PC,GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
012642 000404 BR 60$                 ;GO TO 60$ IF NO ERROR
012644 000240 NOP                   ;RETURN HERE IF ERROR
012646 104000 EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
012650 000137 013170 60$: JMP 260$                ;GO TO 260$ IF ERROR
331 012654
332
333 ;VERIFY THE RESULTS OF THE SEEK COMMAND
334 012654 004737 045224 JSR PC,SEKSTS          ;GO VERIFY RESULTS OF SEEK OPERATION
012660 000405 BR 70$                 ;GO TO 70$ IF NO ERROR
012662 000240 NOP                   ;RETURN HERE IF ERROR
012664 104000 EMT                      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
012666 004736 JSR PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
012670 000137 013170 70$: JMP 260$                ;GO TO 260$ IF ERROR
335 012674
336
337 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
338 012674 012737 000063 001410 MOV #WH!GO,RMCS10      ;WRITE HEADER AND DATA
339 012702 012702 001554 MOV #PUTINX+3,R2       ;EXTEND REGISTER INDEX TABLE
340 012706 112722 000002 MOVB #RMWC,(R2)+
341 012712 112722 000004 MOVB #RMBA,(R2)+
342 012716 112722 000000 MOVB #RMCS1,(R2)+
343 012722 112722 000200 MOVB #200,(R2)+
344
345 012726 004737 037360 JSR PC,PUT              ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
012732 000404 BR 80$                 ;GO TO 80$ IF NO ERROR
012734 000240 NOP                   ;RETURN HERE IF ERROR
012736 104000 EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
012740 000137 013170 80$: JMP 260$                ;GO TO 260$ IF ERROR
346 012744
347
348 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
349 012744 004737 037722 JSR PC,TIMOUT          ;WAIT FOR COMMAND TO COMPLETE
350
351 JSR PC,GET              ;GO READ REGISTER(S) WITH GET SUBROUTINE
012750 004737 037110 BR 90$                 ;GO TO 90$ IF NO ERROR
012754 000404 NOP                   ;RETURN HERE IF ERROR
012756 000240 EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
012760 104000 JMP 260$                ;GO TO 260$ IF ERROR
012762 000137 013170 90$:
352 012766
353
354 ;VERIFY RESULTS OF WRITE COMMAND
355 012766 004737 040106 JSR PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
012772 000405 BR 100$                ;GO TO 100$ IF NO ERROR
012774 000240 NOP                   ;RETURN HERE IF ERROR
012776 104000 EMT                      ;ERROR # DEFINED BY PRIERR SUBROUTINE
013000 004736 JSR PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS

```

```

356 013002 000137 013170          JMP      260$          ;GO TO 260$ IF ERROR
357 013006 004737 052622 100$:   JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR        110$          ;GO TO 110$ IF NO ERROR
                                NOP                      ;RETURN HERE IF ERROR
                                EMT                      ;ERROR # DEFINED BY DTASTS SUBROUTINE
358 013022 000137 013170          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP      260$          ;GO TO 260$ IF ERROR
359 013026 004737 040740 110$:   JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
                                BR        120$          ;GO TO 120$ IF NO ERROR
                                NOP                      ;RETURN HERE IF ERROR
                                EMT                      ;ERROR # DEFINED BY SECERR SUBROUTINE
360 013042 000137 013170          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP      260$          ;GO TO 260$ IF ERROR
361 013046 012737 000053 001410 ;WRITE CHECK HEADER AND DATA FOR SECTOR JUST WRITTEN
362 013046 000404 000053 001410 MOV      #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
363 013054 004737 037360          JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
364 013060 000404 037360          BR        130$          ;GO TO 130$ IF NO ERROR
365 013062 000240 037360          NOP                      ;RETURN HERE IF ERROR
366 013064 104000 037360          EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
367 013066 000137 013170          JMP      260$          ;GO TO 260$ IF ERROR
368 013072 004737 037722 130$:   ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
369 013072 004737 037722          JSR      PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
370 013076 004737 037110          JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
371 013102 000404 037110          BR        140$          ;GO TO 140$ IF NO ERROR
372 013104 000240 037110          NOP                      ;RETURN HERE IF ERROR
373 013106 104000 037110          EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
374 013110 000137 013170          JMP      260$          ;GO TO 260$ IF ERROR
375 013114 004737 040106 140$:   ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
376 013120 000404 040106          JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
377 013122 000240 040106          BR        150$          ;GO TO 150$ IF NO ERROR
378 013124 104000 040106          NOP                      ;RETURN HERE IF ERROR
379 013126 004736 040106          EMT                      ;ERROR # DEFINED BY PRIERR SUBROUTINE
380 013130 000137 013170          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
381 013134 000137 013170          JMP      260$          ;GO TO 260$ IF ERROR
382 013134 004737 052622 150$:   JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
383 013140 000404 052622          BR        160$          ;GO TO 160$ IF NO ERROR
384 013142 000240 052622          NOP                      ;RETURN HERE IF ERROR
385 013144 104000 052622          EMT                      ;ERROR # DEFINED BY DTASTS SUBROUTINE
386 013146 004736 052622          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
387 013150 000137 013170          JMP      260$          ;GO TO 260$ IF ERROR
388 013154 004737 040740 160$:   JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
389 013160 000403 040740          BR        170$          ;GO TO 170$ IF NO ERROR
390 013162 000240 040740          NOP                      ;RETURN HERE IF ERROR
391 013164 104000 040740          EMT                      ;ERROR # DEFINED BY SECERR SUBROUTINE
392 013166 004736 040740          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS

```

```

380 013170          170$:
381
382 013170          260$:
383
384
:*****
:*TEST 7          FORMAT CHECK ZEROS W/ WCE ERROR
:*****
TST7:
          SCOPE          :SCOPE CALL
013170  000004          NOP          :START OF TEST
013172  000240          MOV          #STACK,SP      :INITIALIZE STACK POINTER
013174  012706  001100  MOV          $BASE,R0       :R0 = UNIBUS ADDRESS
013200  013700  001276  MOV          TSTQUE,R1      :(R1) = DEVICE BEING TESTED
013204  013701  001464  MOV          #7,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
013210  012737  000007  001226

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
385
386
387 013216  012737  000000  001444  MOV          #0,RMDCO      :CYLINDER = 0
388 013224  012737  000000  001416  MOV          #0,RMDAO      :TRACK = 0, SECTOR = 0
389 013232  012737  010000  001442  MOV          #FMT16,RMOFO  :16 BIT FORMAT
390 013240  012737  177376  001412  MOV          #-258.,RMWCO  :2 + 256 WORDS (2'S COMP)
391 013246  012737  101206  001414  MOV          #BUFONE,RMBAO :DATA BUFFER ADDRESS
392 013254  012737  000062  001410  MOV          #WH,RMCS10    :WRITE HEADER AND DATA
393
394
;VERIFY THAT SECTOR IS NOT BAD
          JSR          PC,BADSCT  :CALL BAD SECTOR MODULE
          BR          10$         :GO TO 10$ IF NO ERROR
          TYPE        ,SCTMSG    :TYPE BAD SECTOR MESSAGE
          EMT
          JMP          260$      :ERROR # DEFINED BY BADSCT SUBROUTINE
                                :GO TO 260$ IF ERROR
10$:
395 013302
396 013302  012737  064620  001174  MOV          #ZEROS,$TMPO  :USE ALL ONES DATA PATTERN
397 013310  012737  000001  001176  MOV          #1,$TMP1
398 013316  004737  036224  JSR          PC,GENBUF     :GO GENERATE DATA BUFFER
399 013322
400
401
;PREPARE DEVICE FOR DATA TRANSFER
          JSR          PC,TSTPRP  :PREPARE DEVICE FOR TEST
          .WORD        154130    :TASK DESCRIPTOR AS FOLLOWS:
                                :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                :CLEAR CONTROLLER & SELECT DEVICE
                                :VERIFY CONTROLLER CLEAR OPERATION
                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                :VERIFY PACK ACKNOWLEDGE
                                :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                :VERIFY RECALIBRATION
                                :GO TO 30$ IF NO ERROR
                                :RETURN HERE IF ERROR
                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                :GO TO 260$ IF ERROR
          BR          30$
          NOP
          EMT
          JMP          260$
30$:
403 013342
404
405
;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
406 013342  012737  000005  001410  MOV          #SEEK!GO,RMCS10 :CHANGE COMMAND TO SEEK
407 013350  012702  001551  MOV          #PUTINX,R2     :WRITE REGISTER INDEX TABLE
408 013354  112722  000006  MOVB        #RMDA,(R2)+
409 013360  112722  000034  MOVB        #RMDC,(R2)+
410 013364  112722  000032  MOVB        #RMOF,(R2)+
    
```

```

411 013370 112722 000000        MOV  B   #RMCS1,(R2)+
412 013374 112722 000200        MOV  B   #200,(R2)+
413
414 013400 004737 037360        JSR   PC,PUT          :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                               BR    40$          :GO TO 40$ IF NO ERROR
                               NOP           :RETURN HERE IF ERROR
                               EMT        104000  :ERROR # DEFINED BY PUT SUBROUTINE
                               JMP        013412 000137 014130 :GO TO 260$ IF ERROR
415 013416                               40$:
416
417                               ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
418 013416 004737 037024        JSR   PC,GETSTS      :SETUP FOR STATUS
419 013422 004737 037722        JSR   PC,TIMOUT      :WAIT FOR SEEK TO COMPLETE
420 013426                               50$:
421
422                               ;GO READ SEEK STATUS
423 013426 004737 037110        JSR   PC,GET          :GO READ REGISTER(S) WITH GET SUBROUTINE
                               BR    60$          :GO TO 60$ IF NO ERROR
                               NOP           :RETURN HERE IF ERROR
                               EMT        104000  :ERROR # DEFINED BY GET SUBROUTINE
                               JMP        013440 000137 014130 :GO TO 260$ IF ERROR
424 013444                               60$:
425
426                               ;VERIFY THE RESULTS OF THE SEEK COMMAND
427 013444 004737 045224        JSR   PC,SEKSTS      :GO VERIFY RESULTS OF SEEK OPERATION
                               BR    70$          :GO TO 70$ IF NO ERROR
                               NOP           :RETURN HERE IF ERROR
                               EMT        104000  :ERROR # DEFINED BY SEKSTS SUBROUTINE
                               JSR   PC,@(SP)+  :GO BACK FOR MORE ERROR CHECKS
                               JMP        013460 000137 014130 :GO TO 260$ IF ERROR
428 013464                               70$:
429
430                               ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
431 013464 012737 000063 001410  MOV   #WH!GO,RMCS10  :WRITE HEADER AND DATA
432 013472 012702 001554        MOV   #PUTINX+3,R2   :EXTEND REGISTER INDEX TABLE
433 013476 112722 000002        MOV  B   #RMWC,(R2)+
434 013502 112722 000004        MOV  B   #RMBA,(R2)+
435 013506 112722 000000        MOV  B   #RMCS1,(R2)+
436 013512 112722 000200        MOV  B   #200,(R2)+
437
438 013516 004737 037360        JSR   PC,PUT          :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                               BR    80$          :GO TO 80$ IF NO ERROR
                               NOP           :RETURN HERE IF ERROR
                               EMT        104000  :ERROR # DEFINED BY PUT SUBROUTINE
                               JMP        013530 000137 014130 :GO TO 260$ IF ERROR
439 013534                               80$:
440
441                               ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
442 013534 004737 037722        JSR   PC,TIMOUT      :WAIT FOR COMMAND TO COMPLETE
443
444 013540 004737 037110        JSR   PC,GET          :GO READ REGISTER(S) WITH GET SUBROUTINE
                               BR    90$          :GO TO 90$ IF NO ERROR
                               NOP           :RETURN HERE IF ERROR
                               EMT        104000  :ERROR # DEFINED BY GET SUBROUTINE
                               JMP        013552 000137 014130 :GO TO 260$ IF ERROR
445 013556                               90$:
446

```

```
447 ;VERIFY RESULTS OF WRITE COMMAND
448 013556 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
013562 000405 BR 100$ ;GO TO 100$ IF NO ERROR
013564 000240 NOP ;RETURN HERE IF ERROR
013566 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
013570 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
013572 000137 014130 JMP 260$ ;GO TO 260$ IF ERROR
449 013576 100$:
450 013576 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
013602 000405 BR 110$ ;GO TO 110$ IF NO ERROR
013604 000240 NOP ;RETURN HERE IF ERROR
013606 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
013610 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
013612 000137 014130 JMP 260$ ;GO TO 260$ IF ERROR
451 013616 110$:
452 013616 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
013622 000405 BR 120$ ;GO TO 120$ IF NO ERROR
013624 000240 NOP ;RETURN HERE IF ERROR
013626 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
013630 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
013632 000137 014130 JMP 260$ ;GO TO 260$ IF ERROR
453 013636 120$:
454
455 ;ALTER DATA BUFFER
456 013636 005137 102210 COM BUFTWO-2 ;COMPLEMENT LAST DATA WORD
457
458 ;SETUP AND WRITE CHECK HEADER AND DATA COMMAND
459 013642 012737 000053 001410 MOV #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
460
461 013650 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
013654 000404 BR 180$ ;GO TO 180$ IF NO ERROR
013656 000240 NOP ;RETURN HERE IF ERROR
013660 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
013662 000137 014130 JMP 260$ ;GO TO 260$ IF ERROR
462 013666 180$:
463
464 ;WAIT FOR WRITE CHECK COMMAND TO COMPLETE AND READ STATUS
465 013666 004737 037722 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
466
467 013672 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
013676 000404 BR 190$ ;GO TO 190$ IF NO ERROR
013700 000240 NOP ;RETURN HERE IF ERROR
013702 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
013704 000137 014130 JMP 260$ ;GO TO 260$ IF ERROR
468 013710 190$:
469
470 ;CHECK FOR PRIMARY ERRORS
471 013710 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
013714 000405 BR 200$ ;GO TO 200$ IF NO ERROR
013716 000240 NOP ;RETURN HERE IF ERROR
013720 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
013722 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
013724 000137 014130 JMP 260$ ;GO TO 260$ IF ERROR
472 013730 200$:
473
474 ;MAKE SURE THE WRITE CHECK ERROR WAS DETECTED
475 013730 032737 040000 001344 BIT #WCE,RMCS21 ;IS WRITE CHECK ERROR SET??
```



```

476 013736 001023          BNE      210$          ;YES!!
477
478 013740 004737 052622   JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
    013744 000405          BR       205$          ;GO TO 205$ IF NO ERROR
    013746 000240          NOP      ;RETURN HERE IF ERROR
    013750 104000          EMT     ;ERROR # DEFINED BY DTASTS SUBROUTINE
    013752 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    013754 000137 014130   JMP      260$          ;GO TO 260$ IF ERROR
479
480 013760 013737 001344 001140 205$:  MOV     RMCS2I,$GDDAT   ;LOAD EXPECTED STATUS
481 013766 052737 040000 001140     BIS     #WCE,$GDDAT
482 013774 013737 001344 001142     MOV     RMCS2I,$BDDAT   ;LOAD RECEIVED STATUS
483 014002 104337          EMT     337
484 014004 000451          BR       260$
485 014006          210$:
486
487          ;VERIFY THE ADDRESS OF THE WRITE CHECK ERROR
488 014006 012737 102210 001134     MOV     #BUFTWO-2,$GDADR;LOAD EXPECTED ADDRESS
489 014014 013737 001340 001136     MOV     RMBAI,$BDADR   ;LOAD RECEIVED ADDRESS
490 014022 162737 000002 001136     SUB     #2,$BDADR      ;DECREMENT RECEIVED ADDRESS
491
492          ;GET WCE DATA AND VERIFY IT IS OK
493 014030 112737 000022 001522     MOV     #RMDB,GETINX   ;SETUP FOR READING RMDB
494 014036 112737 000200 001523     MOV     #200,GETINX+1
495
496 014044 004737 037110          JSR      PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014050 000404          BR       230$          ;GO TO 230$ IF NO ERROR
    014052 000240          NOP      ;RETURN HERE IF ERROR
    014054 104000          EMT     ;ERROR # DEFINED BY GET SUBROUTINE
    014056 000137 014130   JMP      260$          ;GO TO 260$ IF ERROR
497 014062 013737 001356 001142 230$:  MOV     RMDBI,$BDDAT   ;LOAD RECEIVED DATA WORD
498 014070 013737 102210 001140     MOV     BUFTWO-2,$GDDAT;LOAD EXPECTED DATA WORD
499 014076 005137 001140          COM     $GDDAT
500 014102 023737 001134 001136     CMP     $GDADR,$BDADR  ;IS ADDRESS OK??
501 014110 001402          BEQ     220$          ;YES!!
502 014112 104340          EMT     340
503 014114 000405          BR       260$
504 014116 023737 001140 001142 220$:  CMP     $GDDAT,$BDDAT  ;IS DATA WORD OK??
505 014124 001401          BEQ     260$          ;YES!!
506 014126 104341          EMT     341
507 014130          260$:
508
509          ;*****
          ;*TEST 10          FORMAT ONES
          ;*****
          TST10:
    014130          SCOPE          ;SCOPE CALL
    014130 000004          NOP      ;START OF TEST
    014132 000240          MOV     #STACK,SP     ;INITIALIZE STACK POINTER
    014134 012706 001100     MOV     $BASE,R0      ;R0 = UNIBUS ADDRESS
    014140 013700 001276     MOV     TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
    014144 013701 001464     MOV     #10,$TESTN   ;SET TEST NUMBER IN APT MAIL BOX
    014150 012737 000010 001226
510
511          ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
512 014156 012737 000000 001444     MOV     #0,RMDCO      ;CYLINDER = 0
513 014164 012737 000000 001416     MOV     #0,RMDAO      ;TRACK = 0, SECTOR = 0
514 014172 012737 010000 001442     MOV     #FMT16,RMOFO  ;16 BIT FORMAT
  
```

```

T10
515 014200 012737 177376 001412      MOV      #-258.,RMWCO      ;2 + 256 WORDS (2'S COMP)
516 014206 012737 101206 001414      MOV      #BUFONE,RMBAO    ;DATA BUFFER ADDRESS
517 014214 012737 000062 001410      MOV      #WH,RMCS10      ;WRITE HEADER AND DATA
518
519      ;VERIFY THAT SECTOR IS NOT BAD
      014222 004737 034276      JSR      PC,BADSCT      ;CALL BAD SECTOR MODULE
      014226 000405      BR      10$            ;GO TO 10$ IF NO ERROR
      014230 104401 063120      TYPE     ,SCTMSG        ;TYPE BAD SECTOR MESSAGE
      014234 104000      EMT      ;ERROR # DEFINED BY BADSCT SUBROUTINE
      014236 000137 014750      JMP      180$          ;GO TO 180$ IF ERROR
520 014242      10$:
521 014242 012737 064556 001174      MOV      #ONES,$TMPO     ;USE ALL ONES DATA PATTERN
522 014250 012737 000001 001176      MOV      #1,$TMP1
523 014256 004737 036224      JSR      PC,GENBUF      ;GO GENERATE DATA BUFFER
524 014262      20$:
525
526      ;PREPARE DEVICE FOR DATA TRANSFER
527 014262 004737 033352      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      014266 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
      014270 000404      BR      30$            ;GO TO 30$ IF NO ERROR
      014272 000240      NOP
      014274 104000      EMT      ;RETURN HERE IF ERROR
      014276 000137 014750      JMP      180$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      ;GO TO 180$ IF ERROR
528 014302      30$:
529
530      ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
531 014302 012737 000005 001410      MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
532 014310 012702 001551      MOV      #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
533 014314 112722 000006      MOV      #RMDA,(R2)+
534 014320 112722 000034      MOV      #RMDC,(R2)+
535 014324 112722 000032      MOV      #RMOF,(R2)+
536 014330 112722 000000      MOV      #RMCS1,(R2)+
537 014334 112722 000200      MOV      #200,(R2)+
538
539 014340 004737 037360      JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      014344 000404      BR      40$            ;GO TO 40$ IF NO ERROR
      014346 000240      NOP
      014350 104000      EMT      ;RETURN HERE IF ERROR
      014352 000137 014750      JMP      180$          ;ERROR # DEFINED BY PUT SUBROUTINE
      ;GO TO 180$ IF ERROR
540 014356      40$:
541
542      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
543 014356 004737 037024      JSR      PC,GETSTS      ;SETUP FOR STATUS
544 014362 004737 037722      JSR      PC,TIMOUT      ;WAIT FOR SEEK TO COMPLETE
545 014366      50$:
546
547      ;GO READ SEEK STATUS
548 014366 004737 037110      JSR      PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      014372 000404      BR      60$            ;GO TO 60$ IF NO ERROR
      014374 000240      NOP
      ;RETURN HERE IF ERROR

```

```

014376 104000
014400 000137 014750
549 014404
550
551
552 014404 004737 045224
014410 000405
014412 000240
014414 104000
014416 004736
014420 000137 014750
553 014424
554
555
556 014424 012737 000063 001410
557 014432 012702 001554
558 014436 112722 000002
559 014442 112722 000004
560 014446 112722 000000
561 014452 112722 000200
562
563 014456 004737 037360
014462 000404
014464 000240
014466 104000
014470 000137 014750
564 014474
565
566
567 014474 004737 037722
568
569 014500 004737 037110
014504 000404
014506 000240
014510 104000
014512 000137 014750
570 014516
571
572
573 014516 004737 040106
014522 000405
014524 000240
014526 104000
014530 004736
014532 000137 014750
574 014536
575 014536 004737 052622
014542 000405
014544 000240
014546 104000
014550 004736
014552 000137 014750
576 014556
577 014556 004737 040740
014562 000405
014564 000240
014566 104000

      EMT
      JMP      180$      ;ERROR # DEFINED BY GET SUBROUTINE
                        ;GO TO 180$ IF ERROR
60$:
;VERIFY THE RESULTS OF THE SEEK COMMAND
      JSR      PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
      BR       70$      ;GO TO 70$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP      180$      ;GO TO 180$ IF ERROR
70$:
;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
      MOV      #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
      MOVB    #RMWC,(R2)+
      MOVB    #RMBA,(R2)+
      MOVB    #RMCS1,(R2)+
      MOVB    #200,(R2)+
      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      BR       80$      ;GO TO 80$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY PUT SUBROUTINE
      JMP      180$      ;GO TO 180$ IF ERROR
80$:
;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
      JSR      PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
      JSR      PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      BR       90$      ;GO TO 90$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY GET SUBROUTINE
      JMP      180$      ;GO TO 180$ IF ERROR
90$:
;VERIFY RESULTS OF WRITE COMMAND
      JSR      PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      BR       100$     ;GO TO 100$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY PRIERR SUBROUTINE
      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP      180$     ;GO TO 180$ IF ERROR
100$:
      JSR      PC,DTASTS  ;GO VERIFY RESULTS OF DATA TRANSFER
      BR       110$     ;GO TO 110$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR      PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP      180$     ;GO TO 180$ IF ERROR
110$:
      JSR      PC,SECERR  ;GO CHECK FOR SECONDARY ERRORS
      BR       120$     ;GO TO 120$ IF NO ERROR
      NOP      ;RETURN HERE IF ERROR
      EMT      ;ERROR # DEFINED BY SECERR SUBROUTINE
  
```

```

014570 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
014572 000137 014750   JMP      180$          ;GO TO 180$ IF ERROR
578 014576          120$:
579
580          ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
581 014576 012737 000073 001410   MOV      #RH!GO,RMCS10   ;READ HEADER & DATA COMMAND
582 014604 012737 102212 001414   MOV      #BUFTWO,RMBAO   ;CHANGE BUS ADDRESS
583
584 014612 004737 037360          JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
    014616 000404          BR       130$          ;GO TO 130$ IF NO ERROR
    014620 000240          NOP
    014622 104000          EMT          ;RETURN HERE IF ERROR
    014624 000137 014750   JMP      180$          ;ERROR # DEFINED BY PUT SUBROUTINE
585 014630          130$:
586
587          ;WAIT FOR READ TO COMPLETE AND GET STATUS
588 014630 004737 037722          JSR      PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
589
590 014634 004737 037110          JSR      PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
    014640 000404          BR       140$          ;GO TO 140$ IF NO ERROR
    014642 000240          NOP          ;RETURN HERE IF ERROR
    014644 104000          EMT          ;ERROR # DEFINED BY GET SUBROUTINE
    014646 000137 014750   JMP      180$          ;GO TO 180$ IF ERROR
591 014652          140$:
592
593          ;VERIFY THE RESULTS OF READ OPERATION
594 014652 004737 040106          JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
    014656 000405          BR       150$          ;GO TO 150$ IF NO ERROR
    014660 000240          NOP          ;RETURN HERE IF ERROR
    014662 104000          EMT          ;ERROR # DEFINED BY PRIERR SUBROUTINE
    014664 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    014666 000137 014750   JMP      180$          ;GO TO 180$ IF ERROR
595 014672          150$:
596 014672 004737 052622          JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
    014676 000405          BR       160$          ;GO TO 160$ IF NO ERROR
    014700 000240          NGP          ;RETURN HERE IF ERROR
    014702 104000          EMT          ;ERROR # DEFINED BY DTASTS SUBROUTINE
    014704 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    014706 000137 014750   JMP      180$          ;GO TO 180$ IF ERROR
597 014712          160$:
598 014712 004737 040740          JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
    014716 000405          BR       170$          ;GO TO 170$ IF NO ERROR
    014720 000240          NOP          ;RETURN HERE IF ERROR
    014722 104000          EMT          ;ERROR # DEFINED BY SECERR SUBROUTINE
    014724 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
    014726 000137 014750   JMP      180$          ;GO TO 180$ IF ERROR
599 014732          170$:
600
601          ;VERIFY DATA
602 014732 004737 036462          JSR      PC,CMPBUF      ;GO COMPARE WRITE, READ DATA BUFFERS
    014736 101206          .WORD   BUFSIZE        ;STARTING ADDRESS OF WRITE BUFFER
    014740 102212          .WORD   BUFTWO        ;STARTING ADDRESS OF READ BUFFER
    014742 000402          BR       180$          ;GO TO 180$ IF NO ERROR
    014744 000240          NOP          ;RETURN HERE IF ERROR
    014746 104000          EMT          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
603 014750          180$:
604

```

605

014750
014750 000004
014752 000240
014754 012706 001100
014760 013700 001276
014764 013701 001464
014770 012737 000011 001226

```
::*****  
:*TEST 11          FORMAT CHECK ONES  
:*****  
TST11:  
SCOPE                :SCOPE CALL  
NOP                  :START OF TEST  
MOV #STACK,SP        :INITIALIZE STACK POINTER  
MOV $BASE,R0         :R0 = UNIBUS ADDRESS  
MOV TSTQUE,R1        :(R1) = DEVICE BEING TESTED  
MOV #11,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
```

606

607
608 014776 012737 000000 001444
609 015004 012737 000000 001416
610 015012 012737 010000 001442
611 015020 012737 177376 001412
612 015026 012737 101206 001414
613 015034 012737 000062 001410

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

614
615

015042 004737 034276
015046 000405
015050 104401 063120
015054 104000
015056 000137 015540

```
;VERIFY THAT SECTOR IS NOT BAD  
JSR PC,BADSCT       :CALL BAD SECTOR MODULE  
BR 10$              :GO TO 10$ IF NO ERROR  
TYPE ,SCTMSG        :TYPE BAD SECTOR MESSAGE  
EMT                 :ERROR # DEFINED BY BADSCT SUBROUTINE  
JMP 260$            :GO TO 260$ IF ERROR
```

616

617 015062 012737 064556 001174
618 015070 012737 000001 001176
619 015076 004737 036224

```
10$:  
MOV #ONES,$TMP0     :USE ALL ONES DATA PATTERN  
MOV #1,$TMP1  
JSR PC,GENBUF       :GO GENERATE DATA BUFFER
```

620

621

622

623 015102 004737 033352
015106 154130

```
;PREPARE DEVICE FOR DATA TRANSFER  
JSR PC,TSTPRP       :PREPARE DEVICE FOR TEST  
.WORD 154130        :TASK DESCRIPTOR AS FOLLOWS:  
:SELECT DEVICE & VERIFY DEVICE AVAILABLE  
:CLEAR CONTROLLER & SELECT DEVICE  
:VERIFY CONTROLLER CLEAR OPERATION  
:PACK ACKNOWLEDGE IF VOLUME NOT VALID  
:VERIFY PACK ACKNOWLEDGE  
:RECALIBRATE IF "SKI" OR "PIP" IS SET  
:VERIFY RECALIBRATION  
BR 30$              :GO TO 30$ IF NO ERROR  
NOP                 :RETURN HERE IF ERROR  
EMT                 :ERROR # DEFINED BY TSTPRP SUBROUTINE  
JMP 260$            :GO TO 260$ IF ERROR
```

015110 000404

015112 000240

015114 104000

015116 000137 015540

624 015122

625

626

627 015122 012737 000005 001410
628 015130 012702 001551
629 015134 112722 000006
630 015140 112722 000034
631 015144 112722 000032
632 015150 112722 000000
633 015154 112722 000200

```
;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER  
MOV #SEEK!GO,RMCS10 :CHANGE COMMAND TO SEEK  
MOV #PUTINX,R2      :WRITE REGISTER INDEX TABLE  
MOVB #RMDA,(R2)+  
MOVB #RMDC,(R2)+  
MOVB #RMOF,(R2)+  
MOVB #RMCS1,(R2)+  
MOVB #200,(R2)+
```

634

635 015160 004737 037360

```
JSR PC,PUT          :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
```

```

015164 000404 BR 40$ ;GO TO 40$ IF NO ERROR
015166 000240 NOP ;RETURN HERE IF ERROR
015170 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
015172 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
636 015176 40$:
637
638 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
639 015176 004737 037024 JSR PC,GETSTS ;SETUP FOR STATUS
640 015202 004737 037722 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
641 015206 50$:
642
643 ;GO READ SEEK STATUS
644 015206 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
015212 000404 BR 60$ ;GO TO 60$ IF NO ERROR
015214 000240 NOP ;RETURN HERE IF ERROR
015216 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
015220 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
645 015224 60$:
646
647 ;VERIFY THE RESULTS OF THE SEEK COMMAND
648 015224 004737 045224 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
015230 000405 BR 70$ ;GO TO 70$ IF NO ERROR
015232 000240 NOP ;RETURN HERE IF ERROR
015234 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
015236 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015240 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
649 015244 70$:
650
651 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
652 015244 012737 000063 001410 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
653 015252 012702 001554 MOV #PUTINX+3,R2 ;EXTEND REGISTER INDEX TABLE
654 015256 112722 000002 MOVB #RMWC,(R2)+
655 015262 112722 000004 MOVB #RMBA,(R2)+
656 015266 112722 000000 MOVB #RMCS1,(R2)+
657 015272 112722 000200 MOVB #200,(R2)+
658
659 015276 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
015302 000404 BR 80$ ;GO TO 80$ IF NO ERROR
015304 000240 NOP ;RETURN HERE IF ERROR
015306 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
015310 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
660 015314 80$:
661
662 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
663 015314 004737 037722 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
664
665 015320 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
015324 000404 BR 90$ ;GO TO 90$ IF NO ERROR
015326 000240 NOP ;RETURN HERE IF ERROR
015330 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
015332 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
666 015336 90$:
667
668 ;VERIFY RESULTS OF WRITE COMMAND
669 015336 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
015342 000405 BR 100$ ;GO TO 100$ IF NO ERROR
015344 000240 NOP ;RETURN HERE IF ERROR

```

```
015346 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
015350 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE EPROR CHECKS
015352 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
670 015356 100$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
671 015356 004737 052622 BR 110$ ;GO TO 110$ IF NO ERROR
015362 000405 NOP ;RETURN HERE IF ERROR
015364 000240 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
015366 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015370 004736 JMP 260$ ;GO TO 260$ IF ERROR
672 015372 000137 015540 110$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
673 015376 004737 040740 BR 120$ ;GO TO 120$ IF NO ERROR
015402 000405 NOP ;RETURN HERE IF ERROR
015404 000240 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
015406 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015410 004736 JMP 260$ ;GO TO 260$ IF ERROR
674 015412 000137 015540 120$: ;WRITE CHECK HEADER AND DATA FOR SECTOR JUST WRITTEN
675 015416 012737 000053 001410 MOV #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
676 015424 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
677 015430 000404 BR 130$ ;GO TO 130$ IF NO ERROR
678 015432 000240 NOP ;RETURN HERE IF ERROR
679 015434 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
015436 000137 015540 JMP 260$ ;GO TO 260$ IF ERROR
680 015442 130$: ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
681 ;WRITE CHECK COMMAND
682 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
683 015442 004737 037722 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
684 015446 004737 037110 BR 140$ ;GO TO 140$ IF NO ERROR
685 015452 000404 NOP ;RETURN HERE IF ERROR
015454 000240 EMT ;ERROR # DEFINED BY GET SUBROUTINE
015456 104000 JMP 260$ ;GO TO 260$ IF ERROR
686 015460 000137 015540 140$: ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
687 ;GO CHECK FOR PRIMARY ERRORS
688 JSR PC,PRIERR ;GO TO 150$ IF NO ERROR
689 015464 004737 040106 BR 150$ ;RETURN HERE IF ERROR
015470 000405 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
015472 000240 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015474 104000 JMP 260$ ;GO TO 260$ IF ERROR
690 015476 004736 015540 150$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
691 015500 000137 052622 BR 160$ ;GO TO 160$ IF NO ERROR
015504 000405 NOP ;RETURN HERE IF ERROR
015510 000240 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
015512 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
015514 004736 JMP 260$ ;GO TO 260$ IF ERROR
692 015516 000137 015540 160$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
693 015520 004737 040740 BR 170$ ;GO TO 170$ IF NO ERROR
015524 000403 NOP ;RETURN HERE IF ERROR
015530 000240
```

```

015534 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
015536 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
694 015540 170$:
695
696 015540 260$:
697
698
;*****
;*TEST 12 FORMAT CHECK ONES W/ WCE ERRORS
;*****
TST12:
015540 SCOPE ;SCOPE CALL
015540 000004 NOP ;START OF TEST
015542 000240 MOV #STACK,SP ;INITIALIZE STACK POINTER
015544 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
015550 013700 001276 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
015554 013701 001464 MOV #12,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
015560 012737 000012 001226

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
700
701 015566 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
702 015574 012737 000000 001416 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
703 015602 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
704 015610 012737 177376 001412 MOV #-258,RMWCO ;2 + 256 WORDS (2'S (OMP)
705 015616 012737 101206 001414 MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
706 015624 012737 000062 001410 MOV #WH,RMCSI0 ;WRITE HEADER AND DATA
707
708
;VERIFY THAT SECTOR IS NOT BAD
015632 004737 034276 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
015636 000405 BR 10$ ;GO TO 10$ IF NO ERROR
015640 104401 063120 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
015644 104000 EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
015646 000137 016476 JMP 260$ ;GO TO 260$ IF ERROR
709 015652
710 015652 012737 064556 001174 10$: MOV #ONES,$TMP0 ;USE ALL ONES DATA PATTERN
711 015660 012737 000001 001176 MOV #1,$TMP1
712 015666 004737 036224 JSR PC,GENBUF ;GO GENERATE DATA BUFFER
713 015672
714
715
;PREPARE DEVICE FOR DATA TRANSFER
716 015672 004737 033352 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
015676 154130 .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 30$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 260$ IF ERROR

015700 000404 BR 30$
015702 000240 NOP
015704 104000 EMT
015706 000137 016476 JMP 260$
717 015712
718
719
;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
720 015712 012737 000005 001410 MOV #SEEK!GO,RMCSI0 ;CHANGE COMMAND TO SEEK
721 015720 012702 001551 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
722 015724 112722 000006 MOVB #RMDA,(R2)+

```



```

723 015730 112722 000034      MOVB  #RMDC,(R2)+
724 015734 112722 000032      MOVB  #RMOF,(R2)+
725 015740 112722 000000      MOVB  #RMCS1,(R2)+
726 015744 112722 000200      MOVB  #200,(R2)+
727
728 015750 004737 037360      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      015754 000404      BR    40$        ;GO TO 40$ IF NO ERROR
      015756 000240      NOP                    ;RETURN HERE IF ERROR
      015760 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      015762 000137 016476      JMP   260$       ;GO TO 260$ IF ERROR
729 015766
730
731      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
732 015766 004737 037024      JSR   PC,GETSTS   ;SETUP FOR STATUS
733 015772 004737 037722      JSR   PC,TIMOUT   ;WAIT FOR SEEK TO COMPLETE
734 015776
735
736      ;GO READ SEEK STATUS
737 015776 004737 037110      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      016002 000404      BR    60$        ;GO TO 60$ IF NO ERROR
      016004 000240      NOP                    ;RETURN HERE IF ERROR
      016006 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      016010 000137 016476      JMP   260$       ;GO TO 260$ IF ERROR
738 016014
739
740      ;VERIFY THE RESULTS OF THE SEEK COMMAND
741 016014 004737 045224      JSR   PC,SEKSTS   ;GO VERIFY RESULTS OF SEEK OPERATION
      016020 000405      BR    70$        ;GO TO 70$ IF NO ERROR
      016022 000240      NOP                    ;RETURN HERE IF ERROR
      016024 104000      EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      016026 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      016030 000137 016476      JMP   260$       ;GO TO 260$ IF ERROR
742 016034
743
744      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
745 016034 012737 000063 001410      MOV   #WH!GO,RMCS10 ;WRITE HEADER AND DATA
746 016042 012702 001554      MOV   #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
747 016046 112722 000002      MOVB  #RMWC,(R2)+
748 016052 112722 000004      MOVB  #RMBA,(R2)+
749 016056 112722 000000      MOVB  #RMCS1,(R2)+
750 016062 112722 000200      MOVB  #200,(R2)+
751
752 016066 004737 037360      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      016072 000404      BR    80$        ;GO TO 80$ IF NO ERROR
      016074 000240      NOP                    ;RETURN HERE IF ERROR
      016076 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      016100 000137 016476      JMP   260$       ;GO TO 260$ IF ERROR
753 016104
754
755      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
756 016104 004737 037722      JSR   PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
757
758 016110 004737 037110      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      016114 000404      BR    90$        ;GO TO 90$ IF NO ERROR
      016116 000240      NOP                    ;RETURN HERE IF ERROR
      016120 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      016122 000137 016476      JMP   260$       ;GO TO 260$ IF ERROR

```

```

759 016126          90$:
760
761                ;VERIFY RESULTS OF WRITE COMMAND
762 016126 004737 040106      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
                                BR      100$      ;GO TO 100$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY PRIERR SUBROUTINE
763 016142 000137 016476      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP     260$      ;GO TO 260$ IF ERROR
764 016146          100$:
765 016146 004737 052622      JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR      110$      ;GO TO 110$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY DTASTS SUBROUTINE
766 016152 000405
767 016154 000240
768 016156 104000
769 016160 004736
770 016162 000137 016476      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP     260$      ;GO TO 260$ IF ERROR
771 016166          110$:
772 016166 004737 040740      JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
                                BR      120$      ;GO TO 120$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY SECERR SUBROUTINE
773 016172 000405
774 016174 000240
775 016176 104000
776 016200 004736
777 016202 000137 016476      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP     260$      ;GO TO 260$ IF ERROR
778 016206          120$:
779 016206 005137 102210      ;ALTER DATA BUFFER
                                COM     BUFTWO-2      ;COMPLEMENT DATA WORD
780 016212 012737 000053 001410 ;SETUP AND WRITE CHECK HEADER AND DATA COMMAND
781 016212 012737 000053 001410 MOV     #WCH!GO,RMCSTO ;WRITE CHECK COMMAND
782 016220 004737 037360      JSR      PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      180$      ;GO TO 180$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY PUT SUBROUTINE
783 016224 000404
784 016226 000240
785 016230 104000
786 016232 000137 016476      JSR      PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR      180$      ;GO TO 180$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY PUT SUBROUTINE
787 016236          180$:
788 016236 004737 037722      ;WAIT FOR WRITE CHECK COMMAND TO COMPLETE AND READ STATUS
789 016236 004737 037722      JSR      PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
790 016242 004737 037110      JSR      PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR      190$      ;GO TO 190$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY GET SUBROUTINE
791 016246 000404
792 016250 000240
793 016252 104000
794 016254 000137 016476      JSR      PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR      190$      ;GO TO 190$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY GET SUBROUTINE
795 016260          190$:
796 016260 004737 040106      ;CHECK FOR PRIMARY ERRORS
797 016260 004737 040106      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
                                BR      200$      ;GO TO 200$ IF NO ERROR
                                NOP                     ;RETURN HERE IF ERROR
                                EMT                     ;ERROR # DEFINED BY PRIERR SUBROUTINE
798 016264 000405
799 016266 000240
800 016270 104000
801 016272 004736
802 016274 000137 016476      JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
                                JMP     260$      ;GO TO 260$ IF ERROR
803 016300          200$:
804
805

```

```

788 ;MAKE SURE THE WRITE CHECK ERROR WAS DETECTED
789 016300 032737 040000 001344 BIT #WCE,RMCS2I ;IS WRITE CHECK ERROR SET??
790 016306 001022 BNE 210$ ;YES!!
791
792 016310 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
016314 000405 BR 205$ ;GO TO 205$ IF NO ERROR
016316 000240 NOP ;RETURN HERE IF ERROR
016320 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
016322 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
016324 000137 016476 JMP 260$ ;GO TO 260$ IF ERROR
793 016330 013737 001344 001140 205$: MOV RMCS2I,$GDDAT ;LOAD EXPECTED STATUS
794 016336 052737 040000 001140 BIS #WCE,$GDDAT
795 016344 013737 001344 001142 MOV RMCS2I,$BDDAT ;LOAD RECEIVED STATUS
796 016352 104337 EMT 337
797 016354 210$:
798
799 ;VERIFY THE ADDRESS OF THE WRITE CHECK ERROR
800 016354 012737 102210 001134 MOV #BUFTWO-2,$GDADR ;LOAD EXPECTED ADDRESS
801 016362 013737 001340 001136 MOV RMBAI,$BDADR ;LOAD RECEIVED ADDRESS
802 016370 162737 000002 001136 SUB #2,$BDADR ;DECREMENT RECEIVED ADDRESS
803
804 ;GET WCE DATA AND VERIFY IT IS OK
805 016376 112737 000022 001522 MOVB #RMDB,GETINX ;SETUP FOR READING RMDB
806 016404 112737 000200 001523 MOVB #200,GETINX+1
807
808 016412 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
016416 000404 BR 230$ ;GO TO 230$ IF NO ERROR
016420 000240 NOP ;RETURN HERE IF ERROR
016422 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
016424 000137 016476 JMP 260$ ;GO TO 260$ IF ERROR
809 016430 013737 001356 001142 230$: MOV RMDB1,$BDDAT ;LOAD RECEIVED DATA WORD
810 016436 013737 102210 001140 MOV BUFTWO-2,$GDDAT ;LOAD EXPECTED DATA WORD
811 016444 005137 001140 COM $GDDAT
812 016450 023737 001134 001136 CMP $GDADR,$BDADR ;IS ADDRESS OK??
813 016456 001402 BEQ 220$ ;YES!!
814 016460 104340 EMT 340
815 016462 000405 BR 260$
816 016464
817 016464 023737 001140 001142 220$: CMP $GDDAT,$BDDAT ;IS DATA WORD OK??
818 016472 001401 BEQ 260$ ;YES!!
819 016474 104341 EMT 341
820 016476 260$:
821
822 ;*****
; *TEST 13 FORMAT MULTIPLE SECTORS
;*****
TST13:
016476 000004 SCOPE ;SCOPE CALL
016500 000240 NOP ;START OF TEST
016502 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
016506 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
016512 013701 001464 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
016516 012737 000013 001226 MOV #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
823
824 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
825 016524 012737 010000 001442 MOV #FMT16,RMOFO ;16 BIT FORMAT
826 016532 012737 000000 001444 MOV #0,RMDCO ;CYLINDER = 0
  
```

```

827 016540 012737 000000 001416      MOV      #0,RMDAQ      ;TRACK = 0, SECTOR = 0
828 016546 012737 176774 001412      MOV      #-258.*2,RMWC0 ;WORD COUNT FOR 2 SECTORS (2'S COMP)
829 016554 012737 101206 001414      MOV      #BUFONE,RMBA0 ;DATA BUFFER ADDRESS
830 016562 012737 000062 001410      MOV      #WH,RMCS10    ;WRITE HEADER AND DATA
831
832          ;VERIFY THAT SECTOR IS NOT BAD
      016570 004737 034276      JSR      PC,BADSCT    ;CALL BAD SECTOR MODULE
      016574 000405              BR      10$          ;GO TO 10$ IF NO ERROR
      016576 104401 063120      TYPE     ,SCTMSG     ;TYPE BAD SECTOR MESSAGE
      016602 104000              EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
      016604 000137 017272      JMP      170$        ;GO TO 170$ IF ERROR
833 016610          10$:
834 016610 012737 064620 001174      MOV      #ZEROS,$TMPO ;USE ALL ZEROS DATA PATTERN
835 016616 012737 000001 001176      MOV      #1,$TMP1
836 016624 004737 036224          JSR      PC,GENBUF   ;GO GENERATE DATA BUFFER
837 016630          20$:
838
839          ;PREPARE DEVICE FOR DATA TRANSFER
840 016630 004737 033352      JSR      PC,TSTPRP  ;PREPARE DEVICE FOR TEST
      016634 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
      016636 000404          BR      30$          ;GO TO 30$ IF NO ERROR
      016640 000240          NOP
      016642 104000              EMT
      016644 000137 017272      JMP      180$        ;RETURN HERE IF ERROR
                              ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                              ;GO TO 180$ IF ERROR
841 016650          30$:
842
843          ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
844 016650 012737 000005 001410      MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
845 016656 012702 001551              MOV      #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
846 016662 112722 000006      MOVB    #RMDA,(R2)+
847 016666 112722 000034      MOVB    #RMDC,(R2)+
848 016672 112722 000032      MOVB    #RMOF,(R2)+
849 016676 112722 000000      MOVB    #RMCS1,(R2)+
850 016702 112722 000200      MOVB    #200,(R2)+
851
852 016706 004737 037360      JSR      PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      016712 000404          BR      40$          ;GO TO 40$ IF NO ERROR
      016714 000240          NOP
      016716 104000              EMT
      016720 000137 017272      JMP      180$        ;RETURN HERE IF ERROR
                              ;ERROR # DEFINED BY PUT SUBROUTINE
                              ;GO TO 180$ IF ERROR
853 016724          40$:
854
855          ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
856 016724 004737 037024      JSR      PC,GETSTS   ;SETUP FOR STATUS
857 016730 004737 037722      JSR      PC,TIMOUT  ;WAIT FOR SEEK TO COMPLETE
858 016734          50$:
859
860          ;GO READ SEEK STATUS
861 016734 004737 037110      JSR      PC,GET     ;GO READ REGISTER(S) WITH GET SUBROUTINE
      016740 000404          BR      60$          ;GO TO 60$ IF NO ERROR

```

```

016742 000240      NOP
016744 104000      EMT
016746 000137 017272 60$: JMP 180$      ;RETURN HERE IF ERROR
016752      ;ERROR # DEFINED BY GET SUBROUTINE
                        ;GO TO 180$ IF ERROR
;VERIFY THE RESULTS OF THE SEEK COMMAND
016752 004737 045224 JSR PC,SEKSTS    ;GO VERIFY RESULTS OF SEEK OPERATION
016756 000405      BR 70$          ;GO TO 70$ IF NO ERROR
016760 000240      NOP            ;RETURN HERE IF ERROR
016762 104000      EMT            ;ERROR # DEFINED BY SEKSTS SUBROUTINE
016764 004736      JSR PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
016766 000137 017272 70$: JMP 180$          ;GO TO 180$ IF ERROR
016772      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
016772 012737 000063 001410 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
017000 012702 001554 MOV #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
017004 112722 000002 MOVB #RMWC,(R2)+
017010 112722 000004 MOVB #RMBA,(R2)+
017014 112722 000000 MOVB #RMCS1,(R2)+
017020 112722 000200 MOVB #200,(R2)+
017024 004737 037360 JSR PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
017030 000404      BR 80$          ;GO TO 80$ IF NO ERROR
017032 000240      NOP            ;RETURN HERE IF ERROR
017034 104000      EMT            ;ERROR # DEFINED BY PUT SUBROUTINE
017036 000137 017272 80$: JMP 180$          ;GO TO 180$ IF ERROR
017042      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
017042 004737 037722 JSR PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
017046 004737 037110 JSR PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
017052 000404      BR 90$          ;GO TO 90$ IF NO ERROR
017054 000240      NOP            ;RETURN HERE IF ERROR
017056 104000      EMT            ;ERROR # DEFINED BY GET SUBROUTINE
017060 000137 017272 90$: JMP 180$          ;GO TO 180$ IF ERROR
017064      ;VERIFY RESULTS OF WRITE COMMAND
017064 004737 040106 JSR PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
017070 000405      BR 100$         ;GO TO 100$ IF NO ERROR
017072 000240      NOP            ;RETURN HERE IF ERROR
017074 104000      EMT            ;ERROR # DEFINED BY PRIERR SUBROUTINE
017076 004736      JSR PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
017100 000137 017272 100$: JMP 180$          ;GO TO 180$ IF ERROR
017104      ;GO VERIFY RESULTS OF DATA TRANSFER
017104 004737 052622 JSR PC,DTASTS   ;GO TO 110$ IF NO ERROR
017110 000405      BR 110$         ;RETURN HERE IF ERROR
017112 000240      NOP            ;ERROR # DEFINED BY DTASTS SUBROUTINE
017114 104000      EMT            ;GO BACK FOR MORE ERROR CHECKS
017116 004736      JSR PC,@(SP)+    ;GO TO 180$ IF ERROR
017120 000137 017272 110$: JMP 180$          ;GO TO 180$ IF ERROR
017124      ;GO CHECK FOR SECONDARY ERRORS
017124 004737 040740 JSR PC,SECERR   ;GO TO 120$ IF NO ERROR
017130 000405      BR 120$         ;RETURN HERE IF ERROR
017132 000240      NOP

```

```

017134 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
017136 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017140 000137 017272 JMP 180$ ;GO TO 180$ IF ERROR
891 017144 120$:
892
893 ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
894 017144 012737 000053 001410 MOV #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
895
896 017152 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
017156 000404 BR 130$ ;GO TO 130$ IF NO ERROR
017160 000240 NOP ;RETURN HERE IF ERROR
017162 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
017164 000137 017272 JMP 180$ ;GO TO 180$ IF ERROR
897 017170 130$:
898
899 ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
900 017170 004737 037722 JSR PC,TIMOUT ;WAIT FOR WRITE CHECK TO FINISH
901
902 017174 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
017200 000404 BR 140$ ;GO TO 140$ IF NO ERROR
017202 000240 NOP ;RETURN HERE IF ERROR
017204 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
017206 000137 017272 JMP 180$ ;GO TO 180$ IF ERROR
903 017212 140$:
904
905 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
906 017212 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
017216 000405 BR 150$ ;GO TO 150$ IF NO ERROR
017220 000240 NOP ;RETURN HERE IF ERROR
017222 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
017224 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017226 000137 017272 JMP 180$ ;GO TO 180$ IF ERROR
907 017232 150$:
908 017232 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
017236 000405 BR 160$ ;GO TO 160$ IF NO ERROR
017240 000240 NOP ;RETURN HERE IF ERROR
017242 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
017244 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017246 000137 017272 JMP 180$ ;GO TO 180$ IF ERROR
909 017252 160$:
910 017252 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
017256 000405 BR 170$ ;GO TO 170$ IF NO ERROR
017260 000240 NOP ;RETURN HERE IF ERROR
017262 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
017264 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017266 000137 017272 JMP 180$ ;GO TO 180$ IF ERROR
911 017272 170$:
912
913 017272 180$:
914
915
*****
*TEST 14 FORMAT W/ HEAD SWITCHING
*****
TST14:
017272 000004 SCOPE ;SCOPE CALL
017274 000240 NOP ;START OF TEST
017276 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER

```

```

017302 013700 001276      MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
017306 013701 001464      MOV      TSTQUE,R1         ;(R1) = DEVICE BEING TESTED
017312 012737 000014 001226  MOV      #14,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX

916
917
918 017320 012737 000000 001444 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
919 017326 112737 000000 001417      MOV      #0,RMDCO          ;CYLINDER = 0
920 017334 112737 000037 001416 5$:  MOVB     #0,RMDAO+1        ;TRACK = 0
921 017342 012737 010000 001416      MOVB     #31,RMDAO         ;SECTOR = 31.
922 017350 012737 176774 001442      MOV      #FMT16,RMOFO      ;16 BIT FORMAT
923 017356 012737 101206 001412      MOV      #-258,*2,RMWCO    ;WORD COUNT FOR 2 SECTORS (2'S COMP)
924 017364 012737 000062 001414      MOV      #BUFONE,RMBAO     ;DATA BUFFER ADDRESS
925
926      MOV      #WH,RMCS10     ;WRITE HEADER AND DATA

017372 004737 034276      ;VERIFY THAT SECTOR IS NOT BAD
017376 000405      JSR      PC,BADSCT         ;CALL BAD SECTOR MODULE
017400 104401 063120      BR       10$              ;GO TO 10$ IF NO ERROR
017404 104000      TYPE    ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
017406 000137 020104      EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
927 017412      JMP      180$            ;GO TO 180$ IF ERROR

928 017412 123727 001416 000037 10$:  CMPB     RMDAO,#31.        ;IS LAST SECTOR ASSIGNED ?
929 017420 001345      BNE     5$                ;BR IF NO
930 017422 012737 064620 001174      MOV      #ZEROS,$TMP0     ;USE ALL ZEROS DATA PATTERN
931 017430 012737 000001 001176      MOV      #1,$TMP1
932 017436 004737 036224      JSR      PC,GENBUF        ;GO GENERATE DATA BUFFER
933
934
935 017442 004737 033352      ;PREPARE DEVICE FOR DATA TRANSFER
017446 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 30$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 180$ IF ERROR

017450 000404      BR       30$
017452 000240      NOP
017454 104000      EMT
017456 000137 020104      JMP      180$
936 017462 30$:
937
938
939 017462 012737 000005 001410 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
940 017470 012702 001551      MOV      #SEEK!GO,RMCS10  ;CHANGE COMMAND TO SEEK
941 017474 112722 000006      MOV      #PUTINX,R2       ;WRITE REGISTER INDEX TABLE
942 017500 112722 000034      MOVB     #RMDA,(R2)+
943 017504 112722 000032      MOVB     #RMDC,(R2)+
944 017510 112722 000000      MOVB     #RMOF,(R2)+
945 017514 112722 000200      MOVB     #RMCS1,(R2)+
946
947 017520 004737 037360      MOVB     #200,(R2)+
                                JSR      PC,PUT           ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
017524 000404      BR       40$            ;GO TO 40$ IF NO ERROR
017526 000240      NOP
017530 104000      EMT
017532 000137 020104      JMP      180$            ;ERROR # DEFINED BY PUT SUBROUTINE
948 017536 40$:

```

```

949
950 ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
951 017536 004737 037024 JSR PC,GETSTS ;SETUP FOR STATUS
952 017542 004737 037722 JSR PC,TIMOUT ;WAIT FOR SEEK TO COMPLETE
953 017546 50$:
954
955 ;GO READ SEEK STATUS
956 017546 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
017552 000404 BR 60$ ;GO TO 60$ IF NO ERROR
017554 000240 NOP ;RETURN HERE IF ERROR
017556 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
017560 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
957 017564 60$:
958
959 ;VERIFY THE RESULTS OF THE SEEK COMMAND
960 017564 004737 045224 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
017570 000405 BR 70$ ;GO TO 70$ IF NO ERROR
017572 000240 NOP ;RETURN HERE IF ERROR
017574 104000 EMT ;ERROR # DEFINED BY SEKSTS SUBROUTINE
017576 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017600 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
961 017604 70$:
962
963 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
964 017604 012737 000063 001410 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
965 017612 012702 001554 MOV #PUTINX+3,R2 ;EXTEND REGISTER INDEX TABLE
966 017616 112722 000002 MOVB #RMWC,(R2)+
967 017622 112722 000004 MOVB #RMBA,(R2)+
968 017626 112722 000000 MOVB #RMCS1,(R2)+
969 017632 112722 000200 MOVB #200,(R2)+
970
971 017636 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
017642 000404 BR 80$ ;GO TO 80$ IF NO ERROR
017644 000240 NOP ;RETURN HERE IF ERROR
017646 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
017650 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
972 017654 80$:
973
974 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
975 017654 004737 037722 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
976
977 017660 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
017664 000404 BR 90$ ;GO TO 90$ IF NO ERROR
017666 000240 NOP ;RETURN HERE IF ERROR
017670 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
017672 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
978 017676 90$:
979
980 ;VERIFY RESULTS OF WRITE COMMAND
981 017676 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
017702 000405 BR 100$ ;GO TO 100$ IF NO ERROR
017704 000240 NOP ;RETURN HERE IF ERROR
017706 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
017710 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017712 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
982 017716 100$:
983 017716 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
  
```



```

017722 000405 BR 110$ ;GO TO 110$ IF NO ERROR
017724 000240 NOP ;RETURN HERE IF ERROR
017726 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
017730 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017732 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
984 017736 110$:
985 017736 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
017742 000405 BR 120$ ;GO TO 120$ IF NO ERROR
017744 000240 NOP ;RETURN HERE IF ERROR
017746 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
017750 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
017752 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
986 017756 120$:
987
988 ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
989 017756 012737 000053 001410 MOV #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
990
991 017764 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
017770 000404 BR 130$ ;GO TO 130$ IF NO ERROR
017772 000240 NOP ;RETURN HERE IF ERROR
017774 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
017776 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
992 020002 130$:
993
994 ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
995 020002 004737 037722 JSR PC,TIMOUT ;WAIT FOR WRITE CHECK TO FINISH
996
997 020006 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
020012 000404 BR 140$ ;GO TO 140$ IF NO ERROR
020014 000240 NOP ;RETURN HERE IF ERROR
020016 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
020020 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
998 020024 140$:
999
1000 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
1001 020024 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
020030 000405 BR 150$ ;GO TO 150$ IF NO ERROR
020032 000240 NOP ;RETURN HERE IF ERROR
020034 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
020036 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
020040 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
1002 020044 150$:
1003 020044 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
020050 000405 BR 160$ ;GO TO 160$ IF NO ERROR
020052 000240 NOP ;RETURN HERE IF ERROR
020054 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
020056 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
020060 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
1004 020064 160$:
1005 020064 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
020070 000405 BR 170$ ;GO TO 170$ IF NO ERROR
020072 000240 NOP ;RETURN HERE IF ERROR
020074 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
020076 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
020100 000137 020104 JMP 180$ ;GO TO 180$ IF ERROR
1006 020104 170$:
1007

```

```

1008 020104
1009
1010
    020104
    020104 000004
    020106 000240
    020110 012706 001100
    020114 013700 001276
    020120 013701 001464
    020124 012737 000015 001226
1011
1012
1013 020132 012737 000000 001444
1014 020140 013737 001332 001416
1015 020146 112737 000037 001416
1016 020154 012737 010000 001442
1017 020162 012737 176774 001412
1018 020170 012737 101206 001414
1019 020176 012737 000062 001410
1020
1021
    020204 004737 034276
    020210 000405
    020212 104401 063120
    020216 104000
    020220 000137 020716
1022 020224
1023 020224 123737 001417 001333
1024 020232 001342
1025 020234 012737 064620 001174
1026 020242 012737 000001 001176
1027 020250 004737 036224
1028 020254
1029
1030
1031 020254 004737 033352
    020260 154130
    020262 000404
    020264 000240
    020266 104000
    020270 000137 020716
1032 020274
1033
1034
1035 020274 012737 000005 001410
1036 020302 012702 001551
1037 020306 112722 000006
1038 020312 112722 000034
  
```

```

180$:
*****
*TEST 15      FORMAT W/ MID TRANSFER SEEK
*****
TST15:
    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

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
    MOV #0,RMDCO         ;CYLINDER = 0
5$:  MOV LSTRK,RMDAO      ;START LAST TRACK AND
    MOVB #31,RMDAO       ;LAST SECTOR
    MOV #FMT16,RMOFO     ;16 BIT FORMAT
    MOV #-258.*2,RMWCO   ;WORD COUNT FOR 2 SECTORS (2'S COMP)
    MOV #BUFONE,RMBAO    ;DATA BUFFER ADDRESS
    MOV #WH,RMCS10       ;WRITE HEADER AND DATA

;VERIFY THAT SECTOR IS NOT BAD
    JSR PC,BADSCT        ;CALL BAD SECTOR MODULE
    BR 10$               ;GO TO 10$ IF NO ERROR
    TYPE ,SCTMSG         ;TYPE BAD SECTOR MESSAGE
    EMT                  ;ERROR # DEFINED BY BADSCT SUBROUTINE
    JMP 180$             ;GO TO 180$ IF ERROR
10$:  CMPB RMDAO+1,LSTRK+1 ;IS LAST TRACK ASSIGNED ?
    BNE 5$               ;BR IF NO
    MOV #ZEROS,$TMP0     ;USE ALL ZEROS DATA PATTERN
    MOV #1,$TMP1
    JSR PC,GENBUF        ;GO GENERATE DATA BUFFER
20$:

;PREPARE DEVICE FOR DATA TRANSFER
    JSR PC,TSTPRP        ;PREPARE DEVICE FOR TEST
    .WORD 154130         ;TASK DESCRIPTOR AS FOLLOWS:
                        ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                        ;CLEAR CONTROLLER & SELECT DEVICE
                        ;VERIFY CONTROLLER CLEAR OPERATION
                        ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                        ;VERIFY PACK ACKNOWLEDGE
                        ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                        ;VERIFY RECALIBRATION
    BR 30$               ;GO TO 30$ IF NO ERROR
    NOP                  ;RETURN HERE IF ERROR
    EMT                  ;ERROR # DEFINED BY TSTPRP SUBROUTINE
    JMP 180$             ;GO TO 180$ IF ERROR
30$:

;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE ON CYLINDER
    MOV #SEEK!GO,RMCS10  ;CHANGE COMMAND TO SEEK
    MOV #PUTINX,R2       ;WRITE REGISTER INDEX TABLE
    MOVB #RMDA,(R2)+
    MOVB #RMDC,(R2)+
  
```

```

1039 020316 112722 000032      MOVB  #RMOF,(R2)+
1040 020322 112722 000000      MOVB  #RMCS1,(R2)+
1041 020326 112722 000200      MOVB  #200,(R2)+
1042
1043 020332 004737 037360      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020336 000404      BR    40$            ;GO TO 40$ IF NO ERROR
      020340 000240      NOP                    ;RETURN HERE IF ERROR
      020342 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      020344 000137 020716      JMP   180$           ;GO TO 180$ IF ERROR
1044 020350      40$:
1045
1046      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
1047 020350 004737 037024      JSR   PC,GETSTS      ;SETUP FOR STATUS
1048 020354 004737 037722      JSR   PC,TIMOUT     ;WAIT FOR SEEK TO COMPLETE
1049 020360      50$:
1050
1051      ;GO READ SEEK STATUS
1052 020360 004737 037110      JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020364 000404      BR    60$            ;GO TO 60$ IF NO ERROR
      020366 000240      NOP                    ;RETURN HERE IF ERROR
      020370 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      020372 000137 020716      JMP   180$           ;GO TO 180$ IF ERROR
1053 020376      60$:
1054
1055      ;VERIFY THE RESULTS OF THE SEEK COMMAND
1056 020376 004737 045224      JSR   PC,SEKSTS     ;GO VERIFY RESULTS OF SEEK OPERATION
      020402 000405      BR    70$            ;GO TO 70$ IF NO ERROR
      020404 000240      NOP                    ;RETURN HERE IF ERROR
      020406 104000      EMT                    ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      020410 004736      JSR   PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      020412 000137 020716      JMP   180$           ;GO TO 180$ IF ERROR
1057 020416      70$:
1058
1059      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1060 020416 012737 000063 001410      MOV   #WH!GO,RMCS10 ;WRITE HEADER AND DATA
1061 020424 012702 001554      MOV   #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
1062 020430 112722 000002      MOVB  #RMWC,(R2)+
1063 020434 112722 000004      MOVB  #RMBA,(R2)+
1064 020440 112722 000000      MOVB  #RMCS1,(R2)+
1065 020444 112722 000200      MOVB  #200,(R2)+
1066
1067 020450 004737 037360      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020454 000404      BR    80$            ;GO TO 80$ IF NO ERROR
      020456 000240      NOP                    ;RETURN HERE IF ERROR
      020460 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      020462 000137 020716      JMP   180$           ;GO TO 180$ IF ERROR
1068 020466      80$:
1069
1070      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
1071 020466 004737 037722      JSR   PC,TIMOUT     ;WAIT FOR COMMAND TO COMPLETE
1072
1073 020472 004737 037110      JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020476 000404      BR    90$            ;GO TO 90$ IF NO ERROR
      020500 000240      NOP                    ;RETURN HERE IF ERROR
      020502 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      020504 000137 020716      JMP   180$           ;GO TO 180$ IF ERROR
1074 020510      90$:

```

```

1075
1076
1077 020510 004737 040106      ;VERIFY RESULTS OF WRITE COMMAND
      020514 000405           JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      020516 000240           BR    100$          ;GO TO 100$ IF NO ERROR
      020520 104000           NOP                    ;RETURN HERE IF ERROR
      020522 004736           EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      020524 000137 020716     JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      1078 020530           JMP   180$          ;GO TO 180$ IF ERROR
      1079 020530 004737 052622 100$:
      020534 000405           JSR   PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      020536 000240           BR    110$          ;GO TO 110$ IF NO ERROR
      020540 104000           NOP                    ;RETURN HERE IF ERROR
      020542 004736           EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      020544 000137 020716     JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      1080 020550           JMP   180$          ;GO TO 180$ IF ERROR
      1081 020550 004737 040740 110$:
      020554 000405           JSR   PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      020556 000240           BR    120$          ;GO TO 120$ IF NO ERROR
      020560 104000           NOP                    ;RETURN HERE IF ERROR
      020562 004736           EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      020564 000137 020716     JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      1082 020570           JMP   180$          ;GO TO 180$ IF ERROR
      1083
      1084
      1085 020570 012737 000053 001410 ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
      1086 020576 004737 037360      MOV   #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
      1087 020602 000404           JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      020604 000240           BR    130$          ;GO TO 130$ IF NO ERROR
      020606 104000           NOP                    ;RETURN HERE IF ERROR
      020610 000137 020716     EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      1088 020614           JMP   180$          ;GO TO 180$ IF ERROR
      1089
      1090
      1091 020614 004737 037722 130$:
      1092 020620 004737 037110      ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
      1093 020624 000404           JSR   PC,TIMOUT     ;WAIT FOR WRITE CHECK TO FINISH
      020626 000240           JSR   PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      020630 104000           BR    140$          ;GO TO 140$ IF NO ERROR
      020632 000137 020716     NOP                    ;RETURN HERE IF ERROR
      1094 020636           EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      1095
      1096
      1097 020636 004737 040106      ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
      020642 000405           JSR   PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
      020644 000240           BR    150$          ;GO TO 150$ IF NO ERROR
      020646 104000           NOP                    ;RETURN HERE IF ERROR
      020650 004736           EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      020652 000137 020716     JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      1098 020656           JMP   180$          ;GO TO 180$ IF ERROR
      1099 020656 004737 052622 150$:
      020662 000405           JSR   PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
      020664 000240           BR    160$          ;GO TO 160$ IF NO ERROR
      020666 104000           NOP                    ;RETURN HERE IF ERROR
      020670 004736           EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS

```

```

1100 020672 000137 020716          JMP      180$          ;GO TO 180$ IF ERROR
1101 020676 004737 040740          JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
                                BR        170$          ;GO TO 170$ IF NO ERROR
                                NOP                       ;RETURN HERE IF ERROR
                                EMT                       ;ERROR # DEFINED BY SECERR SUBROUTINE
                                JSR      PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
                                JMP      180$          ;GO TO 180$ IF ERROR
1102 020716          170$:
1103
1104 020716          180$:
1105
1106          ;*****
          ;*TEST 16          FORMAT W/ IMPLIED SEEK
          ;*****
          TST16:
          SCOPE          ;SCOPE CALL
          NOP            ;START OF TEST
          MOV      #STACK,SP ;INITIALIZE STACK POINTER
          MOV      $BASE,R0 ;R0 = UNIBUS ADDRESS
          MOV      TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
          MOV      #16,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

          ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
          MOV      #0,RMDCO ;CYLINDER = 0
          MOV      #0,RMDAO ;TRACK = 0, SECTOR = 0
          MOV      #FMT16,RMOFO ;16 BIT FORMAT
          MOV      #-258.*2,RMWCO ;WORD COUNT FOR 2 SECTORS (2'S COMP)
          MOV      #BUFONE,RMBAO ;DATA BUFFER ADDRESS
          MOV      #WH,RMCS10 ;WRITE HEADER AND DATA

          ;VERIFY THAT SECTOR IS NOT BAD
          JSR      PC,BADSCT ;CALL BAD SECTOR MODULE
          BR        10$          ;GO TO 10$ IF NO ERROR
          TYPE      ,SCTMSG    ;TYPE BAD SECTOR MESSAGE
          EMT                       ;ERROR # DEFINED BY BADSCT SUBROUTINE
          JMP      180$          ;GO TO 180$ IF ERROR
1117 021030          10$:
1118 021030 012737 064620 001174    MOV      #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
1119 021036 012737 000001 001176    MOV      #1,$TMP1
1120 021044 004737 036224          JSR      PC,GENBUF    ;GO GENERATE DATA BUFFER
1121 021050          20$:
1122
1123          ;PREPARE DEVICE FOR DATA TRANSFER
1124 021050 004737 033352          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
          021054 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        30$          ;GO TO 30$ IF NO ERROR
          021056 000404          NOP                       ;RETURN HERE IF ERROR
          021060 000240          EMT                       ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          021062 104000          JMP      180$          ;GO TO 180$ IF ERROR
          021064 000137 021534
  
```

```

1125 021070          30$:
1126
1127
1128 021070 013737 001444 021536 ;SETUP PARAMETERS AND EXECUTE SEEK TO GET DRIVE OFF CYLINDER
1129 021076 012737 001466 001444      MOV      RMDCO,190$      ;SAVE CYLINDER ADDRESS
1130 021104 012737 000005 001410      MOV      #822.,RMDCO    ;SEEK TO LAST CYLINDER
1131 021112 012702 001551      MOV      #SEEK!GO,RMCS10 ;CHANGE COMMAND TO SEEK
1132 021116 112722 000006      MOV      #PUTINX,R2     ;WRITE REGISTER INDEX TABLE
1133 021122 112722 000034      MOVB     #RMDA,(R2)+
1134 021126 112722 000032      MOVB     #RMDC,(R2)+
1135 021132 112722 000000      MOVB     #RMOF,(R2)+
1136 021136 112722 000200      MOVB     #RMCS1,(R2)+
1137
1138 021142 004737 037360      JSR      PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021146 000404      BR       40$           ;GO TO 40$ IF NO ERROR
      021150 000240      NOP
      021152 104000      EMT
      021154 000137 021534      JMP      180$          ;ERROR # DEFINED BY PUT SUBROUTINE
                          ;GO TO 180$ IF ERROR
1139 021160          40$:
1140
1141
1142 021160 004737 037024      ;SETUP FOR READING STATUS AND THEN WAIT FOR SEEK TO COMPLETE
1143 021164 004737 037722      JSR      PC,GETSTS     ;SETUP FOR STATUS
1144 021170          JSR      PC,TIMOUT     ;WAIT FOR SEEK TO COMPLETE
1145
1146          50$:
1147 021170 004737 037110      ;GO READ SEEK STATUS
      021174 000404      JSR      PC,GET       ;GO READ REGISTER(S) WITH GET SUBROUTINE
      021176 000240      BR       60$           ;GO TO 60$ IF NO ERROR
      021200 104000      NOP                   ;RETURN HERE IF ERROR
      021202 000137 021534      EMT                   ;ERROR # DEFINED BY GET SUBROUTINE
1148 021206          JMP      180$          ;GO TO 180$ IF ERROR
1149
1150          60$:
1151 021206 004737 045224      ;VERIFY THE RESULTS OF THE SEEK COMMAND
      021212 000405      JSR      PC,SEKSTS    ;GO VERIFY RESULTS OF SEEK OPERATION
      021214 000240      BR       70$           ;GO TO 70$ IF NO ERROR
      021216 104000      NOP                   ;RETURN HERE IF ERROR
      021220 004736      EMT                   ;ERROR # DEFINED BY SEKSTS SUBROUTINE
      021222 000137 021534      JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
1152 021226          JMP      180$          ;GO TO 180$ IF ERROR
1153
1154          70$:
1155 021226 013737 021536 001444 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1156 021234 012737 000063 001410      MOV      190$,RMDCO    ;RESTORE DISK ADDRESS
1157 021242 012702 001554      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
1158 021246 112722 000002      MOV      #PUTINX+3,R2  ;EXTEND REGISTER INDEX TABLE
1159 021252 112722 000004      MOVB     #RMWC,(R2)+
1160 021256 112722 000000      MOVB     #RMBA,(R2)+
1161 021262 112722 000200      MOVB     #RMCS1,(R2)+
1162
1163 021266 004737 037360      JSR      PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      021272 000404      BR       80$           ;GO TO 80$ IF NO ERROR
      021274 000240      NOP                   ;RETURN HERE IF ERROR
      021276 104000      EMT                   ;ERROR # DEFINED BY PUT SUBROUTINE
1164 021304 000137 021534      JMP      180$          ;GO TO 180$ IF ERROR

```

```

1165
1166
1167 021304 004737 037722 ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
1168 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
1169 021310 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
021314 000404 BR 90$ ;GO TO 90$ IF NO ERROR
021316 000240 NOP ;RETURN HERE IF ERROR
021320 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
021322 000137 021534 JMP 180$ ;GO TO 180$ IF ERROR
1170 021326 90$:
1171
1172 ;VERIFY RESULTS OF WRITE COMMAND
1173 021326 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
021332 000405 BR 100$ ;GO TO 100$ IF NO ERROR
021334 000240 NOP ;RETURN HERE IF ERROR
021336 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
021340 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021342 000137 021534 JMP 180$ ;GO TO 180$ IF ERROR
1174 021346 100$:
1175 021346 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
021352 000405 BR 110$ ;GO TO 110$ IF NO ERROR
021354 000240 NOP ;RETURN HERE IF ERROR
021356 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
021360 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021362 000137 021534 JMP 180$ ;GO TO 180$ IF ERROR
1176 021366 110$:
1177 021366 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
021372 000405 BR 120$ ;GO TO 120$ IF NO ERROR
021374 000240 NOP ;RETURN HERE IF ERROR
021376 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
021400 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
021402 000137 021534 JMP 180$ ;GO TO 180$ IF ERROR
1178 021406 120$:
1179
1180 ;WRITE CHECK HEADER AND DATA FOR SECTORS JUST WRITTEN
1181 021406 012737 000053 001410 MOV #WCH!GO,RMCS10 ;WRITE CHECK COMMAND
1182
1183 021414 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
021420 000404 BR 130$ ;GO TO 130$ IF NO ERROR
021422 000240 NOP ;RETURN HERE IF ERROR
021424 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
021426 000137 021534 JMP 180$ ;GO TO 180$ IF ERROR
1184 021432 130$:
1185
1186 ;WAIT FOR WRITE CHECK TO COMPLETE AND GET STATUS
1187 021432 004737 037722 JSR PC,TIMOUT ;WAIT FOR WRITE CHECK TO FINISH
1188
1189 021436 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
021442 000404 BR 140$ ;GO TO 140$ IF NO ERROR
021444 000240 NOP ;RETURN HERE IF ERROR
021446 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
021450 000137 021534 JMP 180$ ;GO TO 180$ IF ERROR
1190 021454 140$:
1191
1192 ;VERIFY THE RESULTS OF WRITE CHECK OPERATION
1193 021454 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
021460 000405 BR 150$ ;GO TO 150$ IF NO ERROR
  
```

```

021462 000240      NOP
021464 104000      EMT
021466 004736      JSR      PC,@(SP)+
021470 000137 021534 JMP      180$
1194 021474      150$:
1195 021474 004737 052622 JSR      PC,DTASTS
021500 000405      BR      160$
021502 000240      NOP
021504 104000      EMT
021506 004736      JSR      PC,@(SP)+
021510 000137 021534 JMP      180$
1196 021514      160$:
1197 021514 004737 040740 JSR      PC,SECERR
021520 000405      BR      170$
021522 000240      NOP
021524 104000      EMT
021526 004736      JSR      PC,@(SP)+
021530 000137 021534 JMP      180$
1198 021534      170$:
1199
1200 021534 000401      180$: BR      200$
1201
1202 021536 000000      190$: .WORD 0
1203
1204 021540      200$:
1205
1206

```

```

;*****
;*TEST 17      FORMAT EACH SECTOR ADDRESS
;*****
TST17:

```

```

021540
021540 000004      SCOPE
021542 000240      NOP
021544 012706 001100 MOV      #STACK,SP
021550 013700 001276 MOV      $BASE,R0
021554 013701 001464 MOV      TSTQUE,R1
021560 012737 000017 001226 MOV      #17,$TESTN
1207
1208
1209 021566 012737 000000 001444 ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1210 021574 012737 000000 001416 MOV      #0,RMDCO
1211 021602 012737 010000 001442 MOV      #0,RMDAO
1212 021610      MOV      #FMT16,RMOFO
1213 021610 012737 177376 001412 5$: MOV      #-258.,RMWCO
1214 021616 012737 101206 001414 MOV      #BUFONE,RMBAO
1215 021624 012737 000062 001410 MOV      #WH,RMCS10
1216
1217
021632 004737 034276 ;VERIFY THAT SECTOR IS NOT BAD
021636 000405      JSR      PC,BADSCT
021640 104401 063120 BR      10$
021644 104000      TYPE      ,SCTMSG
021646 000137 022302 EMT
1218 021652      JMP      190$
1219 021652 012737 001416 001174 10$: MOV      #RMDAO,$TMPO
1220 021660 012737 000001 001176 MOV      #1,$TMP1
1221 021666 004737 036224 JSR      PC,GENBUF
1222 021672      20$:

```

```

;RETURN HERE IF ERROR
;ERROR # DEFINED BY PRIERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 180$ IF ERROR

;GO VERIFY RESULTS OF DATA TRANSFER
;GO TO 160$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY DTASTS SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 180$ IF ERROR

;GO CHECK FOR SECONDARY ERRORS
;GO TO 170$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY SECERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 180$ IF ERROR

;TEMPORARY STORAGE

;SCOPE CALL
;START OF TEST
;INITIALIZE STACK POINTER
;R0 = UNIBUS ADDRESS
;(R1) = DEVICE BEING TESTED
;;SET TEST NUMBER IN APT MAIL BOX

;CYLINDER = 0
;TRACK = 0, SECTOR = 0
;16 BIT FORMAT

;2 + 256 WORDS (2'S COMP)
;DATA BUFFER ADDRESS
;WRITE HEADER AND DATA

;CALL BAD SECTOR MODULE
;GO TO 10$ IF NO ERROR
;TYPE BAD SECTOR MESSAGE
;ERROR # DEFINED BY BADSCT SUBROUTINE
;GO TO 190$ IF ERROR

;USE SECTOR FOR DATA PATTERN

;GO GENERATE DATA BUFFER

```



```

022054 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
022056 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022060 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1253 022064 110$:
1254 022064 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
022070 000405 BR 120$ ;GO TO 120$ IF NO ERROR
022072 000240 NOP ;RETURN HERE IF ERROR
022074 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
022076 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022100 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1255 022104 120$:
1256
1257 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
1258 022104 012737 000073 001410 MOV #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
1259 022112 012737 102212 001414 MOV #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
1260
1261 022120 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
022124 000404 BR 130$ ;GO TO 130$ IF NO ERROR
022126 000240 NOP ;RETURN HERE IF ERROR
022130 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
022132 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1262 022136 130$:
1263
1264 ;WAIT FOR READ TO COMPLETE AND GET STATUS
1265 022136 004737 037722 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
1266
1267 022142 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
022146 000404 BR 140$ ;GO TO 140$ IF NO ERROR
022150 000240 NOP ;RETURN HERE IF ERROR
022152 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
022154 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1268 022160 140$:
1269
1270 ;VERIFY THE RESULTS OF READ OPERATION
1271 022160 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
022164 000405 BR 150$ ;GO TO 150$ IF NO ERROR
022166 000240 NOP ;RETURN HERE IF ERROR
022170 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
022172 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022174 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1272 022200 150$:
1273 022200 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
022204 000405 BR 160$ ;GO TO 160$ IF NO ERROR
022206 000240 NOP ;RETURN HERE IF ERROR
022210 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
022212 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022214 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1274 022220 160$:
1275 022220 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
022224 000405 BR 170$ ;GO TO 170$ IF NO ERROR
022226 000240 NOP ;RETURN HERE IF ERROR
022230 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
022232 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
022234 000137 022302 JMP 190$ ;GO TO 190$ IF ERROR
1276 022240 170$:
1277
1278 ;VERIFY DATA
  
```

```

1279 022240 004737 036462      JSR    PC,CMPBUF      ;GO COMPARE WRITE, READ DATA BUFFERS
      022244 101206          .WORD  BUFONE        ;STARTING ADDRESS OF WRITE BUFFER
      022246 102212          .WORD  BUFTWO        ;STARTING ADDRESS OF READ BUFFER
      022250 000404          BR     180$          ;GO TO 180$ IF NO ERROR
      022252 000240          NOP                    ;RETURN HERE IF ERROR
      022254 104000          EMT                    ;ERROR # DEFINED BY CMPBUF SUBROUTINE
      022256 000137 022302    JMP     190$          ;GO TO 190$ IF ERROR
  
```

```

1280
1281      ;INCREMENT ADDRESS AND FORMAT NEXT SECTOR
  
```

```

1282 022262      180$:
1283 022262 005237 001416      INC     RMDAO        ;ADVANCE SECTOR COUNT
1284 022266 122737 000037 001416  CMPB   #31.,RMDAO   ;DONE ALL SECTORS??
1285 022274 103402          BLO    190$          ;YES!!
1286 022276 000137 021610    JMP     5$           ;GO DO NEXT SECTOR
  
```

```

1287 022302      190$:
1288
1289
  
```

```

:*****
:*TEST 20      FORMAT EACH TRACK ADDRESS
:*****
  
```

```

022302      TST20:
022302 000004          SCOPE                ;SCOPE CALL
022304 000240          NOP                    ;START OF TEST
022306 012706 001100    MOV     #STACK,SP    ;INITIALIZE STACK POINTER
022312 013700 001276    MOV     $BASE,R0     ;R0 = UNIBUS ADDRESS
022316 013701 001464    MOV     TSTQUE,R1    ;(R1) = DEVICE BEING TESTED
022322 012737 000020 001226  MOV     #20,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
  
```

```

1290
1291      ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1292 022330 012737 000000 001444  MOV     #0,RMDCO     ;CYLINDER = 0
1293 022336 012737 000000 001416  MOV     #0,RMDAO     ;TRACK = 0, SECTOR = 0
1294 022344 012737 010000 001442  MOV     #FMT16,RMOFO ;16 BIT FORMAT
  
```

```

1295 022352      5$:
1296 022352 012737 177376 001412  MOV     #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
1297 022360 012737 101206 001414  MOV     #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1298 022366 012737 000062 001410  MOV     #WH,RMCS10   ;WRITE HEADER AND DATA
  
```

```

1299
1300      ;VERIFY THAT SECTOR IS NOT BAD
      022374 004737 034276      JSR    PC,BADSCT    ;CALL BAD SECTOR MODULE
      022400 000405          BR     10$          ;GO TO 10$ IF NO ERROR
      022402 104401 063120      TYPE   ,SCTMSG     ;TYPE BAD SECTOR MESSAGE
      022406 104000          EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
      022410 000137 023044      JMP     190$        ;GO TO 190$ IF ERROR
  
```

```

1301 022414      10$:
1302 022414 012737 001416 001174  MOV     #RMDAO,$TMPO ;USE TRACK FOR DATA PATTERN
1303 022422 012737 000001 001176  MOV     #1,$TMP1
1304 022430 004737 036224      JSR    PC,GENBUF    ;GO GENERATE DATA BUFFER
  
```

```

1305 022434      20$:
1306
1307      ;PREPARE DEVICE FOR DATA TRANSFER
  
```

```

1308 022434 004737 033352      JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
      022440 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
  
```

```

022442 000404 BR 30$ ;VERIFY RECALIBRATION
022444 000240 NOP ;GO TO 30$ IF NO ERROR
022446 104000 EMT ;RETURN HERE IF ERROR
022450 000137 023044 JMP 190$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
1309 022454 30$: ;GO TO 190$ IF ERROR
1310
1311 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1312 022454 012737 000063 001410 MOV #WH!GO, RMCS10 ;WRITE HEADER AND DATA
1313 022462 012702 001551 MOV #PUTINX, R2 ;WRITE REGISTER INDEX TABLE
1314 022466 112722 000006 MOVB #RMDA, (R2)+
1315 022472 112722 000034 MOVB #RMDC, (R2)+
1316 022476 112722 000032 MOVB #RMOF, (R2)+
1317 022502 112722 000002 MOVB #RMWC, (R2)+
1318 022506 112722 000004 MOVB #RMBA, (R2)+
1319 022512 112722 000000 MOVB #RMCS1, (R2)+
1320 022516 112722 000200 MOVB #200, (R2)+
1321
1322 022522 004737 037360 JSR PC, PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
022526 000404 BR 80$ ;GO TO 80$ IF NO ERROR
022530 000240 NOP ;RETURN HERE IF ERROR
022532 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
022534 000137 023044 JMP 190$ ;GO TO 190$ IF ERROR
1323 022540 80$:
1324
1325 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1326 022540 004737 037024 JSR PC, GETSTS
1327 022544 004737 037722 JSR PC, TIMEOUT ;WAIT FOR COMMAND TO COMPLETE
1328
1329 022550 004737 037110 JSR PC, GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
022554 000404 BR 90$ ;GO TO 90$ IF NO ERROR
022556 000240 NOP ;RETURN HERE IF ERROR
022560 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
022562 000137 023044 JMP 190$ ;GO TO 190$ IF ERROR
1330 022566 90$:
1331
1332 ;VERIFY RESULTS OF WRITE COMMAND
1333 022566 004737 040106 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
022572 000405 BR 100$ ;GO TO 100$ IF NO ERROR
022574 000240 NOP ;RETURN HERE IF ERROR
022576 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
022600 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
022602 000137 023044 JMP 190$ ;GO TO 190$ IF ERROR
1334 022606 100$:
1335 022606 004737 052622 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
022612 000405 BR 110$ ;GO TO 110$ IF NO ERROR
022614 000240 NOP ;RETURN HERE IF ERROR
022616 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
022620 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
022622 000137 023044 JMP 190$ ;GO TO 190$ IF ERROR
1336 022626 110$:
1337 022626 004737 040740 JSR PC, SECERR ;GO CHECK FOR SECONDARY ERRORS
022632 000405 BR 120$ ;GO TO 120$ IF NO ERROR
022634 000240 NOP ;RETURN HERE IF ERROR
022636 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
022640 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
022642 000137 023044 JMP 190$ ;GO TO 190$ IF ERROR

```

```

1338 022646          120$:
1339
1340          ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
1341 022646 012737 000073 001410      MOV      #RH!GO,RMCS10  ;READ HEADER & DATA COMMAND
1342 022654 012737 102212 001414      MOV      #BUFTWO,RMBAO  ;CHANGE BUS ADDRESS
1343
1344 022662 004737 037360          JSR      PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
          022666 000404          BR       130$          ;GO TO 130$ IF NO ERROR
          022670 000240          NOP
          022672 104000          EMT
          022674 000137 023044      JMP      190$          ;ERROR # DEFINED BY PUT SUBROUTINE
          ;GO TO 190$ IF ERROR
1345 022700          130$:
1346
1347          ;WAIT FOR READ TO COMPLETE AND GET STATUS
1348 022700 004737 037722          JSR      PC,TIMOUT      ;WAIT FOR READ TO COMPLETE
1349
1350 022704 004737 037110          JSR      PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
          022710 000404          BR       140$          ;GO TO 140$ IF NO ERROR
          022712 000240          NOP
          022714 104000          EMT
          022716 000137 023044      JMP      190$          ;ERROR # DEFINED BY GET SUBROUTINE
          ;GO TO 190$ IF ERROR
1351 022722          140$:
1352
1353          ;VERIFY THE RESULTS OF READ OPERATION
1354 022722 004737 040106          JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
          022726 000405          BR       150$          ;GO TO 150$ IF NO ERROR
          022730 000240          NOP
          022732 104000          EMT
          022734 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
          022736 000137 023044      JMP      190$          ;GO TO 190$ IF ERROR
1355 022742          150$:
1356 022742 004737 052622          JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
          022746 000405          BR       160$          ;GO TO 160$ IF NO ERROR
          022750 000240          NOP
          022752 104000          EMT
          022754 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
          022756 000137 023044      JMP      190$          ;GO TO 190$ IF ERROR
1357 022762          160$:
1358 022762 004737 040740          JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
          022766 000405          BR       170$          ;GO TO 170$ IF NO ERROR
          022770 000240          NOP
          022772 104000          EMT
          022774 004736          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
          022776 000137 023044      JMP      190$          ;GO TO 190$ IF ERROR
1359 023002          170$:
1360
1361          ;VERIFY DATA
1362 023002 004737 036462          JSR      PC,CMPBUF      ;GO COMPARE WRITE, READ DATA BUFFERS
          023006 101206          .WORD   BUFSIZE        ;STARTING ADDRESS OF WRITE BUFFER
          023010 102212          .WORD   BUFTWO         ;STARTING ADDRESS OF READ BUFFER
          023012 000404          BR       180$          ;GO TO 180$ IF NO ERROR
          023014 000240          NOP
          023016 104000          EMT
          023020 000137 023044      JMP      190$          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
          ;GO TO 190$ IF ERROR
1363
1364          ;INCREMENT ADDRESS AND FORMAT NEXT TRACK
1365 023024          180$:

```

```

1366 023024 105237 001417          INCB   RMDAO+1      ;ADVANCE TRACK COUNT
1367 023030 123737 001417 001333    CMPB   RMDAO+1,LSTRK+1 ;LAST TRACK ?
1368 023036 101002          BHI    190$        ;YES!!
1369 023040 000137 022352          JMP    5$          ;GO DO NEXT SECTOR
1370 023044          190$:
1371
1372
;*****
;*TEST 21          FORMAT PRIME CYLINDERS
;*****
TST21:
023044          000004          SCOPE          ;SCOPE CALL
023044          000240          NOP           ;START OF TEST
023050          012706 001100          MOV    #STACK,SP ;INITIALIZE STACK POINTER
023054          013700 001276          MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
023060          013701 001464          MOV    TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
023064          012737 000021 001226          MOV    #21,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

1373
1374          ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1375 023072 012737 000001 001444          MOV    #1,RMDCO  ;CYLINDER = 1
1376 023100 012737 000000 001416          MOV    #0,RMDAO  ;TRACK = 0, SECTOR = 0
1377 023106          5$:
1378 023106 012737 010000 001442          MOV    #FMT16,RMOFO ;16 BIT FORMAT
1379 023114 012737 177376 001412          MOV    #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
1380 023122 012737 101206 001414          MOV    #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1381 023130 012737 000062 001410          MOV    #WH,RMCS10 ;WRITE HEADER AND DATA
1382
1383          ;VERIFY THAT SECTOR IS NOT BAD
023136 004737 034276          JSR    PC,BADSCT  ;CALL BAD SECTOR MODULE
023142 000405          BR     10$        ;GO TO 10$ IF NO ERROR
023144 104401 063120          TYPE   ,SCTMSG   ;TYPE BAD SECTOR MESSAGE
023150 104000          EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
023152 000137 023606          JMP    190$      ;GO TO 190$ IF ERROR
1384 023156          10$:
1385 023156 012737 001444 001174          MOV    #RMDCO,$TMP0 ;USE CYLINDER FOR DATA PATTERN
1386 023164 012737 000001 001176          MOV    #1,$TMP1
1387 023172 004737 036224          JSR    PC,GENBUF  ;GO GENERATE DATA BUFFER
1388
1389 023176          20$:
1390
1391          ;PREPARE DEVICE FOR DATA TRANSFER
1392 023176 004737 033352          JSR    PC,TSTPRP ;PREPARE DEVICE FOR TEST
023202 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 30$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 190$ IF ERROR

023204 000404          BR     30$
023206 000240          NOP
023210 104000          EMT
023212 000137 023606          JMP    190$
1393 023216          30$:
1394
1395          ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1396 023216 012737 000063 001410          MOV    #WH!GO,RMCS10 ;WRITE HEADER AND DATA

```

```

1397 023224 012702 001551      MOV    #PUTINX,R2      ;WRITE REGISTER INDEX TABLE
1398 023230 112722 000006      MOVB  #RMDA,(R2)+
1399 023234 112722 000034      MOVB  #RMDC,(R2)+
1400 023240 112722 000032      MOVB  #RMOF,(R2)+
1401 023244 112722 000002      MOVB  #RMWC,(R2)+
1402 023250 112722 000004      MOVB  #RMBA,(R2)+
1403 023254 112722 000000      MOVB  #RMCS1,(R2)+
1404 023260 112722 000200      MOVB  #200,(R2)+
1405 023264 004737 037360      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR    80$                   ;GO TO 80$ IF NO ERROR
                                NOP                   ;RETURN HERE IF ERROR
                                EMT                   ;ERROR # DEFINED BY PUT SUBROUTINE
                                JMP   190$                  ;GO TO 190$ IF ERROR

1406 023302      80$:
1407
1408      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1409 023302 004737 037024      JSR   PC,GETSTS
1410
1411      ;WAIT FOR WRITE COMMAND TO COMPLETE AND READ STATUS
1412 023306 004737 037722      JSR   PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1413 023312 004737 037110      JSR   PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
                                BR    90$                   ;GO TO 90$ IF NO ERROR
                                NOP                   ;RETURN HERE IF ERROR
                                EMT                   ;ERROR # DEFINED BY GET SUBROUTINE
                                JMP   190$                  ;GO TO 190$ IF ERROR

1414 023324 000137 023606      90$:
1415
1416      ;VERIFY RESULTS OF WRITE COMMAND
1417 023330 004737 040106      JSR   PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
                                BR    100$                  ;GO TO 100$ IF NO ERROR
                                NOP                   ;RETURN HERE IF ERROR
                                EMT                   ;ERROR # DEFINED BY PRIERR SUBROUTINE
                                JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
                                JMP   190$                  ;GO TO 190$ IF ERROR

1418 023342 004736 023606      100$:
1419 023344 000137 052622      JSR   PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
                                BR    110$                  ;GO TO 110$ IF NO ERROR
                                NOP                   ;RETURN HERE IF ERROR
                                EMT                   ;ERROR # DEFINED BY DTASTS SUBROUTINE
                                JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
                                JMP   190$                  ;GO TO 190$ IF ERROR

1420 023350 004737 023606      110$:
1421 023350 004737 040740      JSR   PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
                                BR    120$                  ;GO TO 120$ IF NO ERROR
                                NOP                   ;RETURN HERE IF ERROR
                                EMT                   ;ERROR # DEFINED BY SECERR SUBROUTINE
                                JSR   PC,@(SP)+          ;GO BACK FOR MORE ERROR CHECKS
                                JMP   190$                  ;GO TO 190$ IF ERROR

1422 023364 000137 023606      120$:
1423
1424      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
1425 023410 012737 000073 001410      MOV   #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
1426 023416 012737 102212 001414      MOV   #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
1427 023424 004737 037360      JSR   PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
                                BR    130$                  ;GO TO 130$ IF NO ERROR
                                NOP                   ;RETURN HERE IF ERROR
                                EMT                   ;ERROR # DEFINED BY PUT SUBROUTINE

                                023430 000404
                                023432 000240
                                023434 104000

```

1428 023436 000137 023606
 1429 023442
 1430
 1431 023442 004737 037722
 1432 023446 004737 037110
 023452 000404
 023454 000240
 023456 104000
 023460 000137 023606
 1433 023464
 1434
 1435
 1436 023464 004737 040106
 023470 000405
 023472 000240
 023474 104000
 023476 004736
 023500 000137 023606
 1437 023504
 1438 023504 004737 052622
 023510 000405
 023512 000240
 023514 104000
 023516 004736
 023520 000137 023606
 1439 023524
 1440 023524 004737 040740
 023530 000405
 023532 000240
 023534 104000
 023536 004736
 023540 000137 023606
 1441 023544
 1442
 1443
 1444 023544 004737 036462
 023550 101206
 023552 102212
 023554 000404
 023556 000240
 023560 104000
 023562 000137 023606
 1445 023566
 1446
 1447
 1448 023566 006337 001444
 1449 023572 023727 001444 001000
 1450 023600 003002
 1451 023602 000137 023106
 1452 023606
 1453
 1454

```

130$: JMP 190$ ;GO TO 190$ IF ERROR

;WAIT FOR READ TO COMPLETE AND GET STATUS
JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
EMT ;ERROR # DEFINED BY GET SUBROUTINE
JMP 190$ ;GO TO 190$ IF ERROR

140$:

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

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

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

170$:

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

180$:

;INCREMENT ADDRESS AND FORMAT NEXT PRIME CYLINDER
ASL RMDCO ;ADVANCE CYLINDER COUNT
CMP RMDCO,#512. ;DONE ALL PRIME CYLINDERS ?
BGT 190$ ;YES!!
JMP 5$ ;GO DO NEXT CYLINDER

190$:

;*****
;*TEST 22 READ HEADER & DATA IN LAST SECTOR
;*****
TST22: SCOPE ;SCOPE CALL

```



```

1488 024026 000137 024112          JMP      190$          ;GO TO 190$ IF ERROR
1489 024032          140$:
1490          ;VERIFY THE RESULTS OF READ OPERATION
1491 024032 004737 040106          JSR      PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
          024036 000405          BR       150$          ;GO TO 150$ IF NO ERROR
          024040 000240          NOP                      ;RETURN HERE IF ERROR
          024042 104000          EMT                      ;ERROR # DEFINED BY PRIERR SUBROUTINE
          024044 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
          024046 000137 024112          JMP      190$          ;GO TO 190$ IF ERROR
1492 024052          150$:
1493 024052 004737 052622          JSR      PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
          024056 000405          BR       160$          ;GO TO 160$ IF NO ERROR
          024060 000240          NOP                      ;RETURN HERE IF ERROR
          024062 104000          EMT                      ;ERROR # DEFINED BY DTASTS SUBROUTINE
          024064 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
          024066 000137 024112          JMP      190$          ;GO TO 190$ IF ERROR
1494 024072          160$:
1495 024072 004737 040740          JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
          024076 000405          BR       170$          ;GO TO 170$ IF NO ERROR
          024100 000240          NOP                      ;RETURN HERE IF ERROR
          024102 104000          EMT                      ;ERROR # DEFINED BY SECERR SUBROUTINE
          024104 004736          JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
          024106 000137 024112          JMP      190$          ;GO TO 190$ IF ERROR
1496 024112          170$:
1497          190$:
1498 024112
1499
1500

```

```

:*****
:*TEST 23      READ HEADER & DATA W/ AOE ERROR
:*****

```

```

024112          TST23:
024112 000004          SCOPE          ;SCOPE CALL
024114 000240          NOP          ;START OF TEST
024116 012706 001100          MOV      #STACK,SP  ;INITIALIZE STACK POINTER
024122 013700 001276          MOV      $BASE,R0   ;R0 = UNIBUS ADDRESS
024126 013701 001464          MOV      TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
024132 012737 000023 001226          MOV      #23,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
1501
1502          ;SETUP PARAMETERS FOR READING 2 SECTORS STARTING WITH LAST SECTOR
1503 024140 013737 001332 001416          MOV      LSTRK,RMDAO ;SET LAST TRACK AND
1504 024146 112737 000036 001416          MOV      #30.,RMDAO ;LAST SECTOR
1505 024154 012737 010000 001442          MOV      #FMT16,RMOFO ;16 BIT FORMAT
1506 024162          20$:
1507 024162 012737 001466 001444          MOV      #822.,RMDCO ;LAST CYLINDER
1508 024170 012737 176774 001412          MOV      #-258.*2,RMWCO ;READ 2 SECTORS
1509 024176 012737 102212 001414          MOV      #BUFTWO,RMBAO ;DATA BUFFER ADDRESS
1510 024204 012737 000073 001410          MOV      #RH!GO,RMCS10 ;READ HEADER AND DATA
1511
1512          ;PREPARE DEVICE FOR DATA TRANSFER
1513 024212 004737 033352          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
          024216 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 30$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 190$ IF ERROR
024220 000404 BR 30$
024222 000240 NOP
024224 104000 EMT
024226 000137 024416 JMP 190$
1514 024232 30$:
1515 024232 012702 001551 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
1516 024236 112722 000006 MOVB #RMDA,(R2)+
1517 024242 112722 000034 MOVB #RMDC,(R2)+
1518 024246 112722 000032 MOVB #RMOF,(R2)+
1519 024252 112722 000002 MOVB #RMWC,(R2)+
1520 024256 112722 000004 MOVB #RMBA,(R2)+
1521 024262 112722 000000 MOVB #RMCS1,(R2)+
1522 024266 112722 000200 MOVB #200,(R2)+
1523
1524 024272 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
024276 000404 BR 130$ ;GO TO 130$ IF NO ERROR
024300 000240 NOP ;RETURN HERE IF ERROR
024302 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
024304 000137 024416 JMP 190$ ;GO TO 190$ IF ERROR
1525 024310 130$:
1526
1527 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1528 024310 004737 037024 JSR PC,GETSTS
1529 024314 004737 037722 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
1530
1531 024320 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
024324 000404 BR 140$ ;GO TO 140$ IF NO ERROR
024326 000240 NOP ;RETURN HERE IF ERROR
024330 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
024332 000137 024416 JMP 190$ ;GO TO 190$ IF ERROR
1532 024336 140$:
1533
1534 ;VERIFY THE RESULTS OF READ OPERATION
1535 024336 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
024342 000405 BR 150$ ;GO TO 150$ IF NO ERROR
024344 000240 NOP ;RETURN HERE IF ERROR
024346 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
024350 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
024352 000137 024416 JMP 190$ ;GO TO 190$ IF ERROR
1536 024356 150$:
1537 024356 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
024362 000405 BR 160$ ;GO TO 160$ IF NO ERROR
024364 000240 NOP ;RETURN HERE IF ERROR
024366 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
024370 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
024372 000137 024416 JMP 190$ ;GO TO 190$ IF ERROR
1538 024376 160$:
1539 024376 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
024402 000405 BR 170$ ;GO TO 170$ IF NO ERROR
024404 000240 NOP ;RETURN HERE IF ERROR
024406 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
024410 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
024412 000137 024416 JMP 190$ ;GO TO 190$ IF ERROR
1540 024416 170$:
1541

```

1542 024416
 1543
 1544

190\$:

```

:*****
:*TEST 24 READ INVALID SECTOR ADDRESS
:*****
TST:

```

024416
 024416 000004
 024420 000240
 024422 012706 001100
 024426 013700 001276
 024432 013701 001464
 024436 012737 000024 001226
 1545
 1546 024444 012737 000000 001444
 1547 024452 012737 000036 001416
 1548 024460 012737 000000 001442
 1549 024466 012737 101206 001414
 1550 024474 012737 177376 001412
 1551 024502 012737 000073 001410
 1552 024510
 1553
 1554

```

SCOPE :SCOPE CALL
NOP :START OF TEST
MOV #STACK,SP :INITIALIZE STACK POINTER
MOV $BASE,R0 :R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 : (R1) = DEVICE BEING TESTED
MOV #24,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #0,RMDCO :CYLINDER = 0
MOV #30,RMDAO :TRACK = 0, INVALID SECTOR = 30.
MOV #0,RMOFO :18 BIT FORMAT
MOV #BUFONE,RMBAO :CHANGE BUS ADDRESS
MOV #-258,RMWCO :2 + 256 WORDS (2'S COMP)
MOV #RH!GO,RMCS10 :READ HEADER AND DATA

```

10\$:

1555 024510 004737 033352
 024514 154130

```

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP :PREPARE DEVICE FOR TEST
.WORD 154130 :TASK DESCRIPTOR AS FOLLOWS:
:SELECT DEVICE & VERIFY DEVICE AVAILABLE
:CLEAR CONTROLLER & SELECT DEVICE
:VERIFY CONTROLLER CLEAR OPERATION
:PACK ACKNOWLEDGE IF VOLUME NOT VALID
:VERIFY PACK ACKNOWLEDGE
:RECALIBRATE IF "SKI" OR "PIP" IS SET
:VERIFY RECALIBRATION
:GO TO 20$ IF NO ERROR
:RETURN HERE IF ERROR
:ERROR # DEFINED BY TSTPRP SUBROUTINE
:GO TO 90$ IF ERROR
BR 20$
NOP
EMT
JMP 90$

```

024516 000404
 024520 000240
 024522 104000
 024524 000137 024760
 1556 024530
 1557
 1558

20\$:

1559 024530 012702 001551
 1560 024534 112722 000006
 1561 024540 112722 000034
 1562 024544 112722 000032
 1563 024550 112722 000002
 1564 024554 112722 000004
 1565 024560 112722 000000
 1566 024564 112722 000200
 1567
 1568 024570 004737 037360
 024574 000404
 024576 000240
 024600 104000
 024602 000137 024760
 1569 024606
 1570
 1571

```

;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
MOV #PUTINX,R2 :LOAD REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+ :SET TERMINATOR BYTE
JSR PC,PUT :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
BR 30$ :GO TO 30$ IF NO ERROR
NOP :RETURN HERE IF ERROR
EMT :ERROR # DEFINED BY PUT SUBROUTINE
JMP 90$ :GO TO 90$ IF ERROR

```

30\$:

1572 024606 004737 037024
 1573 024612 004737 037722

```

;SETUP INPUT REGISTER BUFFER FOR READING STATUS
JSR PC,GETSTS
JSR PC,TIMOUT :WAIT FOR COMMAND TO COMPLETE

```

```

1574
1575 024616 004737 037110      JSR    PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      024622 000404          BR     40$            ;GO TO 40$ IF NO ERROR
      024624 000240          NOP                    ;RETURN HERE IF ERROR
      024626 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      024630 000137 024760      JMP    90$            ;GO TO 90$ IF ERROR
1576 024634      40$:
1577
1578 ;VERIFY RESULTS OF READ COMMAND
1579 024634 004737 040106      JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      024640 000405          BR     50$            ;GO TO 50$ IF NO ERROR
      024642 000240          NOP                    ;RETURN HERE IF ERROR
      024644 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      024646 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      024650 000137 024760      JMP    90$            ;GO TO 90$ IF ERROR
1580 024654      50$:
1581 024654 004737 052622      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      024660 000405          BR     60$            ;GO TO 60$ IF NO ERROR
      024662 000240          NOP                    ;RETURN HERE IF ERROR
      024664 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      024666 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      024670 000137 024760      JMP    90$            ;GO TO 90$ IF ERROR
1582 024674      60$:
1583 024674 004737 040740      JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      024700 000405          BR     70$            ;GO TO 70$ IF NO ERROR
      024702 000240          NOP                    ;RETURN HERE IF ERROR
      024704 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      024706 004736          JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      024710 000137 024760      JMP    90$            ;GO TO 90$ IF ERROR
1584 024714      70$:
1585
1586 ;INCREMENT ADDRESS AND READ NEXT SECTOR
1587 024714 005237 001416      INC    RMDAO          ;ADVANCE SECTOR COUNT
1588 024720 022737 000037 001416  CMP    #31.,RMDAO      ;DONE ALL SECTORS??
1589 024726 103012          BHS    80$            ;NO !!
1590
1591 024730 032737 010000 001442  BIT    #FMT16,RMOFO    ;TEST 16 BIT MODE YET ?
1592 024736 001010          BNE    90$            ;YES !!
1593 024740 012737 000037 001416  MOV    #31.,RMDAO      ;TRACK = 0, INVALID SECTOR = 31.
1594 024746 012737 010000 001442  MOV    #FMT16,RMOFO    ;SET 16 BIT MODE IN OFFSET AND
1595 024754 000137 024510      JMP    10$           ;TEST AGAIN.
1596
1597 024760      80$:
1598
1599
;*****
;*TEST 25      READ INVALID TRACK ADDRESS
;*****
TST25:
      024760      SCOPE          ;SCOPE CALL
      024760 000004          NOP                    ;START OF TEST
      024762 000240          MOV    #STACK,SP      ;INITIALIZE STACK POINTER
      024764 012706 001100          MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
      024770 013700 001276          MOV    TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
      024774 013701 001464          MOV    #25,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
      025000 012737 000025 001226
1600
1601 025006 012737 000000 001444  MOV    #0,RMDCO        ;CYLINDER = 0
1602 025014 013737 001332 001416  MOV    LSTRK,RMDAO     ;LOAD LAST TRACK, SECTOR = 0
  
```

```

1603 025022 105237 001417          INCB      RMDAO+1          ;INCREMENT TO FIRST INVALID TRACK
1604 025026 012737 010000 001442    MOV       #FMT16,RMOFO    ;16 BIT FORMAT
1605 025034 012737 177376 001412    MOV       #-258.,RMWCO    ;2 + 256 WORDS (2'S COMP)
1606 025042 012737 101206 001414    MOV       #BUFONE,RMBAO   ;DATA BUFFER ADDRESS
1607 025050 012737 000073 001410    MOV       #RH!GO,RMCS10   ;READ HEADER AND DATA
1608 025056
1609
1610
1611 025056 004737 033352          ;PREPARE DEVICE FOR DATA TRANSFER
      025062 154130          JSR       PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      .WORD 154130          .WORD     ;TASK DESCRIPTOR AS FOLLOWS:
                                ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                ;CLEAR CONTROLLER & SELECT DEVICE
                                ;VERIFY CONTROLLER CLEAR OPERATION
                                ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                ;VERIFY PACK ACKNOWLEDGE
                                ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                ;VERIFY RECALIBRATION
                                ;GO TO 20% IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 80% IF ERROR
10$:
      025064 000404          BR        20%
      025066 000240          NOP
      025070 104000          EMT
      025072 000137 025302    JMP       80%
1612 025076
1613
1614          ;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
1615 025076 012702 001551          MOV       #PUTINX,R2      ;LOAD REGISTER INDEX TABLE
1616 025102 112722 000006          MOVB     #RMDA,(R2)+
1617 025106 112722 000034          MOVB     #RMDC,(R2)+
1618 025112 112722 000032          MOVB     #RMOF,(R2)+
1619 025116 112722 000002          MOVB     #RMWC,(R2)+
1620 025122 112722 000004          MOVB     #RMBA,(R2)+
1621 025126 112722 000000          MOVB     #RMCS1,(R2)+
1622 025132 112722 000200          MOVB     #200,(R2)+      ;SET TERMINATOR BYTE
1623
1624 025136 004737 037360          JSR       PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      025142 000404          BR        30%
      025144 000240          NOP
      025146 104000          EMT
      025150 000137 025302    JMP       80%
1625 025154
1626
1627          ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1628 025154 004737 037024          JSR       PC,GETSTS
1629 025160 004737 037722          JSR       PC,TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1630
1631 025164 004737 037110          JSR       PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      025170 000404          BR        40%
      025172 000240          NOP
      025174 104000          EMT
      025176 000137 025302    JMP       80%
1632 025202
1633
1634          ;VERIFY RESULTS OF READ COMMAND
1635 025202 004737 040106          JSR       PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      025206 000405          BR        50%
      025210 000240          NOP
      025212 104000          EMT
      025214 004736          JSR       PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
  
```

```

1636 025216 000137 025302          50$: JMP      80$          ;GO TO 80$ IF ERROR
1637 025222 004737 052622          JSR     PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
      025226 000405          BR      60$          ;GO TO 60$ IF NO ERROR
      025230 000240          NOP                    ;RETURN HERE IF ERROR
      025232 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      025234 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      025236 000137 025302          JMP     80$          ;GO TO 80$ IF ERROR
1638 025242 004737 040740          60$: JSR     PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
1639 025246 000405          BR      70$          ;GO TO 70$ IF NO ERROR
      025250 000240          NOP                    ;RETURN HERE IF ERROR
      025252 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      025254 004736          JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      025256 000137 025302          JMP     80$          ;GO TO 80$ IF ERROR
1640 025262 105237 001417          70$: INCB    RMDAO+1    ;ADVANCE TRACK COUNT
1641 025266 123727 001417 000200    CMPB   RMDAO+1,#128. ;DONE ALL TRACKS??
1642 025274 101002          BHI    80$          ;YES!!
1643 025276 000137 025056          JMP     10$         ;GO DO NEXT TRACK
1644 025302
1645
1646
1647
1648
1649
;*****
;*TEST 26      READ INVALID CYLINDER ADDRESS
;*****
TST26:
      025302 000004          SCOPE                    ;SCOPE CALL
      025302 000240          NOP                    ;START OF TEST
      025304 012706 001100    MOV     #STACK,SP      ;INITIALIZE STACK POINTER
      025312 013700 001276    MOV     $BASE,R0       ;R0 = UNIBUS ADDRESS
      025316 013701 001464    MOV     TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
      025322 012737 000026 001226  MOV     #26,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
1650 025330 012737 001467 001444    MOV     #823.,RMDCO    ;START AT FIRST INVALID CYLINDER
1651 025336 012737 000000 001416    MOV     #0,RMDAO       ;TRACK = 0, SECTOR = 0
1652 025344 012737 010000 001442    MOV     #FMT16,RMOFO   ;16 BIT FORMAT
1653 025352 012737 177376 001412    MOV     #-258.,RMWCO   ;2 + 256 WORDS (2'S COMP)
1654 025360 012737 101206 001414    MOV     #BUFONE,RMBAO  ;DATA BUFFER ADDRESS
1655 025366 012737 000073 001410    MOV     #RH!GO,RMCS10 ;READ HEADER AND DATA
1656 025374
1657
1658
1659
1660 025374 004737 033352          10$: JSR     PC,TSTPRP    ;PREPARE DEVICE FOR DATA TRANSFER
      025400 154130          .WORD  154130        ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE
;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
      025402 000404          BR      20$          ;GO TO 20$ IF NO ERROR
      025404 000240          NOP                    ;RETURN HERE IF ERROR
      025406 104000          EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      025410 000137 025620          JMP     80$          ;GO TO 80$ IF ERROR
  
```

```

1661 025414          20$:
1662
1663                ;SETUP AND EXECUTE READ HEADER AND DATA COMMAND
1664 025414 012702 001551      MOV    #PUTINX,R2      ;LOAD REGISTER INDEX TABLE
1665 025420 112722 000006      MOVB  #RMDA,(R2)+
1666 025424 112722 000034      MOVB  #RMDC,(R2)+
1667 025430 112722 000032      MOVB  #RMOF,(R2)+
1668 025434 112722 000002      MOVB  #RMWC,(R2)+
1669 025440 112722 000004      MOVB  #RMBA,(R2)+
1670 025444 112722 000000      MOVB  #RMCS1,(R2)+
1671 025450 112722 000200      MOVB  #200,(R2)+      ;SET TERMINATOR BYTE
1672
1673 025454 004737 037360      JSR   PC,PUT          ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      025460 000404      BR    30$            ;GO TO 30$ IF NO ERROR
      025462 000240      NOP                    ;RETURN HERE IF ERROR
      025464 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      025466 000137 025620      JMP   80$            ;GO TO 80$ IF ERROR
1674 025472
1675
1676                ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1677 025472 004737 037024      JSR   PC,GETSTS
1678 025476 004737 037722      JSR   PC,!TIMOUT      ;WAIT FOR COMMAND TO COMPLETE
1679
1680 025502 004737 037110      JSR   PC,GET          ;GO READ REGISTER(S) WITH GET SUBROUTINE
      025506 000404      BR    40$            ;GO TO 40$ IF NO ERROR
      025510 000240      NOP                    ;RETURN HERE IF ERROR
      025512 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      025514 000137 025620      JMP   80$            ;GO TO 80$ IF ERROR
1681 025520
1682
1683                ;VERIFY RESULTS OF READ COMMAND
1684 025520 004737 040106      JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      025524 000405      BR    50$            ;GO TO 50$ IF NO ERROR
      025526 000240      NOP                    ;RETURN HERE IF ERROR
      025530 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      025532 004736      JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      025534 000137 025620      JMP   80$            ;GO TO 80$ IF ERROR
1685 025540
1686 025540 004737 052622      JSR   PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
      025544 000405      BR    60$            ;GO TO 60$ IF NO ERROR
      025546 000240      NOP                    ;RETURN HERE IF ERROR
      025550 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      025552 004736      JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      025554 000137 025620      JMP   80$            ;GO TO 80$ IF ERROR
1687 025560
1688 025560 004737 040740      JSR   PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
      025564 000405      BR    70$            ;GO TO 70$ IF NO ERROR
      025566 000240      NOP                    ;RETURN HERE IF ERROR
      025570 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      025572 004736      JSR   PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      025574 000137 025620      JMP   80$            ;GO TO 80$ IF ERROR
1689 025600
1690
1691                ;INCREMENT ADDRESS AND READ NEXT CYLINDER
1692 025600 005237 001444      INC   RMDCO          ;ADVANCE CYLINDER COUNT
1693 025604 023727 001444 002000  CMP   RMDCO,#1024.    ;DONE ALL CYLINDERS??
1694 025612 002002      BGE  80$            ;YES!!
  
```



```

1695 025614 000137 025374          JMP      10$          ;GO DO NEXT SECTOR
1696 025620          80$:
1697
1698          ;*****
          ;*TEST 27      FORMAT AT OFFSET
          ;*****
          TST27:
          SCOPE          ;SCOPE CALL
          NOP            ;START OF TEST
          MOV      #STACK,SP  ;INITIALIZE STACK POINTER
          MOV      $BASE,R0   ;R0 = UNIBUS ADDRESS
          MOV      TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
          MOV      #27,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

1699          ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
1700          MOV      #0,RMDCO  ;CYLINDER = 0
1701 025646 012737 000000 001444  MOV      #0,RMDAO  ;TRACK = 0, SECTOR = 0
1702 025654 012737 000000 001416  MOV      #FMT16,RMOFO ;16 BIT FORMAT
1703 025662 012737 010000 001442  MOV      #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
1704 025670 012737 177376 001412  MOV      #BUFONE,RMBAO ;DATA BUFFER ADDRESS
1705 025676 012737 101206 001414  MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA
1706 025704 012737 000063 001410
1707
1708          ;VERIFY THAT SECTOR IS NOT BAD
          JSR      PC,BADSCT  ;CALL BAD SECTOR MODULE
          BR      10$        ;GO TO 10$ IF NO ERROR
          TYPE      ,SCTMSG  ;TYPE BAD SECTOR MESSAGE
          EMT          ;ERROR # DEFINED BY BADSCT SUBROUTINE
          JMP      190$      ;GO TO 190$ IF ERROR

1709          10$:
1710 025732 012737 001416 001174  MOV      #RMDAO,$TMP0  ;USE SECTOR FOR DATA PATTERN
1711 025740 012737 000001 001176  MOV      #1,$TMP1
1712 025746 004737 036224  JSR      PC,GENBUF    ;GO GENERATE DATA BUFFER
1713 025752
1714
1715          20$:
1716 025752 004737 033352  ;PREPARE DEVICE FOR DATA TRANSFER
          JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
          .WORD 154130      ;TASK DESCRIPTOR AS FOLLOWS:
          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
          ;CLEAR CONTROLLER & SELECT DEVICE
          ;VERIFY CONTROLLER CLEAR OPERATION
          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
          ;VERIFY PACK ACKNOWLEDGE
          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
          ;VERIFY RECALIBRATION
          ;GO TO 30$ IF NO ERROR
          ;RETURN HERE IF ERROR
          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
          ;GO TO 190$ IF ERROR

          BR      30$
          NOP
          EMT
          JMP      190$

1717          30$:
1718
1719          ;OFFSET DEVICE FOR WRITE
1720 025772 012737 000015 001410  MOV      #OFFSET!GO,RMCS10 ;CHANGE TO OFFSET COMMAND
1721 026000 012702 001551  MOV      #PUTINX,R2      ;WRITE PUT INDEX TABLE
1722 026004 112722 000006  MOVB     #RMDA,(R2)+
1723 026010 112722 000034  MOVB     #RMDC,(R2)+
1724 026014 112722 000032  MOVB     #RMOF,(R2)+
1725 026020 112722 000000  MOVB     #RMCS1,(R2)+
  
```

```

1726 026024 112712 000200      MOVB    #200,(R2)      ;TERMINATE TABLE
1727
1728 026030 004737 037360      JSR     PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026034 000404      BR      35$          ;GO TO 35$ IF NO ERROR
      026036 000240      NOP                     ;RETURN HERE IF ERROR
      026040 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      026042 000137 026532      JMP     190$         ;GO TO 190$ IF ERROR
1729 026046
      35$:
1730
1731      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND AT OFFSET
1732 026046 012737 000063 001410      MOV     #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
1733 026054 012702 001551      MOV     #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
1734 026060 112722 000002      MOVB   #RMWC,(R2)+
1735 026064 112722 000004      MOVB   #RMBA,(R2)+
1736 026070 112722 000000      MOVB   #RMCS1,(R2)+
1737 026074 112722 000200      MOVB   #200,(R2)+
1738
1739 026100 004737 037360      JSR     PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026104 000404      BR      80$          ;GO TO 80$ IF NO ERROR
      026106 000240      NOP                     ;RETURN HERE IF ERROR
      026110 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      026112 000137 026532      JMP     190$         ;GO TO 190$ IF ERROR
1740 026116
      80$:
1741
1742      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1743 026116 004737 037024      JSR     PC,GETSTS
1744 026122 004737 037722      JSR     PC,TIMOUT    ;WAIT FOR COMMAND TO COMPLETE
1745
1746 026126 004737 037110      JSR     PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      026132 000404      BR      90$          ;GO TO 90$ IF NO ERROR
      026134 000240      NOP                     ;RETURN HERE IF ERROR
      026136 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      026140 000137 026532      JMP     190$         ;GO TO 190$ IF ERROR
1747 026144
      90$:
1748
1749      ;VERIFY RESULTS OF WRITE COMMAND
1750 026144 004737 040106      JSR     PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      026150 000405      BR      100$         ;GO TO 100$ IF NO ERROR
      026152 000240      NOP                     ;RETURN HERE IF ERROR
      026154 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      026156 004736      JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      026160 000137 026532      JMP     190$         ;GO TO 190$ IF ERROR
1751 026164
      100$:
1752 026164 004737 052622      JSR     PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
      026170 000405      BR      110$         ;GO TO 110$ IF NO ERROR
      026172 000240      NOP                     ;RETURN HERE IF ERROR
      026174 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      026176 004736      JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      026200 000137 026532      JMP     190$         ;GO TO 190$ IF ERROR
1753 026204
      110$:
1754 026204 004737 040740      JSR     PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
      026210 000405      BR      120$         ;GO TO 120$ IF NO ERROR
      026212 000240      NOP                     ;RETURN HERE IF ERROR
      026214 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      026216 004736      JSR     PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      026220 000137 026532      JMP     190$         ;GO TO 190$ IF ERROR
1755 026224
      120$:

```

```

1756 ;OFFSET DEVICE FOR READ
1757 026224 012737 000015 001410 MOV #OFFSET!GO,RMCS10 ;CHANGE TO OFFSET COMMAND
1758 026232 012702 001551 MOV #PUTINX,R2 ;WRITE PUT INDEX TABLE
1759 026236 112722 000006 MOVB #RMDA,(R2)+
1760 026242 112722 000034 MOVB #RMDC,(R2)+
1761 026246 112722 000032 MOVB #RMOF,(R2)+
1762 026252 112722 000000 MOVB #RMCS1,(R2)+
1763 026256 112722 000200 MOVB #200,(R2)+
1764
1765 026262 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
026266 000404 BR 125$ ;GO TO 125$ IF NO ERROR
026270 000240 NOP ;RETURN HERE IF ERROR
026272 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
026274 000137 026532 JMP 190$ ;GO TO 190$ IF ERROR
1766 026300 125$:
1767
1768
1769 ;READ HEADER AND DATA AT OFFSET
1770 026300 012737 000073 001410 MOV #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
1771 026306 012737 102212 001414 MOV #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
1772 026314 012702 001551 MOV #PUTINX,R2 ;WRITE PUT INDEX TABLE
1773 026320 112722 000002 MOVB #RMWC,(R2)+
1774 026324 112722 000004 MOVB #RMBA,(R2)+
1775 026330 112722 000000 MOVB #RMCS1,(R2)+
1776 026334 112722 000200 MOVB #200,(R2)+
1777
1778 026340 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
026344 000404 BR 130$ ;GO TO 130$ IF NO ERROR
026346 000240 NOP ;RETURN HERE IF ERROR
026350 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
026352 000137 026532 JMP 190$ ;GO TO 190$ IF ERROR
1779 026356 130$:
1780
1781 ;WAIT FOR READ TO COMPLETE AND GET STATUS
1782 026356 004737 037722 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
1783
1784 026362 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
026366 000404 BR 140$ ;GO TO 140$ IF NO ERROR
026370 000240 NOP ;RETURN HERE IF ERROR
026372 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
026374 000137 026532 JMP 190$ ;GO TO 190$ IF ERROR
1785 026400 140$:
1786
1787 ;VERIFY THE RESULTS OF READ OPERATION
1788 026400 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
026404 000405 BR 150$ ;GO TO 150$ IF NO ERROR
026406 000240 NOP ;RETURN HERE IF ERROR
026410 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
026412 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
026414 000137 026532 JMP 190$ ;GO TO 190$ IF ERROR
1789 026420 150$:
1790 026420 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
026424 000405 BR 160$ ;GO TO 160$ IF NO ERROR
026426 000240 NOP ;RETURN HERE IF ERROR
026430 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
026432 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
026434 000137 026532 JMP 190$ ;GO TO 190$ IF ERROR

```

```

1791 026440
1792 026440 004737 040740
      026444 000405
      026446 000240
      026450 104000
      026452 004736
      026454 000137 026532
1793 026460
1794
1795
1796 026460 004737 036462
      026464 101206
      026466 102212
      026470 000404
      026472 000240
      026474 104000
      026476 000137 026532
1797 026502
1798 026502 032737 000200 001442
1799 026510 001010
1800 026512 052737 000200 001442
1801 026520 012737 101206 001414
1802 026526 000137 025752
1803 026532
1804
1805

      026532
      026532 000004
      026534 000240
      026536 012706 001100
      026542 013700 001276
      026546 013701 001464
      026552 012737 000030 001226
1806
1807
1808 026560 012737 000000 001444
1809 026566 012737 000000 001416
1810 026574 012737 010000 001442
1811 026602 012737 177376 001412
1812 026610 012737 101206 001414
1813 026616 012737 000062 001410
1814
1815
      026624 004737 034276
      026630 000405
      026632 104401 063120
      026636 104000
      026640 000137 027354
1816 026644
1817 026644 012737 001416 001174
1818 026652 012737 000001 001176
1819 026660 004737 036224
1820 026664
1821
1822

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

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

180$:
      BIT #OFD,RMOFO ;DONE BOTH OFFSETS??
      BNE 190$ ;YES!!
      BIS #OFD,RMOFO ;SET FORWARD OFFSET
      MOV #BUFO1E,RMBAO ;CHANGE DATA BUFFER
      JMP 20$

190$:
;*****
;*TEST 30 IVC FORMAT TEST
;*****
TST30:
      SCOPE ;SCOPE CALL
      NOP ;START OF TEST
      MOV #STACK,SP ;INITIALIZE STACK POINTER
      MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
      MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
      MOV #30,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
      MOV #0,RMDCO ;CYLINDER = 0
      MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
      MOV #FMT16,RMOFO ;16 BIT FORMAT
      MOV #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
      MOV #BUFO1E,RMBAO ;DATA BUFFER ADDRESS
      MOV #WH,RMCS10 ;WRITE HEADER AND DATA

;VERIFY THAT SECTOR IS NOT BAD
      JSR PC,BADSCT ;CALL BAD SECTOR MODULE
      BR 10$ ;GO TO 10$ IF NO ERROR
      TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
      EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
      JMP 190$ ;GO TO 190$ IF ERROR

10$:
      MOV #RMDAO,$TMPO ;USE SECTOR FOR DATA PATTERN
      MOV #1,$TMP1
      JSR PC,GENBUF ;GO GENERATE DATA BUFFER

20$:
;PREPARE DEVICE FOR DATA TRANSFER

```

```

1823 026664 004737 033352      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      026670 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 30$ IF NO ERROR
                                          ;RETURN HERE IF ERROR
                                          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                          ;GO TO 190$ IF ERROR

      026672 000404          BR     30$
      026674 000240          NOP
      026676 104000          EMT
      026700 000137 027354    JMP    190$

1824 026704          30$:
1825
1826          ;RESET VOLUME VALID BY SETTING AND RESETTING DIAGNOSTIC MODE
1827 026704 012737 000001 001434  MOV    #DMD,RMMR10    ;SET DIAGNOSTIC MODE
1828 026712 112737 000024 001551  MOVB  #RMMR1,PUTINX   ;WRITE REGISTER INDEX TABLE
1829 026720 112737 000200 001552  MOVB  #200,PUTINX+1
1830
1831 026726 004737 037360      JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      026732 000404          BR     35$           ;GO TO 35$ IF NO ERROR
      026734 000240          NOP           ;RETURN HERE IF ERROR
      026736 104000          EMT           ;ERROR # DEFINED BY PUT SUBROUTINE
      026740 000137 027354    JMP    190$           ;GO TO 190$ IF ERROR
1832 026744          35$:
1833
1834          ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1835 026744 012737 000063 001410  MOV    #WH!GO,RMCS10  ;WRITE HEADER AND DATA
1836 026752 012737 000000 001434  MOV    #0,RMMR10     ;RESET DIAGNOSTIC MODE
1837 026760 012702 001551          MOV    #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
1838 026764 112722 000024          MOVB  #RMMR1,(R2)+
1839 026770 112722 000006          MOVB  #RMDA,(R2)+
1840 026774 112722 000034          MOVB  #RMDC,(R2)+
1841 027000 112722 000032          MOVB  #RMOF,(R2)+
1842 027004 112722 000002          MOVB  #RMWC,(R2)+
1843 027010 112722 000004          MOVB  #RMBA,(R2)+
1844 027014 112722 000000          MOVB  #RMCS1,(R2)+
1845 027020 112722 000200          MOVB  #200,(R2)+
1846
1847 027024 004737 037360      JSR    PC,PUT         ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027030 000404          BR     80$           ;GO TO 80$ IF NO ERROR
      027032 000240          NOP           ;RETURN HERE IF ERROR
      027034 104000          EMT           ;ERROR # DEFINED BY PUT SUBROUTINE
      027036 000137 027354    JMP    190$           ;GO TO 190$ IF ERROR
1848 027042          80$:
1849
1850          ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1851 027042 004737 037024      JSR    PC,GETSTS
1852 027046 004737 037722      JSR    PC,TIMOUT     ;WAIT FOR COMMAND TO COMPLETE
1853
1854 027052 004737 037110      JSR    PC,GET         ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027056 000404          BR     90$           ;GO TO 90$ IF NO ERROR
      027060 000240          NOP           ;RETURN HERE IF ERROR
      027062 104000          EMT           ;ERROR # DEFINED BY GET SUBROUTINE
      027064 000137 027354    JMP    190$           ;GO TO 190$ IF ERROR
1855 027070          90$:

```

```

1856
1857
1858 027070 004737 040106      ;VERIFY RESULTS OF WRITE COMMAND
      027074 000405             JSR    PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      027076 000240             BR     100$           ;GO TO 100$ IF NO ERROR
      027100 104000             NOP                    ;RETURN HERE IF ERROR
      027102 004736             EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      027104 000137 027354      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      1859 027110 000137 027354  JMP    190$           ;GO TO 190$ IF ERROR
100$:
1860 027110 032737 010000 001376 BIT    #IVC,RMER21    ;IS IVC STATUS SET??
1861 027116 001012             BNE    110$           ;YES!!
1862 027120 013737 001376 001142 MOV    RMER21,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
1863 027126 013737 001376 001140 MOV    RMER21,$GDDAT ;EXPECTED STATUS
1864 027134 052737 010000 001140 BIS    #IVC,$GDDAT
1865 027142 104342             EMT    342
1866 027144
1867 027144 004737 040740      110$:
      027150 000405             JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
      027152 000240             BR     120$           ;GO TO 120$ IF NO ERROR
      027154 104000             NOP                    ;RETURN HERE IF ERROR
      027156 004736             EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      027160 000137 027354      JSR    PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
      1868 027164 000137 027354  JMP    190$           ;GO TO 190$ IF ERROR
120$:
1869
1870
1871 027164 004737 033352      ;CLEAR IVC ERROR
      027170 040100             JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
      027172 000404             .WORD 040100        ;TASK DESCRIPTOR AS FOLLOWS:
      027174 000240             BR     125$           ;CLEAR CONTROLLER & SELECT DEVICE
      027176 104000             NOP                    ;VERIFY CONTROLLER CLEAR OPERATION
      027200 000137 027354      EMT                    ;GO TO 125$ IF NO ERROR
      1872 027204 000137 027354  JMP    190$           ;RETURN HERE IF ERROR
      1873
      1874
      1875 027204 012737 000073 001410 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
      1876 027212 012737 102212 001414 MCV    #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
      1877
      1878 027220 004737 037360      MOV    #BUFTWO,RMBAO ;CHANGE BUS ADDRESS
      027224 000404             JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027226 000240             BR     130$           ;GO TO 130$ IF NO ERROR
      027230 104000             NOP                    ;RETURN HERE IF ERROR
      027232 000137 027354      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      1879 027236 000137 027354  JMP    190$           ;GO TO 190$ IF ERROR
130$:
1880
1881
1882 027236 004737 037024      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
      1883 027242 004737 037722      JSR    PC,GETSTS     ;WAIT FOR READ TO COMPLETE
      1884
      1885 027246 004737 037110      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027252 000404             BR     140$           ;GO TO 140$ IF NO ERROR
      027254 000240             NOP                    ;RETURN HERE IF ERROR
      027256 104000             EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      027260 000137 027354      JMP    190$           ;GO TO 190$ IF ERROR
140$:
1886 027264
1887

```

```

1888
1889 027264 004737 040106      ;VERIFY THE RESULTS OF READ OPERATION
      027270 000405              JSR   PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
      027272 000240              BR    150$          ;GO TO 150$ IF NO ERROR
      027274 104000              NOP                   ;RETURN HERE IF ERROR
      027276 004736              EMT                   ;ERROR # DEFINED BY PRIERR SUBROUTINE
      027300 000137 027354      JSR   PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      1890 027304 000137 027354  JMP   190$          ;GO TO 190$ IF ERROR
      1891 027304 032737 010000 001376 150$: BIT   #IVC,RMER2I   ;IS IVC STATUS SET??
      1892 027312 001012              BNE  160$          ;YES!!
      1893 027314 013737 001376 001142  MOV  RMER2I,$BDDAT ;RECEIVED STATUS FOR TYPEOUT
      1894 027322 013737 001376 001140  MOV  RMER2I,$GDDAT ;EXPECTED STATUS
      1895 027330 052737 010000 001140  BIS  #IVC,$GDDAT
      1896 027336 104342              EMT   342
      1897 027340
      1898 027340 004737 040740      160$: JSR  PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      027344 000403              BR    170$          ;GO TO 170$ IF NO ERROR
      027346 000240              NOP                   ;RETURN HERE IF ERROR
      027350 104000              EMT                   ;ERROR # DEFINED BY SECERR SUBROUTINE
      027352 004736              JSR  PC,@(SP)+     ;GO BACK FOR MORE ERROR CHECKS
      1899 027354
      1900
      1901 027354
      1902
      1903
      ;*****
      ;*TEST 31      FORMAT ERROR TEST
      ;*****
      TST31:
      027354
      027354 000004              SCOPE                ;SCOPE CALL
      027356 000240              NOP                  ;START OF TEST
      027360 012706 001100      MOV  #STACK,SP      ;INITIALIZE STACK POINTER
      027364 013700 001276      MOV  $BASE,R0       ;R0 = UNIBUS ADDRESS
      027370 013701 001464      MOV  TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
      027374 012737 000031 001226  MOV  #31,$TESTN     ;:SET TEST NUMBER IN APT MAIL BOX
      1904
      1905
      1906 027402 012737 000000 001444  ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
      1907 027410 012737 000000 001416  MOV  #0,RMDCO       ;CYLINDER = 0
      1908 027416 012737 000000 001442  MOV  #0,RMDAO       ;TRACK = 0, SECTOR = 0
      1909 027424 012737 177376 001412  MOV  #0,RMOFO       ;18 BIT FORMAT
      1910 027432
      1911 027432 012737 101206 001414  MOV  #-258.,RMWCO   ;2 + 256 WORDS (2'S COMP)
      1912 027440 012737 000062 001410  5$:  MOV  #BUFONE,RMBAO ;DATA BUFFER ADDRESS
      1913
      1914
      1915 027446 004737 034276      ;VERIFY THAT SECTOR IS NOT BAD
      027452 000405              JSR  PC,BADSCT     ;CALL BAD SECTOR MODULE
      027454 104401 063120      BR    10$          ;GO TO 10$ IF NO ERROR
      027460 104000              TYPE  ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
      027462 000137 030136      EMT                   ;ERROR # DEFINED BY BADSCT SUBROUTINE
      1916 027466 012737 064620 001174  JMP   180$          ;GO TO 180$ IF ERROR
      1917 027474 012737 000001 001176  10$: MOV  #ZEROS,$TMPO  ;USE ALL ZEROS DATA PATTERN
      1918 027502 004737 036224      MOV  #1,$TMP1
      1919 027506
      1920
      1921
      20$: JSR  PC,GENBUF    ;GO GENERATE DATA BUFFER
      ;PREPARE DEVICE FOR DATA TRANSFER

```

```

1922 027506 004737 033352      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
      027512 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
      027514 000404          BR     30$          ;GO TO 30$ IF NO ERROR
      027516 000240          NOP                    ;RETURN HERE IF ERROR
      027520 104000          EMT                    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
      027522 000137 030136    JMP    180$         ;GO TO 180$ IF ERROR
1923 027526          30$:
1924
1925          ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
1926 027526 012737 000063 001410  MOV    #WH!GO,RMCS10 ;WRITE HEADER AND DATA
1927 027534 012702 001551      MOV    #PUTINX,R2    ;WRITE REGISTER INDEX TABLE
1928 027540 112722 000006      MOVB  #RMDA,(R2)+
1929 027544 112722 000034      MOVB  #RMDC,(R2)+
1930 027550 112722 000032      MOVB  #RMOF,(R2)+
1931 027554 112722 000002      MOVB  #RMWC,(R2)+
1932 027560 112722 000004      MOVB  #RMBA,(R2)+
1933 027564 112722 000000      MOVB  #RMCS1,(R2)+
1934 027570 112722 000200      MOVB  #200,(R2)+
1935
1936 027574 004737 037360      JSR    PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      027600 000404          BR     80$          ;GO TO 80$ IF NO ERROR
      027602 000240          NOP                    ;RETURN HERE IF ERROR
      027604 104000          EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      027606 000137 030136    JMP    180$         ;GO TO 180$ IF ERROR
1937 027612          80$:
1938
1939          ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
1940 027612 004737 037024      JSR    PC,GETSTS
1941 027616 004737 037722      JSR    PC,TIMOUT    ;WAIT FOR COMMAND TO COMPLETE
1942
1943 027622 004737 037110      JSR    PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
      027626 000404          BR     90$          ;GO TO 90$ IF NO ERROR
      027630 000240          NOP                    ;RETURN HERE IF ERROR
      027632 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      027634 000137 030136    JMP    180$         ;GO TO 180$ IF ERROR
1944 027640          90$:
1945
1946          ;VERIFY RESULTS OF WRITE COMMAND
1947 027640 004737 040106      JSR    PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
      027644 000405          BR     100$        ;GO TO 100$ IF NO ERROR
      027646 000240          NOP                    ;RETURN HERE IF ERROR
      027650 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      027652 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
      027654 000137 030136    JMP    180$         ;GO TO 180$ IF ERROR
1948 027660          100$:
1949 027660 004737 052622      JSR    PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
      027664 000405          BR     110$        ;GO TO 110$ IF NO ERROR
      027666 000240          NOP                    ;RETURN HERE IF ERROR
      027670 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      027672 004736          JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
    
```



```

1950 027674 000137 030136          JMP      180$          ;GO TO 180$ IF ERROR
1951 027700 004737 040740          110$:   JSR      PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
        027704 000405                BR       120$          ;GO TO 120$ IF NO ERROR
        027706 000240                NOP                      ;RETURN HERE IF ERROR
        027710 104000                EMT                      ;ERROR # DEFINED BY SECERR SUBROUTINE
        027712 004736                JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
        027714 000137 030136          JMP      180$          ;GO TO 180$ IF ERROR
1952 027720 005137 001442          120$:   COM      RMOFO          ;CHANGE TO OPPOSITE FORMAT
1953 027724 042737 167777 001442    BIC      #^C<FMT16>,RMOFO
1954 027732 012737 102212 001414    MOV      #BUFTWO,RMBAO  ;CHANGE BUS ADDRESS
1955 027740 012737 000073 001410    MOV      #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
1961 027746 004737 037360          JSR      PC,PUT        ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
        027752 000404                BR       130$          ;GO TO 130$ IF NO ERROR
        027754 000240                NOP                      ;RETURN HERE IF ERROR
        027756 104000                EMT                      ;ERROR # DEFINED BY PUT SUBROUTINE
        027760 000137 030136          JMP      180$          ;GO TO 180$ IF ERROR
1962 027764 004737 037722          130$:   JSR      PC,TIMOUT      ;WAIT FOR READ TO COMPLETE AND GET STATUS
1963 027770 004737 037110          JSR      PC,GET        ;GO READ REGISTER(S) WITH GET SUBROUTINE
1964 027774 000404                BR       140$          ;GO TO 140$ IF NO ERROR
1965 027776 000240                NOP                      ;RETURN HERE IF ERROR
1966 030000 104000                EMT                      ;ERROR # DEFINED BY GET SUBROUTINE
1967 030002 000137 030136          JMP      180$          ;GO TO 180$ IF ERROR
1968 030006 004737 040106          140$:   JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
1969 030012 000405                BR       150$          ;GO TO 150$ IF NO ERROR
1970 030014 000240                NOP                      ;RETURN HERE IF ERROR
1971 030016 104000                EMT                      ;ERROR # DEFINED BY PRIERR SUBROUTINE
1972 030020 004736                JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
1973 030022 000137 030136          JMP      180$          ;GO TO 180$ IF ERROR
1974 030026 032737 000020 001350    150$:   ;VERIFY THE RESULTS OF READ OPERATION
1975 030034 001022                BIT      #FER,RMER11    ;IS FORMAT ERROR SET??
1976 030036 004737 052622          BNE     160$          ;YES!!
1977 030042 000405                JSR      PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
1978 030044 000240                BR       155$          ;GO TO 155$ IF NO ERROR
1979 030046 104000                NOP                      ;RETURN HERE IF ERROR
1980 030050 004736                EMT                      ;ERROR # DEFINED BY DTASTS SUBROUTINE
1981 030052 000137 030136          JSR      PC,@(SP)+      ;GO BACK FOR MORE ERROR CHECKS
1982 030056 013737 001350 001142    JMP      180$          ;GO TO 180$ IF ERROR
1983 030064 013737 001350 001140    155$:   MOV      RMER11,$BDDAT  ;BAD DATA FOR TYPEOUT
1984 030072 052737 000020 001140    MOV      RMER11,$GDDAT ;EXPECTED DATA
1985 030074 000020 001140    BIS      #FER,$GDDAT
  
```

1983 030100 104343
 1984 030102
 1985 030102 004737 040740
 030106 000405
 030110 000240
 030112 104000
 030114 004736
 030116 000137 030136
 1986 030122
 1987 030122 032737 010000 001442
 1988
 1989 030130 001402
 1990 030132 000137 027432
 1991 030136
 1992
 1993

```

160$: EMT 343
      JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
      BR 170$ ;GO TO 170$ IF NO ERROR
      NOP ;RETURN HERE IF ERROR
      EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
      JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
      JMP 180$ ;GO TO 180$ IF ERROR

170$: BIT #FMT16,RMOFO ;TEST WRITE 16 BIT FORMAT AND
      ;READ 18 BIT MODE ?
      BEQ 180$ ;BR IF YES
      JMP 5$ ;TEST AGAIN

180$:

```

```

*****
;*TEST 32 FORMAT HCE, FIRST HEADER WORD
*****
TST32:

```

030136
 030136 000004
 030140 000240
 030142 012706 001100
 030146 013700 001276
 030152 013701 001464
 030156 012737 000032 001226
 1994
 1995 030164 012737 000001 031134
 1996
 1997
 1998 030172
 1999 030172 012737 010000 001442
 2000 030200 012737 000000 001444
 2001 030206 012737 000000 001416
 2002 030214 012737 177376 001412
 2003 030222 012737 101206 001414
 2004 030230 012737 000062 001410
 2005
 2006
 030236 004737 034276
 030242 000405
 030244 104401 063120
 030250 104000
 030252 000137 031306
 2007 030256
 2008 030256 012737 064620 001174
 2009 030264 012737 000001 001176
 2010 030272 004737 036224
 2011 030276
 2012
 2013
 2014 030276 004737 033352
 030302 154130

```

SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #32,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

MOV #1,230$ ;INIT BIT POSITION

;SETUP PARAMETERS FOR GENERATING DATA BUFFER
10$: MOV #FMT16,RMOFO ;16 BIT FORMAT
      MOV #0,RMDCO ;CYLINDER = 0
      MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
      MOV #-258.,RMWCO ;2 + 256 WORDS (2'S COMP)
      MOV #BUFONE,RMBAO ;DATA BUFFER ADDRESS
      MOV #WH,RMCS10 ;WRITE HEADER AND DATA

;VERIFY THAT SECTOR IS NOT BAD
JSR PC,BADSCT ;CALL BAD SECTOR MODULE
BR 20$ ;GO TO 20$ IF NO ERROR
TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
EMT ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 270$ ;GO TO 270$ IF ERROR

20$: MOV #ZEROS,$TMP0 ;USE ALL ZEROS DATA PATTERN
      MOV #1,$TMP1
      JSR PC,GENBUF ;GO GENERATE DATA BUFFER

30$:

;PREPARE DEVICE FOR DATA TRANSFER
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
.WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
;SELECT DEVICE & VERIFY DEVICE AVAILABLE
;CLEAR CONTROLLER & SELECT DEVICE
;VERIFY CONTROLLER CLEAR OPERATION
;PACK ACKNOWLEDGE IF VOLUME NOT VALID
;VERIFY PACK ACKNOWLEDGE

```

```

;RECALIBRATE IF "SKI" OR "PIP" IS SET
;VERIFY RECALIBRATION
;GO TO 40% IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 270% IF ERROR
030304 000404 BR 40%
030306 000240 NOP
030310 104000 EMT
030312 000137 031306 JMP 270%
2015 030316 40%:
2016
2017 ;COMPLEMENT DATA BIT IN FIRST HEADER WORD
2018 030316 033737 031134 101206 BIT 230%,BUFONE ;IS BIT IN HEADER ON??
2019 030324 001404 BEQ 50% ;NO!!
2020 030326 043737 031134 101206 BIC 230%,BUFONE ;RESET BIT IN HEADER
2021 030334 000403 BR 60%
2022 030336 053737 031134 101206 50%: BIS 230%,BUFONE ;SET BIT IN HEADER
2023
2024 ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
2025 030344 60%:
2026 030344 012737 000063 001410 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA
2027 030352 012702 001551 MOV #PUTINX,R2 ;WRITE REGISTER INDEX TABLE
2028 030356 112722 000006 MOVB #RMDA,(R2)+
2029 030362 112722 000034 MOVB #RMDC,(R2)+
2030 030366 112722 000032 MOVB #RMOF,(R2)+
2031 030372 112722 000002 MOVB #RMWC,(R2)+
2032 030376 112722 000004 MOVB #RMBA,(R2)+
2033 030402 112722 000000 MOVB #RMCS1,(R2)+
2034 030406 112722 000200 MOVB #200,(R2)+
2035
2036 030412 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030416 000404 BR 70% ;GO TO 70% IF NO ERROR
030420 000240 NOP ;RETURN HERE IF ERROR
030422 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
030424 000137 031132 JMP 220% ;GO TO 220% IF ERROR
2037 030430 70%:
2038
2039 ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
2040 030430 004737 037024 JSR PC,GETSTS
2041 030434 004737 037722 JSR PC,TIMOUT ;WAIT FOR COMMAND TO COMPLETE
2042
2043 030440 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
030444 000404 BR 80% ;GO TO 80% IF NO ERROR
030446 000240 NOP ;RETURN HERE IF ERROR
030450 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
030452 000137 031132 JMP 220% ;GO TO 220% IF ERROR
2044 030456 80%:
2045
2046 ;VERIFY RESULTS OF WRITE COMMAND
2047 030456 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
030462 000405 BR 90% ;GO TO 90% IF NO ERROR
030464 000240 NOP ;RETURN HERE IF ERROR
030466 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
030470 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030472 000137 031132 JMP 220% ;GO TO 220% IF ERROR
2048 030476 90%:
2049 030476 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
030502 000405 BR 100% ;GO TO 100% IF NO ERROR
030504 000240 NOP ;RETURN HERE IF ERROR
030506 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE

```

```

030510 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030512 000137 031132 JMP 220$ ;GO TO 220$ IF ERROR
2050 030516 100$:
2051 030516 004737 040740 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
030522 000405 BR 110$ ;GO TO 110$ IF NO ERROR
030524 000240 NOP ;RETURN HERE IF ERROR
030526 104000 EMT ;ERROR # DEFINED BY SECERR SUBROUTINE
030530 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030532 000137 031132 JMP 220$ ;GO TO 220$ IF ERROR
2052 030536 110$:
2053
2054 ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
2055 030536 012737 000073 001410 MOV #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
2056 030544 012737 102212 001414 MOV #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
2057
2058 030552 004737 037360 JSR PC,PUT ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
030556 000404 BR 120$ ;GO TO 120$ IF NO ERROR
030560 000240 NOP ;RETURN HERE IF ERROR
030562 104000 EMT ;ERROR # DEFINED BY PUT SUBROUTINE
030564 000137 031132 JMP 220$ ;GO TO 220$ IF ERROR
2059 030570 120$:
2060
2061 ;WAIT FOR READ TO COMPLETE AND GET STATUS
2062 030570 004737 037722 JSR PC,TIMOUT ;WAIT FOR READ TO COMPLETE
2063
2064 030574 004737 037110 JSR PC,GET ;GO READ REGISTER(S) WITH GET SUBROUTINE
030600 000404 BR 130$ ;GO TO 130$ IF NO ERROR
030602 000240 NOP ;RETURN HERE IF ERROR
030604 104000 EMT ;ERROR # DEFINED BY GET SUBROUTINE
030606 000137 031132 JMP 220$ ;GO TO 220$ IF ERROR
2065 030612 130$:
2066
2067 ;VERIFY THE RESULTS OF READ OPERATION
2068 030612 004737 040106 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
030616 000405 BR 140$ ;GO TO 140$ IF NO ERROR
030620 000240 NOP ;RETURN HERE IF ERROR
030622 104000 EMT ;ERROR # DEFINED BY PRIERR SUBROUTINE
030624 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030626 000137 031132 JMP 220$ ;GO TO 220$ IF ERROR
2069 030632 140$:
2070 030632 032737 140000 031134 BIT #MSE!USE,230$ ;SHOULD BSE BE SET ?
2071 030640 001033 BNE 160$ ;YES !!
2072 030642 032737 010000 031134 BIT #FMT16,230$ ;IS THIS FER ??
2073 030650 001456 BEQ 180$ ;NO !!
2074
2075 ;VERIFY THAT FER IS SET
2076 030652 032737 000020 001350 BIT #FER,RMER11 ;IS FER SET ??
2077 030660 001107 BNE 200$ ;YES !!
2078
2079 030662 004737 052622 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
030666 000405 BR 150$ ;GO TO 150$ IF NO ERROR
030670 000240 NOP ;RETURN HERE IF ERROR
030672 104000 EMT ;ERROR # DEFINED BY DTASTS SUBROUTINE
030674 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
030676 000137 031132 JMP 220$ ;GO TO 220$ IF ERROR
2080 030702 150$:
2081 030702 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
  
```

```

2082 030710 013737 001350 001140      MOV      RMER11,$GDDAT      ;EXPECTED STATUS
2083 030716 052737 000020 001140      BIS      #FER,$GDDAT
2084 030724 104343                      EMT      343
2085 030726 000501                      BR       220$
2086
2087                                     ;VERIFY THAT BSE IS SET
2088 030730                                     160$:
2089 030730 032737 100000 001376      BIT      #BSE,RMER21      ;IS BSE SET ??
2090 030736 001060                      BNE     200$              ;YES !!
2091
2092 030740 004737 052622                      JSR     PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      030744 000405                      BR      170$            ;GO TO 170$ IF NO ERROR
      030746 000240                      NOP
      030750 104000                      EMT
      030752 004736                      JSR     PC,@(SP)+        ;ERROR # DEFINED BY DTASTS SUBROUTINE
      030754 000137 031132                      JMP     220$            ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 220$ IF ERROR
2093 030760                                     170$:
2094 030760 013737 001376 001142      MOV      RMER21,$BDDAT    ;RECEIVED STATUS
2095 030766 013737 001376 001140      MOV      RMER21,$GDDAT    ;EXPECTED STATUS
2096 030774 052737 100000 001140      BIS      #BSE,$GDDAT
2097 031002 104345                      EMT      345
2098 031004 000452                      BR       220$            ;SKIP REST OF TEST
2099
2100                                     ;VERIFY THAT HCE IS SET
2101 031006                                     180$:
2102 031006 032737 000200 001350      BIT      #HCE,RMER11     ;IS 'HCE' SET??
2103 031014 001031                      BNE     200$              ;YES!!
2104
2105 031016 004737 052622                      JSR     PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      031022 000405                      BR      190$            ;GO TO 190$ IF NO ERROR
      031024 000240                      NOP
      031026 104000                      EMT
      031030 004736                      JSR     PC,@(SP)+        ;ERROR # DEFINED BY DTASTS SUBROUTINE
      031032 000137 031132                      JMP     220$            ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 220$ IF ERROR
2106 031036                                     190$:
2107 031036 013737 001350 001142      MOV      RMER11,$BDDAT    ;RECEIVED STATUS FOR TYPEOUT
2108 031044 013737 001350 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
2109 031052 052737 000200 001140      BIS      #HCE,$GDDAT
2110 031060 012737 000001 001174      MOV      #1,$TMP0        ;GET HEADER WORD NUMBER
2111 031066 013737 031134 001176      MOV      230$,$TMP1      ;GET FAILING BIT POSITION
2112 031074 104344                      EMT      344
2113 031076 000415                      BR       220$
2114 031100                                     200$:
2115
2116                                     ;CHECK FOR OTHER ERRORS
2117 031100 004737 040740                      JSR     PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
      031104 000405                      BR      210$            ;GO TO 210$ IF NO ERROR
      031106 000240                      NOP
      031110 104000                      EMT
      031112 004736                      JSR     PC,@(SP)+        ;ERROR # DEFINED BY SECERR SUBROUTINE
      031114 000137 031132                      JMP     220$            ;GO BACK FOR MORE ERROR CHECKS
      ;GO TO 220$ IF ERROR
2118 031120                                     210$:
2119
2120                                     ;ADVANCE THE BIT POSITION AND FORMAT AGAIN IF NOT DONE
2121 031120 006337 031134                      ASL     230$              ;SHIFT TO NEXT BIT POSITION
2122 031124 001404                      BEQ     240$              ;EXIT IF DONE
2123 031126 000137 030172                      JMP     10$              ;GO DO NEXT SECTOR
    
```

```

2124
2125 031132 000465      220$: BR      270$      :JUMP TO NEXT TEST
2126 031134 000000      230$: .WORD   0          :STORAGE FOR BIT POSITION
2127
2128
2129
2130
2131 031136
2132 031136 012737 010000 001442
2133 031144 012737 177776 001412
2134 031152 012737 101206 001414
2135 031160 012737 000062 001410
2136 031166 004737 036224
2137
2138 031172 004737 033352      JSR      PC,TSTPRP      :PREPARE DEVICE FOR TEST
      031176 154130      .WORD   154130         :TASK DESCRIPTOR AS FOLLOWS:
                                                                :SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                                                :CLEAR CONTROLLER & SELECT DEVICE
                                                                :VERIFY CONTROLLER CLEAR OPERATION
                                                                :PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                                                :VERIFY PACK ACKNOWLEDGE
                                                                :RECALIBRATE IF "SKI" OR "PIP" IS SET
                                                                :VERIFY RECALIBRATION
                                                                :GO TO 250$ IF NO ERROR
                                                                :RETURN HERE IF ERROR
                                                                :ERROR # DEFINED BY TSTPRP SUBROUTINE
                                                                :GO TO 250$ IF ERROR
      031200 000404      BR      250$
      031202 000240      NOP
      031204 104000      EMT
      031206 000137 031212      JMP     250$
2139 031212
2140 031212 012737 000063 001410 250$: MOV     #WH!GO,RMCS10    :SET THE GO BIT
2141 031220 012702 001551      MOV     #PUTINX,R2      :PUT THE REGISTER ADDRESS INTO TABLE
2142 031224 112722 000006      MOVB   #RMDA,(R2)+
2143 031230 112722 000034      MOVB   #RMDC,(R2)+
2144 031234 112722 000032      MOVB   #RMOF,(R2)+
2145 031240 112722 000002      MOVB   #RMWC,(R2)+
2146 031244 112722 000004      MOVB   #RMBA,(R2)+
2147 031250 112722 000000      MOVB   #RMCS1,(R2)+
2148 031254 112722 000200      MOVB   #200,(R2)+
2149 031260 004737 037360      JSR    PC,PUT           :TERMINATOR
      031264 000404      BR     260$           :GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      031266 000240      NOP           :GO TO 260$ IF NO ERROR
      031270 104000      EMT           :RETURN HERE IF ERROR
      031272 000137 031306      JMP     270$         :ERROR # DEFINED BY PUT SUBROUTINE
                                                                :GO TO 270$ IF ERROR
2150 031276
2151 031276 000240      260$: NOP
2152 031300 004737 037722      JSR    PC,TIMOUT      :WAIT UNTIL IT FINISH
2153 031304 000240      NOP
2154 031306
2155
2156

```

```

*****
:*TEST 33      FORMAT HCE, SECOND HEADER WORD
*****
TST33:
031306
031306 000004      SCOPE          :SCOPE CALL
031310 000240      NOP           :START OF TEST
031312 012706 001100      MOV     #STACK,SP    :INITIALIZE STACK POINTER
031316 013700 001276      MOV     $BASE,R0     :R0 = UNIBUS ADDRESS
031322 013701 001464      MOV     TSTQUE,R1    :(R1) = DEVICE BEING TESTED

```

```

2157 031326 012737 000033 001226      MOV    #33,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2158 031334 012737 000001 032130      MOV    #1,190$        ;INIT BIT POSITION
2159
2160      ;SETUP PARAMETERS FOR GENERATING DATA BUFFER
2161 031342      10$:
2162 031342 012737 000000 001444      MOV    #0,RMDCO        ;CYLINDER = 0
2163 031350 012737 000000 001416      MOV    #0,RMDAO        ;TRACK = 0, SECTOR = 0
2164 031356 012737 010000 001442      MOV    #FMT16,RMOFO    ;16 BIT FORMAT
2165 031364 012737 177376 001412      MOV    #-258.,RMWCO    ;2 + 256 WORDS (2'S COMP)
2166 031372 012737 101206 001414      MOV    #BUFONE,RMBAO   ;DATA BUFFER ADDRESS
2167 031400 012737 000062 001410      MOV    #WH,RMCS10     ;WRITE HEADER AND DATA
2168
2169      ;VERIFY THAT SECTOR IS NOT BAD
031406 004737 034276      JSR    PC,BADSCT      ;CALL BAD SECTOR MODULE
031412 000405      BR     20$           ;GO TO 20$ IF NO ERROR
031414 104401 063120      TYPE   ,SCTMSG       ;TYPE BAD SECTOR MESSAGE
031420 104000      EMT                    ;ERROR # DEFINED BY BADSCT SUBROUTINE
031422 000137 032302      JMP    230$         ;GO TO 230$ IF ERROR
2170 031426      20$:
2171 031426 012737 064620 001174      MOV    #ZEROS,$TMP0   ;USE ALL ZEROS DATA PATTERN
2172 031434 012737 000001 001176      MOV    #1,$TMP1
2173 031442 004737 036224      JSR    PC,GENBUF     ;GO GENERATE DATA BUFFER
2174 031446      30$:
2175
2176      ;PREPARE DEVICE FOR DATA TRANSFER
2177 031446 004737 033352      JSR    PC,TSTPRP     ;PREPARE DEVICE FOR TEST
031452 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 40$ IF NO ERROR
                                ;RETURN HERE IF ERROR
                                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
                                ;GO TO 230$ IF ERROR
031454 000404      BR     40$
031456 000240      NOP
031460 104000      EMT
031462 000137 032302      JMP    230$
2178
2179      ;COMPLEMENT DATA BIT IN SECOND HEADER WORD
2180 031466      40$:
2181 031466 033737 032130 101210      BIT    190$,BUFONE+2  ;IS BIT IN HEADER ON??
2182 031474 001404      BEQ    50$           ;NO!!
2183 031476 043737 032130 101210      BIC    190$,BUFONE+2  ;RESET BIT IN HEADER
2184 031504 000403      BR     60$
2185 031506 053737 032130 101210      50$: BIS    190$,BUFONE+2 ;SET BIT IN HEADER
2186
2187      ;SETUP AND EXECUTE WRITE HEADER AND DATA COMMAND
2188 031514      60$:
2189 031514 012737 000063 001410      MOV    #WH!GO,RMCS10  ;WRITE HEADER AND DATA
2190 031522 012702 001551      MOV    #PUTINX,R2     ;WRITE REGISTER INDEX TABLE
2191 031526 112722 000006      MOVB   #RMDA,(R2)+
2192 031532 112722 000034      MOVB   #RMDC,(R2)+
2193 031536 112722 000032      MOVB   #RMOF,(R2)+
2194 031542 112722 000002      MOVB   #RMWC,(R2)+
2195 031546 112722 000004      MOVB   #RMBA,(R2)+
  
```

```

2196 031552 112722 000000      MOVB  #RMCS1,(R2)+
2197 031556 112722 000200      MOVB  #200,(R2)+
2198
2199 031562 004737 037360      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      031566 000404      BR    70$        ;GO TO 70$ IF NO ERROR
      031570 000240      NOP                    ;RETURN HERE IF ERROR
      031572 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      031574 000137 032126      JMP   180$       ;GO TO 180$ IF ERROR
2200 031600      70$:
2201
2202      ;SETUP INPUT REGISTER BUFFER FOR READING STATUS
2203 031600 004737 037024      JSR   PC,GETSTS
2204 031604 004737 037722      JSR   PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
2205
2206 031610 004737 037110      JSR   PC,GET      ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031614 000404      BR    80$        ;GO TO 80$ IF NO ERROR
      031616 000240      NOP                    ;RETURN HERE IF ERROR
      031620 104000      EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      031622 000137 032126      JMP   180$       ;GO TO 180$ IF ERROR
2207 031626      80$:
2208
2209      ;VERIFY RESULTS OF WRITE COMMAND
2210 031626 004737 040106      JSR   PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
      031632 000405      BR    90$        ;GO TO 90$ IF NO ERROR
      031634 000240      NOP                    ;RETURN HERE IF ERROR
      031636 104000      EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      031640 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      031642 000137 032126      JMP   180$       ;GO TO 180$ IF ERROR
2211 031646      90$:
2212 031646 004737 052622      JSR   PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
      031652 000405      BR    100$       ;GO TO 100$ IF NO ERROR
      031654 000240      NOP                    ;RETURN HERE IF ERROR
      031656 104000      EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      031660 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      031662 000137 032126      JMP   180$       ;GO TO 180$ IF ERROR
2213 031666      100$:
2214 031666 004737 040740      JSR   PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
      031672 000405      BR    110$       ;GO TO 110$ IF NO ERROR
      031674 000240      NOP                    ;RETURN HERE IF ERROR
      031676 104000      EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      031700 004736      JSR   PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
      031702 000137 032126      JMP   180$       ;GO TO 180$ IF ERROR
2215 031706      110$:
2216
2217      ;READ HEADER AND DATA FOR SECTOR JUST WRITTEN
2218 031706 012737 000073 001410      MOV   #RH!GO,RMCS10 ;READ HEADER & DATA COMMAND
2219 031714 012737 102212 001414      MOV   #BUFTWO,RMBA0 ;CHANGE BUS ADDRESS
2220
2221 031722 004737 037360      JSR   PC,PUT      ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      031726 000404      BR    120$       ;GO TO 120$ IF NO ERROR
      031730 000240      NOP                    ;RETURN HERE IF ERROR
      031732 104000      EMT                    ;ERROR # DEFINED BY PUT SUBROUTINE
      031734 000137 032126      JMP   180$       ;GO TO 180$ IF ERROR
2222 031740      120$:
2223
2224      ;WAIT FOR READ TO COMPLETE AND GET STATUS
2225 031740 004737 037722      JSR   PC,TIMOUT   ;WAIT FOR READ TO COMPLETE
  
```



```

2226
2227 031744 004737 037110      JSR   PC,GET           ;GO READ REGISTER(S) WITH GET SUBROUTINE
      031750 000404          BR    130$            ;GO TO 130$ IF NO ERROR
      031752 000240          NOP                    ;RETURN HERE IF ERROR
      031754 104000          EMT                    ;ERROR # DEFINED BY GET SUBROUTINE
      031756 000137 032126      JMP   180$            ;GO TO 180$ IF ERROR
2228 031762      130$:
2229
2230      ;VERIFY THE RESULTS OF READ OPERATION
2231 031762 004737 040106      JSR   PC,PRIERR        ;GO CHECK FOR PRIMARY ERRORS
      031766 000405          BR    140$            ;GO TO 140$ IF NO ERROR
      031770 000240          NOP                    ;RETURN HERE IF ERROR
      031772 104000          EMT                    ;ERROR # DEFINED BY PRIERR SUBROUTINE
      031774 004736          JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      031776 000137 032126      JMP   180$            ;GO TO 180$ IF ERROR
2232 032002      140$:
2233
2234      ;VERIFY THAT HEADER COMPARE ERROR IS SET
2235 032002 032737 000200 001350  BIT   #HCE,RMER1I      ;IS 'HCE' SET??
2236 032010 001031          BNE   160$            ;YES!!
2237
2238 032012 004737 052622      JSR   PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
      032016 000405          BR    150$            ;GO TO 150$ IF NO ERROR
      032020 000240          NOP                    ;RETURN HERE IF ERROR
      032022 104000          EMT                    ;ERROR # DEFINED BY DTASTS SUBROUTINE
      032024 004736          JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      032026 000137 032126      JMP   180$            ;GO TO 180$ IF ERROR
2239 032032      150$:
2240 032032 013737 001350 001142  MOV   RMER1I,$BDDAT    ;RECEIVED STATUS FOR TYPEOUT
2241 032040 013737 001350 001140  MOV   RMER1I,$GDDAT    ;EXPECTED STATUS
2242 032046 052737 000200 001140  BIS   #HCE,$GDDAT
2243 032054 012737 000002 001174  MOV   #2,$TMP0         ;GET HEADER WORD NUMBER
2244 032062 013737 032130 001176  MOV   190$,$TMP1       ;GET FAILING BIT POSITION
2245 032070 104344          EMT   344
2246 032072 000415          BR    180$
2247 032074      160$:
2248
2249      ;CHECK FOR OTHER ERRORS
2250 032074 004737 040740      JSR   PC,SECERR        ;GO CHECK FOR SECONDARY ERRORS
      032100 000405          BR    170$            ;GO TO 170$ IF NO ERROR
      032102 000240          NOP                    ;RETURN HERE IF ERROR
      032104 104000          EMT                    ;ERROR # DEFINED BY SECERR SUBROUTINE
      032106 004736          JSR   PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
      032110 000137 032126      JMP   180$            ;GO TO 180$ IF ERROR
2251 032114      170$:
2252
2253      ;ADVANCE THE BIT POSITION AND FORMAT NEXT SECTOR IF NOT DONE
2254 032114 006337 032130      ASL   190$            ;SHIFT TO NEXT BIT POSITION
2255 032120 001404          BEQ   200$            ;EXIT IF DONE
2256 032122 000137 031342      JMP   10$            ;GO DO NEXT SECTOR
2257
2258 032126 000465      180$: BR    230$        ;JUMP OVER STORAGE
2259 032130 000000      190$: .WORD 0         ;STORAGE FOR BIT POSITION
2260
2261      ;*****
2262      ;*REFORMAT SECTOR THAT WAS WRITTEN WITH BAD HEADER
2263      ;*****

```

```

2264 032132          200$:
2265 032132 012737 010000 001442  MOV #FMT16,RMOFO ;16 BIT MODE
2266 032140 012737 177776 001412  MOV #-2,RMWCO ;2 HEADER WORDS ONLY
2267 032146 012737 101206 001414  MOV #BUFONE,RMBAO ;BUFFER ADDRESS
2268 032154 012737 000062 001410  MOV #WH,RMCS10 ;WRITE HEAD AND DATA COMMAND
2269 032162 004737 036224          JSR PC,GENBUF ;SET UP THE BUFFER
2270
2271 032166 004737 033352          JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
      032172 154130          .WORD 154130 ;TASK DESCRIPTOR AS FOLLOWS:
                                          ;SELECT DEVICE & VERIFY DEVICE AVAILABLE
                                          ;CLEAR CONTROLLER & SELECT DEVICE
                                          ;VERIFY CONTROLLER CLEAR OPERATION
                                          ;PACK ACKNOWLEDGE IF VOLUME NOT VALID
                                          ;VERIFY PACK ACKNOWLEDGE
                                          ;RECALIBRATE IF "SKI" OR "PIP" IS SET
                                          ;VERIFY RECALIBRATION
032174 000404          BR 210$ ;GO TO 210$ IF NO ERROR
032176 000240          NOP ;RETURN HERE IF ERROR
032200 104000          EMT ;ERROR # DEFINED BY TSTPRP SUBROUTINE
032202 000137 032206          JMP 210$ ;GO TO 210$ IF ERROR
2272 032206          210$:
2273 032206 012737 000063 001410  MOV #WH!GO,RMCS10 ;SET GO BIT
2274 032214 012702 001551          MOV #PUTINX,R2 ;TABLE ADDRESS
2275 032220 112722 000006          MOVB #RMDA,(R2)+
2276 032224 112722 000034          MOVB #RMDC,(R2)+
2277 032230 112722 000032          MOVB #RMOF,(R2)+
2278 032234 112722 000004          MOVB #RMBA,(R2)+
2279 032240 112722 000002          MOVB #RMWC,(R2)+
2280 032244 112722 000000          MOVB #RMCS1,(R2)+
2281 032250 112722 000200          MOVB #200,(R2)+
2282 032254 004737 037360          JSR PC,PUT ;TERMINATOR
      032260 000404          BR 220$ ;GO WRITE REGISTER(S) WITH PUT SUBROUTINE
      032262 000240          NOP ;GO TO 220$ IF NO ERROR
      032264 104000          EMT ;RETURN HERE IF ERROR
      032266 000137 032302          JMP 230$ ;ERROR # DEFINED BY PUT SUBROUTINE
                                          ;GO TO 230$ IF ERROR
2283 032272          220$:
2284 032272 000240          NOP
2285 032274 004737 037722          JSR PC,TIMOUT ;WAIT FOR TIME OUT
2286 032300 000240          NOP
2287 032302          230$:
2288
  
```

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 'READY' ROUTINE.

\$EOSP: SCOPE
NOP
MOV TSTQUE,RO ;GET POINTER TO TSTQUE
ADD #2,RO ;ADJUST POINTER TO NEXT DEVICE
MOV RO,TSTQUE ;SAVE POINTER TO TSTQUE
TST (RO) ;ANY MORE DEVICES FOR TEST ?
BEQ 1\$;BR IF NO
JMP SHUT ;JUMP TO SHUT AND CHECK FOR CONTROL C
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

032340
032340 000240
032342 005037 001116
032346 005037 001206
032352 005237 001230
032356 042737 100000 001230
032364 005327
032366 000001
032370 003063
032372 012737
032374 000001
032376 032366
032400 104401 032406
032404 000407

032424
032424 013746 001230

032430 104405
032432 104401 032440
032436 000421

032502
032502 013746 001126

032506 104405
032510 104401 001217
032514 005037 001126
032520 013700 000042

\$EOP:
NOP
CLR \$TSTNM ;:ZERO THE TEST NUMBER
CLR \$TIMES ;:ZERO THE NUMBER OF ITERATIONS
INC \$PASS ;:INCREMENT THE PASS NUMBER
BIC #100000,\$PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
\$EOPCT: .WORD 1
BGT \$DOAGN ;:YES
MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
\$ENDCT: .WORD 1
TYPE ,65\$;:TYPE ASCIZ STRING
BR 64\$;:GET OVER THE ASCIZ
65\$: .ASCIZ <12><15>/END PASS #/
64\$: MOV \$PASS,-(SP) ;:SAVE \$PASS FOR TYPEOUT
;:TYPE PASS NUMBER
TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
TYPE ,67\$;:TYPE ASCIZ STRING
BR 66\$;:GET OVER THE ASCIZ
67\$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66\$: MOV \$ERTTL,-(SP) ;:SAVE \$ERTTL FOR TYPEOUT
;:TOTAL NUMBER OF ERRORS
TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
TYPE ,\$CRLF ;:TYPE CARRIAGE RETURN, LINE FEED
CLR \$ERTTL ;:CLEAR ERROR TOTAL
\$GET42: MOV @#42,RO ;:GET MONITOR ADDRESS

032524	001405			BEG	\$DOAGN	::BRANCH IF NO MONITOR
032526	000005			RESET		::CLEAR THE WORLD
032530	004710			SENDAD: JSR	PC,(R0)	::GO TO MONITOR
032532	000240			NOP		::SAVE ROOM
032534	000240			NOP		::FOR
032536	000240			NOP		::ACT11
032540				\$DOAGN:		
032540	000137			JMP	@(PC)+	::RETURN
032542	056756			\$RTNAD: .WORD	SHUT	
032544	377	377	000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
				.EVEN		

21

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15 032550
16 032550 104414
17 032552 032777 020000 146374
18 032560 001402
19 032562 000137 033300
20
21 032566 104401 001217
22 032572 104401 033314
23 032576 013746 001234

    032602 104403
    032604 003
    032605 000
24 032606 005037 033304
25 032612 013737 001226 033304
26 032620 104401 033322
27 032624 013746 033304

    032630 104403
    032632 003
    032633 000
28 032634 005037 033306
29 032640 113737 001130 033306
30 032646 001406
31 032650 104401 033332
32 032654 013746 033306

    032660 104403
    032662 003
    032663 000
33 032664 104401 033341
34 032670 013746 001132

    032674 104403
    032676 006
    032677 001
35
36 032700 005737 033306
37 032704 001575
38 032706 104401 001217
39 032712 105037 033312
40 032716 105037 033313
41 032722 013700 033306
    
```

```

.SBTTL SUBROUTINES
:*****
.SBTTL ERROR TYPEOUT ROUTINE

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

ERRTYP:
    SAVREG
    BIT    #SW13,@SWR    ;INHIBIT TYPEOUTS??
    BEQ    1$            ;NO!!
    JMP    21$          ;YES!!
;TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
1$:
    TYPE  ,%CRLF
    TYPE  ,ERTY00        ;TYPE "UNT#"
    MOV   $UNIT,-(SP)   ;;SAVE $UNIT FOR TYPEOUT
                        ;;TYPE UNIT NUMBER
    TYPOS
    .BYTE 3              ;;GO TYPE--OCTAL ASCII
    .BYTE 0              ;;TYPE 3 DIGIT(S)
    CLR   TSTNMB        ;;SUPPRESS LEADING ZEROS
    MOV   $TSTN,TSTNMB ;LOAD TEST NUMBER FOR
    TYPE  ,ERTY01        ;TYPE "TST#"
    MOV   TSTNMB,-(SP)  ;;SAVE TSTNMB FOR TYPEOUT
                        ;;TYPE TEST NUMBER
    TYPOS
    .BYTE 3              ;;GO TYPE--OCTAL ASCII
    .BYTE 0              ;;TYPE 3 DIGIT(S)
    CLR   ERRNMB        ;;SUPPRESS LEADING ZEROS
    MOV   $ITEMB,ERRNMB;LOAD ERROR NUMBER FOR
    TYPE  ,ERTY02        ;TYPE "ERR#"
    MOV   ERRNMB,-(SP)  ;;SAVE ERRNMB FOR TYPEOUT
                        ;;TYPE ERROR NUMBER
    TYPOS
    .BYTE 3              ;;GO TYPE--OCTAL ASCII
    .BYTE 0              ;;TYPE 3 DIGIT(S)
    .BYTE 1              ;;SUPPRESS LEADING ZEROS
2$:
    TYPE  ,ERTY03        ;TYPE "PC="
    MOV   $ERRPC,-(SP)  ;;SAVE $ERRPC FOR TYPEOUT
                        ;;TYPE PROGRAM COUNTER
    TYPOS
    .BYTE 6              ;;GO TYPE--OCTAL ASCII
    .BYTE 1              ;;TYPE 6 DIGIT(S)
    .BYTE 1              ;;TYPE LEADING ZEROS
;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
3$:
    TST   ERRNMB        ;WAS AN ERROR CALLED?
    BEQ   21$          ;NO!!
    TYPE  ,%CRLF        ;YES-TYPE CRLF
    CLRB  BOTFLG        ;CLEAR BOT FLAG
    CLRB  CHRCNT        ;CLEAR CHARACTER COUNTER
    MOV   ERRNMB,R0    ;R0 POINTS TO FIRST OF
    
```

```

42 032726 006300 ASL RO ;FOUR ENTRIES IN ERROR
43 032730 006300 ASL RO ;TABLE
44 032732 006300 ASL RO
45 032734 062700 001570 ADD #SERRTB-8.,R0
46 032740 011001 MOV (R0),R1 ;R1 POINTS TO ERROR MESSAGE
47 ;TABLE
48 032742 001507 BEQ 13$ ;BRANCH IF NO ERROR MESSAGE
49 ;TYPE THE ERROR MESSAGE
50 032744 012102 4$: MOV (R1)+,R2 ;R2=ADDRESS OF MESSAGE STRING
51 032746 001505 BEQ 12$ ;BRANCH IF END OF MESSAGE
52 032750 010237 033116 MOV R2,11$ ;LOAD ADDRESS OF STRING
53 032754 005037 033310 CLR BOTADR ;CLEAR BOT ADDRESS
54 032760 112203 5$: MOV (R2)+,R3 ;END OF STRING??
55 032762 001454 BEQ 10$ ;YES!!
56 032764 122703 000015 CMPB #CR,R3 ;CARRIAGE RETURN??
57 032770 001003 BNE 6$ ;NO!!
58 032772 105037 033313 CLRB CHRCNT ;YES-CLEAR CHAR COUNT
59 032776 000770 BR 5$ ;GET NEXT CHARACTER
60 033000 122703 000012 6$: CMPB #LF,R3 ;LINE FEED??
61 033004 001765 BEQ 5$ ;YES-GET NEXT CHARACTER
62 033006 122703 000011 CMPB #HT,R3 ;HORIZONTAL TAB??
63 033012 001007 BNE 8$ ;NO!!
64 033014 105237 033313 7$: INCB CHRCNT ;ADJUST CHARACTER COUNT
65 033020 132737 000007 033313 BITB #7,CHRCNT
66 033026 001372 BNE 7$
67 033030 000407 BR 9$
68 033032 105237 033313 8$: INCB CHRCNT ;INCREMENT CHARACTER COUNT
69 033036 122703 000040 CMPB #' ,R3 ;SPACE??
70 033042 001002 BNE 9$ ;NO!!
71 033044 010237 033310 MOV R2,BOTADR ;SAVE ADDRESS OF SPACE
72 033050 122737 000100 033313 9$: CMPB #64.,CHRCNT ;END OF LINE??
73 033056 103340 BHIS 5$ ;NO!!
74 033060 013704 033310 MOV BOTADR,R4 ;GET ADDRESS OF LAST SPACE
75 033064 001007 BNE 90$ ;BRANCH IF SPACE DETECTED
76 033066 104401 001217 TYPE ,$CRLF ;TYPE CRLF
77 033072 105037 033313 CLRB CHRCNT ;CLEAR CHARACTER COUNT
78 033076 013702 033116 MOV 11$,R2 ;SET UP R2 FOR TESTING
79 033102 000726 BR 5$
80 033104 105044 90$: CLRB -(R4) ;REPLACE SPACE
81 033106 112737 177777 033312 MOV #1,BOTFLG ;SET BOT FLAG
82 033114 104401 10$: TYPE ;TYPE ERROR MESSAGE STRING
83 033116 000000 11$: .WORD ;STRING ADDRESS GOES HERE
84 033120 105737 033312 TSTB BOTFLG ;WAS STRING TRUNCATED??
85 033124 001707 BEQ 4$ ;NO!!
86 033126 104401 001217 TYPE ,$CRLF ;YES-TYPE CRLF
87 033132 105037 033312 CLRB BOTFLG ;CLEAR BOT FLAG
88 033136 105037 033313 CLRB CHRCNT ;CLEAR CHARACTER COUNT
89 033142 013702 033310 MOV BOTADR,R2 ;SETUP R2 FOR TESTING
90 033146 010237 033116 MOV R2,11$ ;SETUP 11$ FOR TYPING
91 033152 112742 000040 MOV #,-(R2) ;RESTORE SPACE
92 033156 105722 TSTB (R2)+ ;RESTORE R2
93 033160 000677 BR 5$ ;TYPE REST OF STRING
94 033162
95 12$: ;TYPE ERROR HEADER AND ERROR DATA
96 033162
97 033162 016001 000002 13$: MOV 2(R0),R1 ;R1 POINTS TO ERROR HEADER TABLE
98 033166 001444 BEQ 21$ ;BRANCH IF NO HEADER

```

```

99 033170 104401 001217          TYPE      , $CRLF          ; (ASSUME NO DATA)
100 033174 016002 000004        MOV      4(R0),R2        ; R2 POINTS TO DATA ADDRESS TABLE
101 033200 016003 000006        MOV      6(R0),R3        ; R3 POINTS TO FORMAT TABLE
102 033204 012137 033214        14$:    MOV      (R1)+,15$    ; PUT HEADER ADDRESS FOR TYPE
103 033210 001433                BEQ      21$              ; BRANCH IF END OF HEADERS
104                                ; (ASSUME END OF DATA)
105 033212 104401          TYPE      , $CRLF          ; (ASSUME NO DATA)
106 033214 000000        15$:    .WORD    0              ; HEADER ADDRESS GOES HERE
107 033216 104401 001217        TYPE      , $CRLF          ; (ASSUME NO DATA)
108 033222 005702        TST      R2              ; DATA WITH HEADER??
109 033224 001767        BEQ      14$              ; NO!!
110 033226 012204        MOV      (R2)+,R4        ; R4 POINTS TO DATA ADDRESS
111 033230 012305        MOV      (R3)+,R5        ; R5 POINTS TO FORMAT
112 033232 105725        16$:    TSTB     (R5)+          ; WHAT KIND OF DATA??
113 033234 100407        BMI      18$              ; BINARY
114 033236 001403        BEQ      17$              ; OCTAL
115 033240 013446        MOV      @ (R4)+, -(SP)    ; DECIMAL
116 033242 104405        TYPDS   19$              ;
117 033244 000405        BR       19$              ;
118 033246 013446        17$:    MOV      @ (R4)+, -(SP)    ;
119 033250 104402        TYPOC   19$              ;
120 033252 000402        BR       19$              ;
121 033254 013446        18$:    MOV      @ (R4)+, -(SP)    ;
122 033256 104406        TYPBN   19$              ;
123 033260 005714        19$:    TST      (R4)          ; MORE DATA??
124 033262 001403        BEQ      20$              ; NO!!
125 033264 104401 033347        TYPE     , ERTY04        ; YES-TYPE 2 SPACES
126 033270 000760        BR       16$              ; AND CONTINUE
127 033272 104401 001217        20$:    TYPE     , $CRLF          ; TYPE ONE BLANK LINE
128 033276 000742        BR       14$              ; BEFORE NEXT HEADER
129 033300 104415        21$:    RESREG   PC          ;
130 033302 000207        RTS      PC              ;
131                                ;
132 033304 000000        TSTNMB: .WORD    0          ; TEST NUMBER
133 033306 000000        ERRNMB: .WORD    0          ; ERROR NUMBER
134 033310 000000        BOTADR: .WORD    0          ; BEGINNING OF TEXT ADDRESS
135 033312 000          BOTFLG: .BYTE    0          ; BOT FLAG
136 033313 000          CHRCNT: .BYTE    0          ; CHARACTER COUNT
137                                ;
138 033314 125 116 111  ERTY00: .ASCIZ @UNIT#@
139 033322 054 040 124  ERTY01: .ASCIZ @, TEST#@
140 033332 054 040 105  ERTY02: .ASCIZ @, ERR#@
141 033341 054 040 120  ERTY03: .ASCIZ @, PC=@
142 033347 040 040 000  ERTY04: .ASCIZ @ @
143                                ;
144                                .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 TEST PREPARATION MODULE

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

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

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

TSTPRP:

:STORE TASK DESCRIPTOR AND CLEAR USER'S ERROR CALL
MOV @ (SP),500\$;STORE DESCRIPTOR
ADD #6,(SP) ;MOVE SP TO USERS ERROR CALL
CLRB @ (SP) ;CLEAR ERROR CALL
SUB #4,(SP) ;MOVE SP TO NO ERROR RETURN
JSR PC,GETSTS ;SETUP TO READ ALL REGISTERS
JSR PC,GET ;GET RMER2
BR 15\$;BR IF NO ERROR DETECTED
BR 10\$;GET OVER ERROR NUMBER
.WORD 0 ;ERROR DEFINED BY GET SUBROUTINE
10\$: ADD #4,(SP) ;XFER ERROR TO USER AND
MOVB 10\$-2,@ (SP) ;GET ERROR NUMBER.
JMP 400\$
15\$: MOV RMER21,505\$;GET RMER2 AND SAVE FOR LATER

:*****


```
58 ;SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
59 033436 005737 034272 1ST 500$ ;SELECT DEVICE??
60 033442 100014 BPL 30$ ;NO!!
61
62 033444 004737 045012 JSR PC,DEVSEL ;GO SELECT DEVICE
63 033450 000411 BR 30$ ;NO ERROR - CONTINUE
64 033452 000401 BR 20$
65 033454 000000 .WORD 0 ;ERROR NUMBER FROM DEVSEL
66 033456 062716 000004 20$: ADD #4,(SP) ;TRANSFER ERROR TO USER
67 033462 113776 033454 000000 MOVB 20%-2,@(SP)
68 033470 000137 034262 JMP 400$
69
70 ;:*****
71 ;CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
72 033474 30$:
73 033474 032737 040000 034272 BIT #BIT14,500$ ;CLEAR CONTROLLER??
74 033502 001451 BEQ 120$ ;NO!!
75
76 033504 004737 046464 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
77 033510 000411 BR 60$ ;CONTINUE - NO ERROR
78 033512 000401 BR 50$
79 033514 000000 40$: .WORD 0 ;ERROR NUMBER FROM CNTCLR
80 033516 062716 000004 50$: ADD #4,(SP) ;TRANSFER ERROR TO USER
81 033522 113776 033514 000000 MOVB 40%,@(SP)
82 033530 000137 034262 JMP 400$
83
84 ;:*****
85 ;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
86 033534 60$:
87 033534 032737 000100 034272 BIT #BIT6,500$ ;VERIFY??
88 033542 001431 BEQ 120$ ;NO!!
89
90 033544 004737 037110 JSR PC,GET ;GO GET STATUS
91 033550 000411 BR 90$ ;NO ERROR GETTING STATUS
92 033552 000401 BR 80$
93 033554 000000 70$: .WORD 0 ;ERROR FROM GETTING STATUS
94 033556 062716 000004 80$: ADD #4,(SP) ;TRANSFER ERROR TO USER
95 033562 113776 033554 000000 MOVB 70%,@(SP)
96 033570 000137 034262 JMP 400$
97
98 033574 004737 046602 90$: JSR PC,CLRSTS ;GO VERIFY STATUS CLEAR
99 033600 000412 BR 120$ ;NO ERROR IN CLEAR
100 033602 000401 BR 110$
101 033604 000000 100$: .WORD 0 ;ERROR IN STATUS CLEAR
102 033606 005726 110$: TST (SP)+ ;STRIP RETURN ADDRESS TO
103 033610 062716 000004 ADD #4,(SP) ;SUBROUTINE AND TRANSFER
104 033614 113776 033604 000000 MOVB 100%,@(SP) ;ERROR TO USER
105 033622 000137 034262 JMP 400$
106
107 ;:*****
108 ;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
109 ;NOT VALID
110 033626 120$:
111 033626 032737 010000 034272 BIT #BIT12,500$ ;PACK ACKNOWLEDGE??
112 033634 001503 BEQ 240$ ;NO!!
113
114 033636 004737 037110 JSR PC,GET
```

```

115 033642 000411                BR      150$          ;NO ERROR GETTING RMDS
116 033644 000401                BR      140$
117 033646 000000                130$: .WORD      0
118 033650 062716 000004          140$: ADD      #4,(SP)      ;TRANSFER ERROR TO USER
119 033654 113776 033646 000000  MOVB    130$,a(SP)
120 033662 000137 034262          JMP     400$
121
122 033666 032737 000100 001346 150$:  BIT      #VV,RMDSI      ;IS VOLUME VALID??
123 033674 001063                BNE     240$          ;YES!!
124
125 033676 005037 001510                CLR     MEDENB        ;CLEAR MEDIA ENABLE
126 033702 012737 000023 001410  MOV     #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
127 033710 112737 000000 001551  MOVB   #RMCS1,PUTINX    ;SETUP REGISTER INDEX TABLE
128 033716 112737 000200 001552  MOVB   #200,PUTINX+1
129 033724 004737 037360          JSR     PC,PUT        ;GO WRITE COMMAND
130 033730 000410                BR      180$          ;NO ERROR LOADING REGISTER
131 033732 000401                BR      170$
132 033734 000000                160$: .WORD      0          ;ERROR FROM PUT SUB
133 033736 062716 000004          170$: ADD      #4,(SP)      ;TRANSFER ERROR TO USER
134 033742 113776 033734 000000  MOVB   160$,a(SP)
135 033750 000544                BR      400$
136
137 033752 004737 037722          180$:  JSR     PC,TIMOUT   ;WAIT FOR COMMAND TO COMPLETE
138
139
140
141 033756 032737 000020 034272  ::*****
142 033764 001427          ;VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
143
144 033766 004737 037110                BIT     #BIT4,500$    ;VERIFY PACK ACKNOWLEDGE??
145 033772 000410                BEQ     240$          ;NO!!
146 033774 000401
147 033776 000000                JSR     PC,GET        ;GO GET STATUS
148 034000 062716 000004          BR      210$          ;NO ERROR GETTING STATUS
149 034004 113776 033776 000000  BR      200$
150 034012 000523          190$: .WORD      0          ;ERROR FROM GET SUB
151
152 034014 004737 047462          200$: ADD      #4,(SP)      ;TRANSFER ERROR TO USER
153 034020 000411                MOVB   190$,a(SP)
154 034022 000401                BR      400$
155 034024 000000
156 034026 005726          210$:  JSR     PC,ACKSTS   ;GO CHECK ACKNOWLEDGE
157 034030 062716 000004          BR      240$          ;NO ERROR
158 034034 113776 034024 000000  BR      230$
159 034042 000507          220$: .WORD      0          ;PACK ACKNOWLEDGE ERROR
160
161
162
163
164 034044                230$: TST     (SP)+      ;STRIP RETURN TO SUB AND
165 034044 032737 004000 034272  ADD     #4,(SP)      ;TRANSFER ERROR TO USER
166 034052 001505                MOVB   220$,a(SP)
167
168 034054 004737 037110                BR      400$
169 034060 000410
170 034062 000401
171 034064 000000          240$:  JSR     PC,GET        ;GO GET RMDS
                BR      270$          ;NO ERROR GETTING RMDS
                BR      260$
                250$: .WORD      0          ;ERROR FROM GET SUB

```

```

172 034066 062716 000004      260$:  ADD      #4,(SP)      ;TRANSFER ERROR TO USER
173 034072 113776 034064 000000      MOVB     250$,a(SP)
174 034100 000470      BR       400$
175
176 034102 032737 040000 034274 270$:  BIT      #SKI,505$     ;WAS SKI SET ?
177 034110 001004      BNE     280$          ;YES, GO RECALIBRATE
178 034112 032737 020000 001346      BIT      #PIP,RMDSI   ;IS PIP ACTIVE??
179 034120 001462      BEQ     410$          ;NO!!
180
181 034122 012737 000007 001410 280$:  MOV      #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
182 034130 112737 000000 001551      MOVB     #RMCS1,PUTINX ;AND REGISTER INDEX
183 034136 112737 000200 001552      MOVB     #200,PUTINX+1
184 034144 004737 037360      JSR      PC,PUT       ;GO ISSUE RECALIBRATE
185 034150 000410      BR       300$        ;NO ERROR
186 034152 000401      BR       290$
187 034154 000000      .WORD   0            ;ERROR IN REGISTER TRANSFER
188 034156 062716 000004      290$:  ADD      #4,(SP)      ;TRANSFER ERROR TO USER
189 034162 113776 034154 000000      MOVB     290$-2,a(SP)
190 034170 000434      BR       400$
191
192 034172 004737 037722      300$:  JSR      PC,TIMOUT   ;WAIT FOR COMPLETION
193
194      ;:*****
195      ;VERIFY RECALIBRATE IF BIT 3 SET IN TASK
196 034176 032737 000010 034272      BIT      #BIT3,500$   ;VERIFY RECALIBRATE??
197 034204 001430      BEQ     410$          ;NO!!
198
199 034206 004737 037110      JSR      PC,GET       ;GO GET STATUS
200 034212 000410      BR       330$        ;NO ERROR GETTING STATUS
201 034214 000401      BR       320$
202 034216 000000      310$:  .WORD   0            ;ERROR FROM GET
203 034220 062716 000004      320$:  ADD      #4,(SP)      ;TRANSFER ERROR TO USER
204 034224 113776 034216 000000      MOVB     310$,a(SP)
205 034232 000413      BR       400$
206
207 034234 004737 050256      330$:  JSR      PC,RCLSTS   ;GO CHECK RECALIBRATE
208 034240 000412      BR       410$        ;NO ERROR DURING RECALIBRATE
209 034242 000401      BR       350$
210 034244 000000      340$:  .WORD   0            ;ERROR DURING RECALIBRATE
211 034246 005726      350$:  TST      (SP)+      ;STRIP RETURN TO SUB AND
212 034250 062716 000004      ADD      #4,(SP)      ;TRANSFER ERROR TO USER
213 034254 113776 034244 000000      MOVB     340$,a(SP)
214 034262 162716 000002      400$:  SUB      #2,(SP)     ;MOVE SP BACK BEFORE ERROR
215 034266 000240      410$:  NOP
216 034270 000207      RTS      PC          ;RETURN TO USER
217
218 034272 000000      500$:  .WORD   0            ;TASK/VERIFY DESCRIPTOR
219 034274 000000      505$:  .WORD   0            ;CONTAINS RMER2
220

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 034276
34 034276 062716 000006
35 034302 105076 000000
36 034306 162716 000006
37
38
39
40 034312 005737 001510
41 034316 001402
42 034320 000137 035572
43
44
45
46
47 034324
   034324 010046
48 034326 005000
49 034330 016060 001410 101206 10$:
50 034336 062700 000002
51 034342 022700 000046
52 034346 103370
53
54
55
56 034350 012737 000003 036210

```

```

.SBTTL BAD SECTOR MODULE

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

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

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

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

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

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

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

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

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

```

```

57 034356 012737 001466 001444      MOV      #822.,RMDCO      ;DESIRED CYLINDER = 822.
58 034364 013737 001332 001416      MOV      LSTRK,RMDAO     ;STARTING LAST TRACK, SECTOR = 0
59 034372 012737 177376 001412      MOV      #-258.,RMWCO    ;2 + 256. WORDS (2'S COMP)
60 034400 012737 010000 001442      MOV      #FMT16,RMOFO   ;16 BIT FORMAT
61 034406 012737 103216 001414      MOV      #MFGFIL,RMBAO  ;POINT TO MANUFACTURES FILE BUFFER
62
63 034414 012700 001551      MOV      #PUTINX,R0     ;R0 POINTS TO REGISTER INDEX TABLE
64 034420 112720 000006      MOVB    #RMDA,(R0)+
65 034424 112720 000034      MOVB    #RMDC,(R0)+
66 034430 112720 000002      MOVB    #RMWC,(R0)+
67 034434 112720 000032      MOVB    #RMOF,(R0)+
68 034440 112720 000004      MOVB    #RMBA,(R0)+
69 034444 112720 000000      MOVB    #RMCS1,(R0)+
70 034450 112720 000200      MOVB    #200,(R0)+
71 034454 012600      MOV      (SP)+,R0      ;:POP STACK INTO R0
72
73      ;SET GET INDEX TABLE FOR READING STATUS
74 034456      20$:
75 034456 004737 037024      JSR      PC,GETSTS      ;SETUP GET INDEX REGISTER FOR STATUS
76 034462 004737 037110      JSR      PC,GET        ;GET REGISTERS
77 034466 000411      BR       30$           ;BR IF NO ERROR
78 034470 000401      BR       25$           ;JUMP OVER ERROR NUMBER
79 034472 000000      .WORD   0             ;ERROR DEFINED BY GET SUB
80 034474 062716 000006      25$:  ADD     #6,(SP)      ;XFER ERROR TO USER AND
81 034500 113776 034472 000000      MOV      25$-2,a(SP)   ;GET ERROR NUMBER.
82 034506 000137 035272      JMP      215$
83
84 034512 013737 001376 036222 30$:  MOV      RMER2I,550$    ;GET RMER2 AND SAVE FOR LATER
85
86      ;CLEAR THE DEVICE USING DRIVE CLEAR COMMAND
87 034520 012737 000011 001410      MOV      #DRVCLR!GO,RMCS10 ;LOAD COMMAND IN PUT BUFFER
88 034526 004737 037360      JSR      PC,PUT        ;OUTPUT COMMAND
89 034532 000411      BR       45$           ;RETURN HERE IF NO ERROR
90 034534 000401      BR       40$           ;GET AROUND ERROR #
91 034536 000000      35$:  .WORD   0             ;ERROR # GOES HERE
92 034540 062716 000006      40$:  ADD     #6,(SP)      ;MOVE SP TO USERS ERROR CALL
93 034544 113776 034536 000000      MOV      35$,a(SP)    ;MOVE ERROR NUMBER TO USER
94 034552 000137 035272      JMP      215$
95
96 034556 004737 037722 45$:  JSR      PC,TIMOUT     ;WAIT FOR COMPLETION
97 034562 004737 037110      JSR      PC,GET        ;GO GET STATUS
98 034566 000411      BR       60$           ;RETURN HERE IF NO ERROR
99 034570 000401      BR       55$           ;GET AROUND ERROR #
100 034572 000000      50$:  .WORD   0             ;ERROR # GOES HERE
101 034574 062716 000006      55$:  ADD     #6,(SP)      ;MOVE SP TO USERS ERROR CALL
102 034600 113776 034572 000000      MOV      50$,a(SP)    ;MOVE ERROR # TO USERS ERROR CALL
103 034606 000137 035272      JMP      215$
104
105 034612 004737 052020 60$:  JSR      PC,DRVSTS     ;GO VERIFY DRIVE CLEAR COMMAND
106 034616 000412      BR       75$           ;RETURN HERE IF NO ERROR
107 034620 000401      BR       70$           ;GET AROUND ERROR
108 034622 000000      65$:  .WORD   0             ;ERROR # GOES HERE
109 034624 005726      70$:  TST     (SP)+         ;STRIP RETURN TO SUBROUTINE
110 034626 062716 000006      ADD     #6,(SP)      ;MOVE SP TO USERS ERROR CALL
111 034632 113776 034622 000000      MOV      65$,a(SP)    ;MOVE ERROR # TO USER CALL
112 034640 000137 035272      JMP      215$
113

```

```
114                                     ;ISSUE A PACK ACKNOWLEDGE IF VOLUME VALID IS RESET
115 034644                               75$:
116 034644 032737 000100 001346        BIT    #VV,RMDSI        ;IS VV RESET ??
117 034652 001052                               BNE    120$           ;NO !!
118
119 034654 012737 000023 001410        MOV    #PACACK!GO,RMCS10    ;LOAD COMMAND
120 034662 004737 037360                JSR    PC,PUT            ;GO PUT COMMAND TO DRIVE
121 034666 000411                               BR     90$             ;RETURN HERE IF NO OUTPUT ERROR
122 034670 000401                               BR     85$             ;GET AROUND ERROR #
123 034672 000000                               .WORD  0                ;ERROR # GOES HERE
124 034674 062716 000006                80$:  ADD    #6,(SP)        ;MOVE SP TO USERS ERROR CALL
125 034700 113776 034672 000000        85$:  MOVVB  80$,@ (SP)       ;MOVE ERROR # TO ERROR CALL
126 034706 000137 035272                JMP    215$
127
128 034712 004737 037722                90$:  JSR    PC,TIMOUT      ;WAIT FOR COMPLETION
129 034716 004737 037110                JSR    PC,GET           ;GO GET STATUS FOR PACK ACK
130 034722 000411                               BR     105$            ;RETURN HERE IF NO ERROR
131 034724 000401                               BR     100$            ;GET AROUND ERROR #
132 034726 000000                               .WORD  0                ;ERROR # GOES HERE
133 034730 062716 000006                95$:  ADD    #6,(SP)        ;MOVE SP TO USERS ERROR CALL
134 034734 113776 034726 000000        100$: MOVVB  95$,@ (SP)     ;MOVE ERROR # TO CALL
135 034742 000137 035272                JMP    215$
136
137 034746 004737 047462                105$: JSR    PC,ACKSTS     ;GO VERIFY ACKNOWLEDGE STATUS
138 034752 000412                               BR     120$            ;RETURN HERE IF NO ERROR
139 034754 000401                               BR     115$            ;GET AROUND ERROR #
140 034756 000000                               .WORD  0                ;ERROR # GOES HERE
141 034760 005726                               110$: TST    (SP)+        ;STRIP RETURN TO SUBROUTINE
142 034762 062716 000006                115$: ADD    #6,(SP)        ;MOVE SP TO USERS ERROR CALL
143 034766 113776 034756 000000        MOVVB  110$,@ (SP)     ;MOVE ERROR # TO USERS ERROR CALL
144 034774 000137 035272                JMP    215$
145
146                                     ;RECALIBRATE THE DRIVE IF "SKI" OR "PIP IS SET
147 035000                               120$:
148 035000 032737 040000 036222        BIT    #SKI,550$        ;WAS SKI SET ?
149 035006 001004                               BNE    125$            ;YES, GO RECALIBRATE
150 035010 032737 020000 001346        BIT    #PIP,RMDSI       ;IS PIP SET ??
151 035016 001452                               BEQ    165$            ;NO !!
152
153 035020 012737 000007 001410        125$: MOV    #RECAL!GO,RMCS10    ;LOAD RECALIBRATE COMMAND
154 035026 004737 037360                JSR    PC,PUT            ;PUT THE RECAL COMMAND
155 035032 000411                               BR     135$            ;RETURN HERE IF NO ERROR
156 035034 000401                               BR     130$            ;GET AROUND ERROR #
157 035036 000000                               .WORD  0                ;ERROR # GOES HERE
158 035040 062716 000006                130$: ADD    #6,(SP)        ;MOVE SP TO USERS ERROR CALL
159 035044 113776 035036 000000        MOVVB  130$-2,@ (SP)   ;MOVE ERROR # TO USERS CALL
160 035052 000137 035272                JMP    215$
161
162 035056 004737 037722                135$: JSR    PC,TIMOUT      ;WAIT FOR RECALIBRATE TO COMPLETE
163 035062 004737 037110                JSR    PC,GET           ;GO GET RECAL STATUS
164 035066 000411                               BR     150$            ;RETURN HERE IF NO ERROR
165 035070 000401                               BR     145$            ;GET AROUND ERROR #
166 035072 000000                               .WORD  0                ;ERROR # GOES HERE
167 035074 062716 000006                140$: ADD    #6,(SP)        ;MOVE SP TO USERS ERROR CALL
168 035100 113776 035072 000000        145$: MOVVB  140$,@ (SP)   ;MOVE ERROR TO USERS CALL
169 035106 000137 035272                JMP    215$
170
```

```
171 035112 004737 050256      150$: JSR PC,RCLSTS      ;GO VERIFY RECALIBRATE STATUS
172 035116 000412              BR 165$                ;RETURN HERE IF NO ERROR
173 035120 000401              BR 160$                ;GET AROUND ERROR #
174 035122 000000      155$: .WORD 0                ;ERROR # GOES HERE
175 035124 005726      160$: TST (SP)+            ;STRIP RETURN TO SUBROUTINE
176 035126 062716 000006      ADD #6,(SP)            ;MOVE SP TO USERS ERROR CALL
177 035132 113776 035122 000000  MOVB 155$,@ (SP)      ;MOVE ERROR # TO USERS CALL
178 035140 000137 035272      JMP 215$
179
180      ;READ THE SECTOR IDENTIFIED BY RMDAO, INCLUDING HEADER AND DATA
181 035144      165$:
182 035144 012737 000073 001410  MOV #RH!GO,RMCS10    ;LOAD READ HEADER AND DATA COMMAND
183 035152 004737 037360      JSR PC,PUT            ;PUT COMMAND
184 035156 000411              BR 180$                ;RETURN HERE IF NO ERROR
185 035160 000401              BR 175$                ;GET AROUND ERROR #
186 035162 000000      170$: .WORD 0                ;ERROR # GOES HERE
187 035164 062716 000006      175$: ADD #6,(SP)            ;MOVE SP TO USERS ERROR CALL
188 035170 113776 035162 000000  MOVB 170$,@ (SP)    ;MOVE ERROR # TO USERS ERROR CALL
189 035176 000137 035272      JMP 215$
190
191 035202 004737 037722      180$: JSR PC,TIMOUT        ;WAIT FOR READ OPERATION TO COMPLETE
192 035206 004737 037110      JSR PC,GET            ;GO GET STATUS FOR READ OPERATION
193 035212 000411              BR 195$                ;RETURN HERE IF NO ERROR
194 035214 000401              BR 190$                ;GET AROUND ERROR #
195 035216 000000      185$: .WORD 0                ;ERROR # GOES HERE
196 035220 062716 000006      190$: ADD #6,(SP)            ;MOVE SP TO USERS ERROR CALL
197 035224 113776 035216 000000  MOVB 185$,@ (SP)    ;MOVE ERROR # TO CALL
198 035232 000137 035272      JMP 215$
199
200 035236 004737 052622      195$: JSR PC,DASTS        ;GO VERIFY RESULTS OF READ OPERATION
201 035242 000412              BR 210$                ;RETURN HERE IF NO ERROR
202 035244 000401              BR 205$                ;GET AROUND ERROR #
203 035246 000000      200$: .WORD 0                ;ERROR # GOES HERE
204 035250 005726      205$: TST (SP)+            ;STRIP RETURN ADDRESS TO SUBROUTINE
205 035252 062716 000006      ADD #6,(SP)            ;MOVE SP TO USERS ERROR CALL
206 035256 113776 035246 000000  MOVB 200$,@ (SP)    ;MOVE ERROR # TO USERS CALL
207 035264 000137 035272      JMP 215$
208
209 035270 000450      210$: BR 240$                ;NO ERRORS DETECTED
210
211      ;*****
212      ;AN ERROR HAS BEEN DETECTED IN TRYING TO READ THE BAD SECTOR FILE.
213      ;THE SECTOR WILL BE RETRIED IF POSSIBLE.
214 035272      215$:
215 035272 005337 036210      DEC 500$                ;YES, DECREMENT RETRY COUNT AND
216 035276 100030      BPL 225$                ;RETRY IF COUNT NOT NEGATIVE.
217
218      ;THE RETRY COUNT HAS EXPIRED - SEE IF THE ERROR IS MEDIA RELATED
219 035300      220$:
220 035300 013746 001376      MOV RMER2I,-(SP)      ;GET ER2
221 035304 042726 100000      BIC #BSE,(SP)+        ;ANY NON-MEDIA ERRORS ?
222 035310 001027      BNE 230$                ;YES, EXIT AND REPORT ERROR ON RETURN
223
224 035312 013746 001350      MOV RMER1I,-(SP)      ;GET ER1
225 035316 042726 100720      BIC #DCK!HCRC!HCE!FER!ECH,(SP)+ ;ARE THERE ANY NON-MEDIA ERRORS ?
226
227 035322 001022      BNE 230$                ;YES, EXIT AND REPORT ERROR ON RETURN
```

228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281

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

ADD #2,RMDAO ;ADVANCE SECTOR ADDRESS BY 2
CMPB #10.,RMDAO ;QUIT IF ALL MFG SECTORS HAVE BEEN
BEQ 230\$;TRIED.
CMPB #32.,RMDAO ;QUIT IF ALL USER SECTORS HAVE BEEN
BEQ 230\$;TRIED.
225\$: MOV #3,500\$;REINSTATE RETRY COUNT FOR THIS SECTOR
SUB #6,(SP) ;MOVE SP BACK TO NO ERROR
JMP 20\$;RETRY THE READ OPERATION

:THE BAD SECTOR FILE CANNOT BE READ

230\$: NOP
BIT #SW13,@SWR ;INHIBIT MESSAGE ?
BNE 235\$;YES
SUB #4,(SP) ;MOVE SP TO ERROR RETURN
235\$: JMP 410\$;GO TO MODULE EXIT

:THE SECTOR WAS RECOVERED WITHOUT ERROR - READ THE USER FILE IF
:THIS IS THE MGF FILE OR ELSE DONE.

240\$: CMP #USRFIL,RMBAO ;WAS THE USER FILE READ ??
BEQ 260\$;YES - READ IS COMPLETE
MOVB #10.,RMDAO ;READ THE USER FILE LAST TRACK, SECTOR = 10.
MOV #USRFIL,RMBAO ;POINT TO USERS FILE BUFFER
259: MOV #3,500\$;RELOAD THE RETRY COUNT FOR THIS SECTOR
260: JMP 20\$;GO READ THE USER FILE

:DUMMY THE BAD SECTOR FILES

250\$: MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV #252.,R1 ;R1 = NUMBER OF ENTRIES IN FILES
MOV #14,R0 ;R0 = ADDRESS INDEX TO FILE STORAGE
255\$: MOV #-1,MFGFIL(R0) ;ENTER ALL ONES IN MFG FILE
MOV #-1,USRFIL(R0) ;ENTER ALL ONES IN USER FILE
TST (R0)+ ;ADVANCE ADDRESS
DEC R1 ;DECREMENT COUNT
BNE 255\$;CONTINUE IF NOT DONE
272: MOV #6.,R1 ;CLEAR HEADER, CLEAR ID & SERIAL NUMBERS
273: CLR R0
257\$: CLR MFGFIL(R0)
CLR USRFIL(R0)
TST (R0)+ ;ADVANCE ADDRESS
DEC R1
BNE 257\$
279: MOV (R0)+,R1 ;:POP STACK INTO R1
MOV (SP)+,R0 ;:POP STACK INTO R0

:SET MEDIA ENABLE AND RESTORE THE USERS PUT BUFFER

282 035536
283 035540 010046
284 035542 012737 177777 001510
285 035550 016060 101206 001410
286 035556 062700 000002
287 035562 022700 000046
288 035566 103370
289 035570 012600
290
291
292
293
294
295
296
297
298 035572

260\$: MOV R0,-(SP) ;:PUSH R0 ON STACK
CLR R0 ;:R0 IS REGISTER INDEX
265\$: MOV #-1,MEDENB
MOV BUFFER(R0),PUTBUF(R0)
ADD #2,R0 ;:ADVANCE R0
CMP #46,R0 ;:DONE ??
BHS 265\$
MOV (SP)+,R0 ;:POP STACK INTO R0

:VERIFY THAT THE DESIRED SECTOR IS NOT IN THE MFG BAD SECTOR FILE
:AND NOT IN THE USERS BAD SECTOR FILE. ASSIGN A NEW SECTOR IF THE
:SECTOR IS IN ANY OF THE FILES.

:LOAD INITIAL VARIABLES AND COMPUTE THE NUMBER OF SECTORS

300\$:
300 035572 010046
301 035574 010146
302 035576 010246
299 035600 013737 001444 001512
300 035606 013737 001416 001514
301 035614 005002
302 035616 013700 001412
303 035622 005400
304 035624 012701 000400
305 035630 032737 000002 001410
306 035636 001402
307 035640 012701 000402
308 035644 020100 305\$:
309 035646 101404
310 035650 005700
311 035652 001405
312 035654 005202
313 035656 000403
314 035660 160100 310\$:
315 035662 005202
316 035664 000767
317 035666 010237 036210 315\$:

300\$:
MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R2,-(SP) ;:PUSH R2 ON STACK
MOV RMDCO,ASND C ;:LOAD REQUESTED CYLINDER, TRACK,
RMDAO,ASND A ;:AND SECTOR ADDRESS IN ASSIGNED STORAGE
CLR R2 ;:R2 = NUMBER OF SECTORS
MOV RMWCO,R0 ;:R0 = WORD COUNT
NEG R0 ;:MAKE NUMBER POSITIVE
MOV #256.,R1 ;:R1 = NUMBER OF WORDS PER SECTOR
BIT #BIT1,RMCS10 ;:IS THIS A HEADER AND DATA COMMAND ??
BEQ 305\$;:NO !!
MOV #258.,R1 ;:CHANGE WORDS PER SECTOR
305\$:
CMP R1,R0 ;:IS THERE A FULL SECTOR ??
BLOS 310\$;:YES !!
TST R0 ;:IS R0 ZERO ??
BEQ 315\$;:YES !!
INC R2 ;:INCREMENT FOR PARTIAL SECTOR
310\$:
BR 315\$
SUB R1,R0 ;:SUBTRACT ONE SECTOR FROM WORD COUNT
INC R2 ;:INCREMENT SECTOR COUNT
315\$:
BR 305\$
MOV R2,500\$;:SAVE SECTOR COUNT

:LOAD PARAMETERS AND SEARCH THE MFG/USER SECTOR FILE FOR THE
:ASSIGNED SECTOR. ALSO, SEARCH THE ADJACENT SECTORS IF THE
:SECTOR COUNT IS MORE THAN ONE.

323 035672 012737 103232 036220
324
325 035700 004737 035722
326 035704 012737 104240 036220
327
328 035712 004737 035722
329 035716 000137 036162
330
331 035722 013737 001512 036214 320\$:
332 035730 013737 001514 036216
333 035736 013737 036210 036212
334

MOV #MFGFIL+14,540\$;:THE STARTING ADDRESS OF MFG FILE
;:TO BASE ADDRESS STORAGE.
JSR PC,320\$;:GO SEARCH FILE
MOV #USRFIL+14,540\$;:LOAD STARTING ADDRESS OF USR FILE
;:TO BASE ADDRESS STORAGE.
JSR PC,320\$;:GO SEARCH FILE
JMP 400\$;:DONE WITH ALL FILE SEARCHES !!
320\$:
MOV ASNDC,520\$;:LOAD COMPARING CYLINDER ADDRESS
MOV ASNDA,530\$;:LOAD COMPARING TRACK, SECTOR ADDRESS
MOV 500\$,510\$;:LOAD NUMBER OF SECTORS TO CONFIRM

```

335 ;SETUP FOR A BINARY SEARCH OF THE CURRENT FILE FOR THE COMPARING
336 ;CYLINDER, TRACK AND SECTOR ADDRESS
337 035744 325$:
338 035744 013700 036220      MOV     540$,R0      ;LOAD THE BASE ADDRESS IN R0
339 035750 022710 177777      330$: CMP     #-1,(R0)    ;IS THIS FILE TERMINATOR ?
340 035754 001446              BEQ     370$         ;BR IF YES
341 035756 021037 036214      CMP     (R0),520$    ;DOES TABLE ENTRY = COMPARING CYLINDER ?
342 035762 001010              BNE     350$         ;BR IF NO
343
344 ;FILE ENTRY EQUALS COMPARING CYLINDER. SEE IF THE NEXT ENTRY EQUALS
345 ;THE COMPARING TRACK, AND SECTOR.
346 035764 345$:
347 035764 126037 000003 036217  CMPB    3(R0),530$+1  ;DOES TABLE ENTRY = COMPARING TRACK ?
348 035772 001004              BNE     350$         ;BR IF NO
349
350 035774 126037 000002 036216  CMPB    2(R0),530$    ;DOES TABLE ENTRY = COMPARING SECTOR ?
351 036002 001402              BEQ     360$         ;BR IF YES
352
353 036004 022020 350$: CMP     (R0)+,(R0)+    ;NO, ADJUST CYLINDER POINTER IN BAD FILE
354 036006 000760              BR      330$         ;AND CONTINUE SEARCH.
355
356 ;THE COMPARING CYLINDER, TRACK AND SECTOR IS IN THE BAD SECTOR FILE.
357 ;ADVANCE THE ASSIGNED SECTOR AND START THE SEARCH ALL OVER.
358 036010 360$:
359 036010 105237 001514          INCB    ASNDA         ;INCREMENT SECTOR
360 036014 122737 000037 001514  CMPB    #31.,ASNDA    ;SECTOR OK ??
361 036022 103337              BHIS   320$          ;YES !!
362 036024 105037 001514          CLRB   ASNDA         ;CLEAR SECTOR AND ADVANCE TRACK
363 036030 105237 001515          INCB   ASNDA+1
364 036034 123737 001333 001515  CMPB    LSTRK+1,ASNDA+1 ;TRACK OK ?
365 036042 103327              BHIS   320$          ;YES !!
366 036044 005037 001514          CLR    ASNDA         ;CLEAR TRACK AND SECTOR
367 036050 005237 001512          INC    ASNDC         ;INCREMENT CYLINDER
368 036054 022737 001466 001512  CMP     #822.,ASNDC   ;CYLINDER OK ??
369 036062 103317              BHIS   320$          ;YES !!
370 036064 005037 001512          CLR    ASNDC         ;START AT CYLINDER 0
371 036070 000714              BR     320$         ;SEARCH NEXT SECTOR
372
373 ;THE COMPARING SECTOR IS NOT IN THE BAD SECTOR FILES. DECREMENT THE
374 ;NUMBER OF SECTORS TO COMPARE AND SEARCH THE NEXT SECTOR IF THE NUMBER
375 ;IS NOT ZERO.
376 036072 370$:
377 036072 005337 036212          DEC     510$         ;DECREMENT NUMBER OF SECTORS TO COMPARE
378 036076 001442              BEQ     410$         ;DONE IF ZERO
379
380 036100 105237 036216          INCB    530$         ;INCREMENT THE COMPARING SECTOR
381 036104 122737 000037 036216  CMPB    #31.,530$    ;SECTOR OK ??
382 036112 103022              BHIS   375$         ;YES !!
383 036114 105037 036216          CLRB   530$         ;CLEAR SECTOR
384 036120 105237 036217          INCB   530$+1       ;INCREMENT TRACK
385 036124 123737 001333 036217  CMPB    LSTRK+1,530$+1 ;TRACK OK ??
386 036132 103012              BHIS   375$         ;YES !!
387 036134 005037 036216          CLR    530$         ;CLEAR SECTOR TRACK
388 036140 005237 036214          INC    520$         ;INCREMENT CYLINDER
389 036144 022737 001466 036214  CMP     #822.,520$    ;CYLINDER OK ??
390 036152 103002              BHIS   375$         ;YES !!
391 036154 005037 036214          CLR    520$         ;START AT CYLINDER 0

```

```
392
393 036160 000671          375$: BR      325$          ;SEARCH NEXT SECTOR
394
395          ;ASSIGN THE SECTOR AND RETURN TO USER
396 036162          400$:
397 036162 013737 001512 001444      MOV      ASNDC,RMDCO      ;LOAD CYLINDER
398 036170 013737 001514 001416      MOV      ASNDA,RMDAO      ;LOAD TRACK AND SECTOR
399 036176 012602          MOV      (SP)+,R2        ;;POP STACK INTO R2
      036200 012601          MOV      (SP)+,R1        ;;POP STACK INTO R1
      036202 012600          MOV      (SP)+,R0        ;;POP STACK INTO R0
400
401 036204 000240          410$: NOP
402 036206 000207          RTS      PC
403
404          ;THE FOLLOWING ARE STORAGE LOCATIONS FOR THE MODULE
405
406 036210 000000          500$: .WORD 0          ;RETRY COUNT/ NUMBER OF SECTORS REQUIRED
407 036212 000000          510$: .WORD 0          ;NUMBER OF SECTORS TO COMPARE
408 036214 000000          520$: .WORD 0          ;COMPARING CYLINDER
409 036216 000000          530$: .WORD 0          ;COMPARING TRACK AND SECTOR
410 036220 000000          540$: .WORD 0          ;BASE ADDRESS OF BAD SECTOR FILE BEING SEARCHED
411 036222 000000          550$: .WORD 0          ;CONTAINS RMER2
412
```

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

.SBTTL BUFFER GENERATOR SUBROUTINE

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

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

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

GENBUF:

```

MOV R0,-(SP)      ;; PUSH R0 ON STACK
MOV R1,-(SP)      ;; PUSH R1 ON STACK
MOV R2,-(SP)      ;; PUSH R2 ON STACK
MOV R3,-(SP)      ;; PUSH R3 ON STACK
MOV R4,-(SP)      ;; PUSH R4 ON STACK
MOV RMBAO,R0      ; LOAD DATA BUFFER ADDRESS
MOV RMWCO,R1      ; LOAD WORD COUNT
MOV RMDCO,60$     ; LOAD STARTING CYLINDER ADDRESS
MOV RMDAO,65$     ; LOAD STARTING TRACK,SECTOR ADDRESS

10$: BIT #BIT1,RMCS10 ; WRITE HEADER & DATA??
     BEQ 25$        ; NO!!
     MOV 60$,(R0)   ; WRITE HEADER WORD #1
     BIS #MSE!USE,(R0) ; SET BAD SECTOR FLAGS FOR GOOD SECTOR
     MOV #29.,R2    ; R2 = MAXIMUM SECTOR ADDRESS (29.)

18 BIT #FMT16,RMOFO ; 18 BIT FORMAT??
   BEQ 15$        ; YES !!
   BIS #FMT16,(R0) ; SET 16 FORMAT BIT IN HEADER
   MOV #31.,R2    ; CHANGE MAXIMUM SECTOR ADDRESS (31.)

15$: INC R1        ; INCREMENT WORD COUNT
     BEQ 50$      ; EXIT IF DONE

TST (R0)+        ; MOVE R0 TO HEADER WORD #2
MOV 65$,(R0)+    ; WRITE HEADER WORD #2
INC R1          ; INCREMENT WORD COUNT AND
BEQ 50$        ; EXIT IF DONE
MOV #65$,R3     ; ADVANCE SECTOR ADDRESS
INCB (R3)
CMPB R2,(R3)    ; SECTOR OVERFLOW ??
BHIS 25$        ; NO !!
CLRB (R3)       ; YES - CLEAR SECTOR ADDRESS
INCB 1(R3)      ; ADVANCE TRACK ADDRESS
CMPB LSTRK+1,1(R3) ; TRACK OVERFLOW ??
BHIS 25$        ; NO !!
CLRB 1(R3)     ; YES - CLEAR TRACK ADDRESS
    
```



```

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      ;(7) ???
16
17      CMPBUF:
18      036462 MOV R0,-(SP) ;:PUSH R0 ON STACK
19      036462 MOV R1,-(SP) ;:PUSH R1 ON STACK
20      036464 MOV R2,-(SP) ;:PUSH R2 ON STACK
21      036466 MOV R3,-(SP) ;:PUSH R3 ON STACK
22      036470 CLR 150$ ;CLEAR CORRECTION FLAG
23      036472 005037 037022
24
25      ;DETERMINE IF DATA SHOULD BE CORRECTED
26      036476 BIT ECI,RMOF1 ;WAS ECC CORRECTION ALLOWED ??
27      036504 BNE 80$ ;NO !!
28      036506 BIT #DCK,RMER11 ;WAS THERE A DATA CHECK ??
29      036514 BEQ 80$ ;NO !!
30      036516 BIT #ECH,RMER11 ;IS ERROR CORRECTION HARD SET ?
31      036524 BNE 80$ ;YES !!
32      036526 BIT #FMT16,RMOF1 ;IS THIS 16 BIT FORMAT ??
33      036534 BEQ 80$ ;NO !!
34
35      ;CORRECT DATA USING ECC INFORMATION
36      036536 MOV RMBAD,R0 ;R0 = STARTING BUFFER ADDRESS
37      036542 MOV RMEC11,R1 ;R1 = ECC POSITION
38      036546 BIS #BIT15,150$ ;SET CORRECTION FLAG
39
40      ;MOVE R0 TO WORD BOUNDARY OF ERROR BURST
41      10$: CMP #16.,R1 ;IS BIT POSITION > 1 WORD
42      37 036560 BHIS 20$ ;NO !!
43      38 036562 SUB #16.,R1 ;SUBTRACT 1 WORDS WORTH
44      39 036566 TST (R0)+ ;ADVANCE BUFFER ADDRESS 1 WORD
45      40 036570 BR 10$
46      41 036572 MOV #1,R2 ;R2 = BIT POINTER
47      42 036576 MOV R2,R3 ;R3 = BIT NUMBER
48
49      ;MOVE R2 TO STARTING BIT OF ERROR BURST
50      30$: CMP R3,R1 ;IS R3 SAME AS R1 ??
51      46 036602 BEQ 35$ ;YES !!
52      47 036604 ASL R2 ;SHIFT BIT POINTER
53      48 036606 INC R3 ;INCREMENT BIT NUMBER
54      49 036610 BR 30$
55      50 036612 MOV #11.,R3 ;R3 = LENGTH OF ERROR BURST
56
57      ;CORRECT THE ERROR BURST
58      40$: BIT R2,RMEC21 ;IS THIS BIT SET IN ECC PATTERN ??
59
60      51
61      52
62      53 036616 030237 001402

```

```
54 036622 001405      BEQ      60$      ;NO - DO NOT CORRECT THIS BIT
55 036624 030210      BIT      R2,(R0)  ;IS THE BIT PRESENTLY SET ??
56 036626 001402      BEQ      50$      ;NO
57 036630 040210      BIC      R2,(R0)  ;RESET THE BIT
58 036632 000401      BR       60$
59 036634 050210      50$:     BIS      R2,(R0)  ;SET THE BIT
60 036636 006302      60$:     ASL      R2      ;SHIFT TO NEXT BIT
61 036640 001003      BNE      70$
62 036642 012702 000001  MOV      #1,R2    ;CONTINUE WITH FIRST BIT OF NEXT WORD
63 036646 005720      TST      (R0)+
64 036650 005303      70$:     DEC      R3      ;END OF BURST ??
65 036652 001361      BNE      40$      ;NO !!
66
67 ;COMPARE WRITE BUFFER TO READ BUFFER
68 036654 017600 000010  80$:     MOV      @10(SP),R0  ;R0 = WRITE BUFFER
69 036660 062766 000002 000010  ADD      #2,10(SP)  ;MOVE SP TO READ ADDRESS
70 036666 017601 000010  MOV      @10(SP),R1  ;R1 = READ BUFFER
71 036672 062766 000002 000010  ADD      #2,10(SP)  ;MOVE SP TO RETURN ADDRESS
72 036700 013702 001336  MOV      RMWCI,R2   ;R2 = NUMBER OF WORDS TRANSFER
73 036704 163702 001412  SUB      RMWCO,R2
74 036710 022021      90$:     CMP      (R0)+,(R1)+ ;COMPARE DATA WORDS
75 036712 001003      BNE      100$     ;EXIT IF NOT EQUAL
76 036714 005302      DEC      R2      ;DECREMENT WORD COUNT
77 036716 001374      BNE      90$     ;CONTINUE IF NOT DONE
78 036720 000433      BR       110$    ;DONE COMPARE - NO ERROR
79
80 ;DATA COMPARE FAILED
81 036722 014037 001140  100$:    MOV      -(R0),$GDDAT ;STORE GOOD DATA FOR TYPEOUT
82 036726 014137 001142  MOV      -(R1),$BDDAT ;STORE BAD DATA FOR TYPEOUT
83 036732 010037 001134  MOV      R0,$GDADR   ;STORE ADDRESS OF GOOD DATA
84 036736 010137 001136  MOV      R1,$BDADR   ;STORE ADDRESS OF BAD DATA
85 036742 010237 001174  MOV      R2,$TMP0    ;STORE WORD COUNT OF ERROR
86 036746 062766 000004 000010  ADD      #4,10(SP)  ;MOVE SP TO USER'S ERROR CALL
87 036754 112776 000336 000010  MOVB     #336,@10(SP);WRITE ERROR NUMBER IN CALL
88
89 ;CHANGE ERROR NUMBER IF ECC CORRECTION FAILED
90 036762 032737 100000 037022  BIT      #BIT15,150$ ;WAS ECC CORRECTION USED ??
91 036770 001403      BEQ      105$     ;NO !!
92 036772 112776 000163 000010  MOVB     #163,@10(SP);ECC CORRECTION FAILED
93 037000 162766 000002 000010  105$:    SUB      #2,10(SP)  ;MOVE SP TO RETURN IF ERROR
94 037006 000240      NOP
95 037010      110$:
96 037010 012603      MOV      (SP)+,R3   ;;POP STACK INTO R3
97 037012 012602      MOV      (SP)+,R2   ;;POP STACK INTO R2
98 037014 012601      MOV      (SP)+,R1   ;;POP STACK INTO R1
99 037016 012600      MOV      (SP)+,R0   ;;POP STACK INTO R0
037020 000207      RTS      PC      ;RETURN TO USER
150$: .WORD      ;ECC CORRECTION FLAG
```

```

1      .SBTTL  GET STATUS SUBROUTINE
2
3      ;THIS SUBROUTINE SETS UP THE "GET INDEX TABLE" AND THE "GET
4      ;BUFFER" FOR READING ALL SUBSYSTEM REGISTERS VIA THE GET SUBROUTINE
5      ;AND THEN RETURNS TO THE USER.
6      ;
7      ;CALL:  JSR      PC,GETSTS
8      ;          ???
9
10     GETSTS:
11     MOV      R0,-(SP)      ;;PUSH R0 ON STACK
12     MOV      R1,-(SP)      ;;PUSH R1 ON STACK
13     MOV      R2,-(SP)      ;;PUSH R2 ON STACK
14     MOV      #GETINX,R0    ;R0 = ADDRESS OF INDEX TABLE
15     MOV      #RMEC21+2,R1  ;R1 = ADDRESS OF GET BUFFER
16     MOV      #RMEC2,R2     ;R2 = REGISTER INDE
17     2$:      MOVB     R2,(R0)+ ;WRITE REGISTER INDEX IN TABLE
18     CLR      -(R1)         ;CLEAR CORRESPONDING LOCATION
19     3$:      SUB      #2,R2  ;DECREMENT TO NEXT INDEX
20     BMI      4$           ;BRANCH OUT IF DONE
21     CMP      #RMDB,R2     ;DONT WRITE RMDB INDEX
22     BNE      2$
23     CLR      -(R1)
24     BR      3$
25     4$:      MOVB     #200,(R0)+ ;WRITE TERMINATOR
26     MOV      (SP)+,R2     ;;POP STACK INTO R2
27     MOV      (SP)+,R1     ;;POP STACK INTO R1
28     MOV      (SP)+,R0     ;;POP STACK INTO R0
29     NOP
30     RTS      PC

```


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

```
.SBTTL GET SUBROUTINE

:THIS SUBROUTINE READS THE REGISTERS WHICH ARE LISTED IN THE
:"GET INDEX TABLE" AND STORES THEIR VALUES IN THE CORRESPONDING
:LOCATION IN THE "GET REGISTER BUFFER". FOR EXAMPLE, AN
:ENTRY OF 04 IN THE TABLE WILL CAUSE THE SUBROUTINE TO
:READ "RMB" 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 $BASE,R0
      MOV #GETBUF,R2
      MOV #GETINX,R4
      MOV #5$,ERRVEC ;SETUP FOR TIMEOUT
      MOV #PR6,ERRVEC+2
      MOV RMCS2(R0),$TMP0 ;GET "NED" STATUS
      MOV RMCS1(R0),$TMP1 ;GET "DVA" STATUS
      BIT #DVA,$TMP1 ;DEVICE AVAILABLE??
      BNE 3$ ;YES!!
      ADD #4,16(SP) ;WRITE ERROR NUMBER IN USER'S
      MOVB #112,@16(SP) ;ERROR CALL
      BR 7$
      TSTB (R4) ;DONE??
      BMI 9$ ;YES!!
      MOVB (R4),R1 ;R1 = REGISTER ADDRESS
      BIC #^CIDXMSK,R1 ;CLEAR ANY SIGN EXTENSION
```

```
037110 000240
037112 062716 000004
037116 105076 000000
037122 162716 000004
037126 010046
037130 010146
037132 010246
037134 010346
037136 010446
037140 013746 000004
037144 013746 000006
037150 013700 001276
037154 012702 001334
037160 012704 001522
037164 012737 037272 000004
037172 012737 000300 000006
037200 016037 000010 001174
037206 016037 000000 001176
037214 032737 004000 001176
037222 001007
037224 062766 000004 000016
037232 112776 000112 000016
037240 000423
037242 105714
037244 100433
037246 111401
037250 042701 177700
```

```

52 037254 060001          ADD      R0,R1
53 037256 112403          MOVVB   (R4)+,R3      ;R3 = STORAGE ADDRESS FOR REGISTER
54 037260 042703 177700   BIC     #^CIDXMSK,R3 ;CLEAR ANY SIGN EXTENSION
55 037264 060203          ADD      R2,R3
56 037266 011113          MOV     (R1),(R3)   ;READ REGISTER
57 037270 000764          BR      3$
58
59 037272 022626          5$:     CMP     (SP)+,(SP)+ ;RESTORE STACK
60 037274 062766 000004 000016   ADD     #4,16(SP)   ;WRITE ERROR NUMBER IN
61 037302 112776 000007 000016   MOVVB  #7,@16(SP)  ;USER'S ERROR CALL
62 037310 162766 000002 000016   7$:     SUB     #2,16(SP)
63 037316 105714          8$:     TSTB   (R4)      ;DONE CLEARING??
64 037320 100405          BMI     9$         ;YES!!
65 037322 005003          CLR     R3         ;CLEAR REMAINING STORAGE
66 037324 112403          MOVVB  (R4)+,R3   ;LOCATIONS
67 037326 060203          ADD     R2,R3
68 037330 005013          CLR     (R3)
69 037332 000771          BR      8$
70 037334          9$:
   037334 012637 000006   MOV     (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
   037340 012637 000004   MOV     (SP)+,ERRVEC  ;;POP STACK INTO ERRVEC
   037344 012604          MOV     (SP)+,R4     ;;POP STACK INTO R4
   037346 012603          MOV     (SP)+,R3     ;;POP STACK INTO R3
   037350 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
   037352 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
   037354 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
71 037356 000207          RTS     PC          ;RETURN
72

```

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

.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

```

28 037360 000240
29 037362 010046
   037364 010146
   037366 010246
   037370 010346
   037372 010446
   037374 013746 000004
   037400 013746 000006
30 037404 013700 001276
31 037410 012702 001410
32 037414 012704 001551
33 037420 012737 037530 000004
34 037426 012737 000300 000006
35 037434 016037 000010 001174 1$:
36 037442 016037 000000 001176
37 037450 032737 004000 001176
38 037456 001007
39 037460 062766 000004 000016
40 037466 112776 000112 000016
41 037474 000424
42 037476 105714
43 037500 100425
44 037502 111401
45 037504 042701 177700
46 037510 060001
47 037512 111403
48 037514 042703 177700
49 037520 060203
50 037522 011311
51 037524 105724

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

```
52 037526 000763                    BR            3$
53
54 037530 022626                    5$:    CMP        (SP)+,(SP)+        ;ADJUST STACK
55 037532 062766 000004 000016        ADD        #4,16(SP)        ;WRITE ERROR NUMBER IN
56 037540 112776 000007 000016        MOVB      #7,@16(SP)       ;USER'S ERROR CALL
57 037546 162766 000002 000016        7$:    SUB        #2,16(SP)
58
59 037554                    9$:
   037554 012637 000006                MOV        (SP)+,ERRVEC+2        ;:POP STACK INTO ERRVEC+2
   037560 012637 000004                MOV        (SP)+,ERRVEC       ;:POP STACK INTO ERRVEC
   037564 012604                    MOV        (SP)+,R4        ;:POP STACK INTO R4
   037566 012603                    MOV        (SP)+,R3        ;:POP STACK INTO R3
   037570 012602                    MOV        (SP)+,R2        ;:POP STACK INTO R2
   037572 012601                    MOV        (SP)+,R1        ;:POP STACK INTO R1
   037574 012600                    MOV        (SP)+,R0        ;:POP STACK INTO R0
60 037576 000207                    RTS        PC            ;RETURN
61
```

```

1          .SBTTL  SIZE CLOCK SUBROUTINE
2
3 037600   .SIZCLK:
4 037600   013746 000004   MOV     ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
5 037604   013746 000006   MOV     ERRVEC+2,-(SP)   ;;PUSH ERRVEC+2 ON STACK
6 037610   012737 037646 000004   MOV     #1$,ERRVEC      ;;SET UP FOR BUS TIMEOUT
7 037616   012737 000300 000006   MOV     #PR6,ERRVEC+2
8 037624   012737 177546 001516   MOV     #177546,CLKADR  ;;LOAD ADDRESSES FOR KW11-L
9 037632   012737 000100 001520   MOV     #100,CLKVCT
10 037640  005777 141652   TST    @CLKADR          ;;TEST FOR KW11-L PRESENT
11 037644  000421           BR     3$              ;;YES - KW11-L IS PRESENT
12 037646  022626           1$:  CMP     (SP)+,(SP)+  ;;RESTORE SP
13 037650  012737 037700 000004   MOV     #2$,ERRVEC      ;;SET UP FOR BUS TIMEOUT
14 037656  012737 172540 001516   MOV     #172540,CLKADR  ;;LOAD ADDRESSES FOR KW11-P CLOCK
15 037664  012737 000104 001520   MOV     #104,CLKVCT
16 037672  005777 141620   TST    @CLKADR          ;;TEST FOR KW11-P PRESENT
17 037676  000404           BR     3$              ;;YES - KW11-P IS PRESENT
18 037700  022626           2$:  CMP     (SP)+,(SP)+  ;;RESTORE SP
19 037702  062766 000002 000004   ADD     #2,4(SP)        ;;MOVE RETURN TO ERROR
20 037710           3$:
21 037710  012637 000006   MOV     (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
22 037714  012637 000004   MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
23 037720  000207           RTS     PC              ;;RETURN TO USER

```

```

1      .SBTTL  TIMEOUT SUBROUTINE
2
3      ;THIS SUBROUTINE WAITS FOR RDY = 1 AND GO = 0 OR FOR A TIMEOUT
4      ;GREATER THAN APPROX. 500 MSEC AND THEN RETURNS.
5
6      ;CALL:  JSR      PC,TIMOUT
7      ;      ???
8      ;      RETURN HERE
9
10     TIMOUT:
11     MOV      R0,-(SP)      ;;PUSH R0 ON STACK
12     MOV      R1,-(SP)      ;;PUSH R1 ON STACK
13     MOV      R2,-(SP)      ;;PUSH R2 ON STACK
14     MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
15     MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
16     MOV      #4$,ERRVEC    ;SETUP FOR BUS TIMEOUT - 04 TRAP
17     MOV      #PR6,ERRVEC+2
18     MOV      $BASE,R0      ;R0=BASE ADDRESS
19     MOV      CLKADR,R1     ;R1=CLOCK ADDRESS
20     MOV      #30.,R2       ;R2=NUMBER OF CLOCK CYCLES
21     1$:  CMP      R1,#172540 ;KW11-P CLOCK??
22     BNE      2$           ;NO!!
23     MOV      #1,2(R1)      ;SET COUNTER
24     2$:  MOV      #BIT2!BIT0,(R1) ;START COUNTER
25
26     3$:  MOV      RMCS1(R0),-(SP) ;GET STATUS
27     BIC      #^C<RDY!GO>,(SP)
28     CMP      #RDY,(SP)+    ;RDY=1,GO=0??
29     BEQ      5$           ;YES!!
30     BIT      #BIT7,(R1)    ;TIMER DONE??
31     BEQ      3$           ;NO!!
32     DEC      R2            ;DEC NUMBER OF CYCLES
33     BNE      1$           ;CONTINUE IF NOT DONE
34     BR      5$           ;"RDY" DID NOT SET OR "GO" DID NOT RESET
35     ;      WITHIN 500 MSEC AFTER THE COMMAND WAS ISSUED.
36     4$:  CMP      (SP)+,(SP)+ ;ADJUST STACK
37     ADD      #4,12(SP)     ;MOVE SP TO USER'S CALL
38     MOV      #7,@12(SP)   ;WRITE ERROR NUMBER
39     SUB      #2,12(SP)
40
41     5$:  MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERRVEC+2
42     MOV      (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
43     MOV      (SP)+,R2       ;;POP STACK INTO R2
44     MOV      (SP)+,R1       ;;POP STACK INTO R1
45     MOV      (SP)+,R0       ;;POP STACK INTO R0
46     RTS      PC            ;RETURN TO USER

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41 040106
42
43
44 040106 062716 000004
45 040112 105076 000000
46 040116 162716 000004
47
48
49 040122 013737 001344 001142
50 040130 042737 177770 001142
51 040136 013737 001234 001140
52 040144 042737 177770 001140
53 040152 123737 001140 001142
54
55 040160 001415
56 040162 062716 000004
57 040166 112776 000001 000000

```

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

PRIERR:

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

:REPORT AN ERROR IF THE WRONG UNIT IS SELECTED
MOV     RMCS2I,$BDDAT ;CORRECT UNIT SELECTED??
BIC     #^CUNTMSK,$BDDAT
MOV     $UNIT,$GDDAT  ;GOOD DATA FOR TYPEOUT
BIC     #^CUNTMSK,$GDDAT
CMPB    $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
:DRIVE NUMBERS
:YES!!
BEQ     1$
ADD     #4,(SP)
MOVB    #1,@(SP)    ;ERROR 1
    
```

```

58 040174 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
59 040200 004736                JSR      PC,@(SP)+      ;REPORT WRONG UNIT SELECTED
60 040202 162716 000010          SUB      #10,(SP)       ;RESTORE (SP)
61 040206 000240                NOP
62 040210 000137 040730          JMP      10$           ;SKIP OTHER CHECKS
63 040214
64
65                                ;REPORT AN ERROR IF THE DEVICE IS NOT AVAILABLE OR IF
66                                ;THE DEVICE IS NONEXISTANT
67 040214 032737 004000 001334    BIT      #DVA,RMCS11    ;DEVICE AVAILABLE??
68 040222 001045                BNE      5$            ;YES!!
69 040224 013737 001334 001140    MOV      RMCS11,$GDDAT ;EXPECTED STATUS
70 040232 052737 004000 001140    BIS      #DVA,$GDDAT
71 040240 013737 001334 001142    MOV      RMCS11,$BDDAT ;RECEIVED STATUS
72 040246 062716 000004                ADD      #4,(SP)
73 040252 112776 000002 000000    MOVB     #2,@(SP)      ;ERROR #2
74 040260 032737 010000 001344    BIT      #NED,RMCS21    ;WAS NED SET??
75 040266 001414                BEQ      2$            ;NO!!
76 040270 013737 001344 001140    MOV      RMCS21,$GDDAT ;EXPECTED STATUS
77 040276 013737 001344 001142    MOV      RMCS21,$BDDAT ;RECEIVED STATUS
78 040304 042737 010000 001140    BIC      #NED,$GDDAT
79 040312 112776 000003 000000    MOVB     #3,@(SP)      ;YES - CHANGE ERROR NUMBER
80 040320 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
81 040324 004736                JSR      PC,@(SP)+      ;REPORT DEVICE NOT AVAILABLE
82 040326 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
83 040332 000240                NOP
84 040334 000575                BR       10$           ;SKIP OTHER CHECKS
85 040336
86
87                                ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
88 040336 032737 000200 001334    BIT      #RDY,RMCS11    ;CONTROLLER READY??
89 040344 001030                BNE      7$            ;YES!!
90 040346 013737 001334 001140    MOV      RMCS11,$GDDAT ;EXPECTED STATUS
91 040354 052737 000200 001140    BIS      #RDY,$GDDAT
92 040362 042737 160001 001140    BIC      #SC!TRE!MCPE!GO,$GDDAT
93 040370 013737 001334 001142    MOV      RMCS11,$BDDAT ;RECEIVED STATUS
94 040376 062716 000004                ADD      #4,(SP)
95 040402 112776 000004 000000    MOVB     #4,@(SP)      ;ERROR #4
96 040410 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
97 040414 004736                JSR      PC,@(SP)+      ;REPORT CONTROLLER NOT READY
98 040416 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
99 040422 000240                NOP
100 040424 000541                BR       10$           ;SKIP OTHER CHECKS
101 040426
102
103                                ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
104 040426 032737 000001 001334    BIT      #GO,RMCS11     ;GO RESET??
105 040434 001431                BEQ      8$            ;YES!!
106 040436 032737 000200 001346    BIT      #DRY,RMDS1     ;DRIVE READY??
107 040444 001025                BNE      8$            ;YES!!
108 040446 013737 001334 001140    MOV      RMCS11,$GDDAT ;EXPECTED STATUS
109 040454 042737 160001 001140    BIC      #SC!TRE!MCPE!GO,$GDDAT
110 040462 013737 001334 001142    MOV      RMCS11,$BDDAT ;RECEIVED STATUS
111 040470 062716 000004                ADD      #4,(SP)
112 040474 112776 000005 000000    MOVB     #5,@(SP)      ;ERROR #5
113 040502 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
114 040506 004736                JSR      PC,@(SP)+      ;REPORT DRIVE NOT READY

```



```

115 040510 162716 000010          SUB    #10,(SP)      ;RESTORE (SP)
116 040514 000240          NOP
117 040516 000504          BR     10$
118 040520          8$:
119
120          ;REPORT AN ERROR IF THE RM CONTROLLER DETECTED BAD
121          ;PARITY ON THE MASSBUS CONTROL BUS
122 040520 032737 020000 001334  BIT    #MCPE,RMCS11 ;PARITY ERROR ??
123 040526 001425          BEQ    9$           ;NO!!
124 040530 013737 001334 001140  MOV    RMCS11,$GDDAT ;EXPECTED STATUS
125 040536 042737 160001 001140  BIC    #SC:TRÉ!MCPE!GO,$GDDAT
126 040544 013737 001334 001142  MOV    RMCS11,$BDDAT ;RECEIVED STATUS
127 040552 062716 000004          ADD    #4,(SP)      ;MOVE STACK TO USER'S ERROR
128 040556 112776 000013 000000  MOVVB #13,@(SP)    ;ERROR #13
129 040564 162716 000002          SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
130 040570 004736          JSR   PC,@(SP)+    ;REPORT ERROR VIA USER
131 040572 162716 000010          SUB    #10,(SP)    ;RESTORE STACK
132 040576 000240          NOP
133 040600 000453          BR     10$
134 040602          9$:
135
136          ;REPORT AN ERROR IF DETECTED A CONTROL BUS PARITY ERROR
137 040602 032737 000010 001350  BIT    #PAR,RMER11 ;WAS THERE A PARITY ERROR??
138 040610 001451          BEQ    11$         ;NO!!
139 040612 032737 000010 001376  BIT    #DPE,RMER21 ;WAS IT THE CONTROL BUS??
140 040620 001045          BNE    11$         ;NOT SURE!!
141 040622 032737 000010 001424  BIT    #PAR,RMER10 ;DID TEST SET PAR ??
142 040630 001413          BEQ    93$        ;NO!!
143 040632 010046          MOV    R0,-(SP)   ;;PUSH R0 ON STACK
144 040634 012700 001551          MOV    #PUTINX,R0 ;R0 POINTS TO INDEX TABLE
145 040640 122710 000014 91$:  CMPB  #RMER1,(R0)  ;SEARCH TABLE FOR RMER1
146 040644 001002          BNE    92$
147 040646 012600          MOV    (SP)+,R0  ;;POP STACK INTO R0
148 040650 000431          BR     11$        ;PAR WAS SET BY TEST
149 040652 105720 92$:  TSTB  (R0)+       ;END OF TABLE??
150 040654 100371          BPL    91$        ;NO!!
151 040656 012600          MOV    (SP)+,R0  ;;POP STACK INTO R0
152 040660 013737 001350 001140 93$:  MOV    RMER11,$GDDAT ;EXPECTED STATUS
153 040666 042737 000010 001140  BIC    #PAR,$GDDAT
154 040674 013737 001350 001142  MOV    RMER11,$BDDAT ;RECEIVED STATUS
155 040702 062716 000004          ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
156 040706 112776 000050 000000  MOVVB #50,@(SP)    ;WRITE THE ERROR NUMBER
157 040714 162716 000002          SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
158 040720 004736          JSR   PC,@(SP)+    ;REPORT THE ERROR
159 040722 162716 000010          SUB    #10,(SP)    ;MOVE SP TO NO ERROR RETURN
160 040726 000240          NOP
161 040730 062716 000010 10$:  ADD    #10,(SP)    ;RETURN TO ERROR
162 040734 000240 11$:  NOP           ;RETURN TO NO ERROR
163 040736 000207          RTS
164

```

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    RMCS10,515$    :STORE FUNCTION CODE
BIC    #^C<F0!F1!F2!F3!F4>,515$
ADD    #4,(SP)        :MOVE (SP) TO ERROR CALL
CLRB   @ (SP)         :CLEAR ERROR NUMBER
SUB    #4,(SP)        :MOVE (SP) TO NO ERROR RETURN
    
```

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

```

:REPORT ERROR IF DRIVE IS NOT READY, I.F., IF DRY = 0
BIT    #DRY,RMDSI    :DRIVE READY??
BNE    5$           :YES!!
MOV    RMDSI,$BDDAT :BAD DATA FOR TYPEOUT
BIC    #^CDRY,$BDDAT
MOV    #DRY,$GDDAT  :GOOD DATA FOR TYPEOUT
ADD    #4,(SP)
MOV    #10,@(SP)   :ERROR NUMBER
SUB    #2,(SP)     :MOVE SP TO RETURN FOR ERROR
JSR    PC,@(SP)+   :REPORT NOT READY
SUB    #10,(SP)    :RESTORE (SP) TO ERROR N
NOP
    
```

```

:REPORT ERROR IF GO BIT IS NOT RESET
5$: BIT    #GO,RMCS11 :GO BIT RESET??
BEQ    10$          :YES!!
MOV    RMCS11,$BDDAT :BAD DATA FOR TYPEOUT
BIC    #^CGO,$BDDAT
CLR    $GDDAT       :GOOD DATA FOR TYPEOUT
ADD    #4,(SP)
MOV    #11,@(SP)   :ERROR NUMBER
    
```

040740

040740	013737	001410	044574
040746	042737	177701	044574
040754	062716	000004	
040760	105076	000000	
040764	162716	000004	
040770	032737	000200	001346
040776	001024		
041000	013737	001346	001142
041006	042737	177577	001142
041014	012737	000200	001140
041022	062716	000004	
041026	112776	000010	000000
041034	162716	000002	
041040	004736		
041042	162716	000010	
041046	000240		
041050	032737	000001	001334
041056	001423		
041060	013737	001334	001142
041066	042737	177776	001142
041074	005037	001140	
041100	062716	000004	
041104	112776	000011	000000

```

58 041112 162716 000002      SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
59 041116 004736             JSR      PC,@(SP)+   ;REPORT DEVICE NOT AVAILABLE
60 041120 162716 000010      SUB      #10,(SP)   ;RESTORE (SP)
61 041124 000240             NOP
62
63
64 041126 013737 001334 001142 ;REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
65 041134 042737 177701 001142 10$:  MOV      RMCS11,$BDDAT ;IS FUNCTION CODE CORRECT??
66 041142 013737 044574 001140      BIC      #^C76,$BDDAT
67 041150 023737 001142 001140      MOV      515$,$GDDAT ;EXPECTED FUNCTION CODE
68 041156 001413             CMP      $BDDAT,$GDDAT
69 041160 062716 000004             BEQ      15$         ;YES!!
70 041164 112776 000012 000000      ADD      #4,(SP)
71 041172 162716 000002             MOV      #12,@(SP)   ;ERROR NUMBER
72 041176 004736             SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
73 041200 162716 000010      JSR      PC,@(SP)+   ;REPORT WRONG FUNCTION CODE
74 041204 000240             SUB      #10,(SP)   ;RESTORE (SP)
75 041206
76
77
78
79 041206 005037 001140             CLR      $GDDAT      ;EXPECT 'ERR' = 0
80 041212 005737 001350             TST      RMER1I      ;IS RMER1 = 0??
81 041216 001003             BNE      20$         ;NO!!
82 041220 005737 001376             TST      RMER2I      ;IS RMERZ = 0??
83 041224 001403             BEQ      25$         ;YES!!
84 041226 052737 040000 001140 20$:  BIS      #ERR,$GDDAT ;'ERR' SHOULD BE SET
85 041234 013737 001346 001142 25$:  MOV      RMDSI,$BDDAT
86 041242 042737 137777 001142      BIC      #^CERR,$BDDAT
87 041250 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS 'ERR' OK??
88 041256 001412             BEQ      30$         ;YES!!
89 041260 062716 000004             ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR
90 041264 112776 000047 000000      MOV      #47,@(SP)  ;WRITE ERROR NUMBER
91 041272 162716 000002             SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
92 041276 004736             JSR      PC,@(SP)+   ;REPORT INVALID COMP ERROR
93 041300 162716 000010      SUB      #10,(SP)
94
95
96
97
98 041304 005037 001140             CLR      $GDDAT      ;EXPECT 'TRE' = 0
99 041310 013746 001344             MOV      RMCS21,-(SP) ;WAS DLT, WCE, UPE, NED, NEM
100 041314 042726 000377            BIC      #377,(SP)+  ;PGE, MXF OR MDPE SET
101 041320 001010             BNE      35$         ;YES!!
102 041322 032737 040000 001346      BIT      #ERR,RMDSI  ;WAS EXCEPTION RECEIVED??
103 041330 001407             BEQ      40$         ;NO!!
104 041332 022737 000030 044574      CMP      #SEARCH,515$ ;WAS DATA TRANSFERRED??
105 041340 103003             BHIS     40$         ;NO!!
106 041342 052737 040000 001140 35$:  BIS      #TRE,$GDDAT ;'TRE' SHOULD BE SET
107 041350 013737 001334 001142 40$:  MOV      RMCS11,$BDDAT ;BAD DATA FOR TYPEOUT
108 041356 042737 137777 001142      BIC      #^CTRE,$BDDAT
109 041364 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS 'TRE' OK??
110 041372 001413             BEQ      45$         ;YES!!
111 041374 062716 000004             ADD      #4,(SP)     ;MOVE SP TO USER'S ERROR CALL
112 041400 112776 000014 000000      MOV      #14,@(SP)  ;WRITE ERROR NUMBER
113 041406 162716 000002             SUB      #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
114 041412 004736             JSR      PC,@(SP)+   ;REPORT TRE ERROR
    
```

```

115 041414 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
116 041420 000240          NOP
117 041422          45$:
118
119          ;*****
120          ;USING THE FUNCTION CODE TABLE, CHECK FOR THE FOLLOWING ERRORS:
121          ;      .STATUS BITS NOT SET THAT SHOULD BE SET, E.G., ATA AND ILF
122          ;      .STATUS BITS SET THAT SHOULD NOT BE SET, E.G., WCE AND ECH
123          ;NOTE THAT SOME ERROR BITS ARE CONDITIONAL ON THE COMMAND AND OTHER
124          ;STATUS CONDITIONS, E.G., WRITE LOCK ERROR SHOULD ONLY BE SET IF
125          ;WRITE LOCK IS ON AND THE COMMAND IS A WRITE.
126
127          ;GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE
128 041422 010046          MOV      RO,-(SP)      ;;PUSH RO ON STACK
129 041424 013700 044574      MOV      515$,RO      ;GET FUNCTION CODE
130 041430 016037 064406 044566  MOV      FNCDTB(RO),500$ ;STORE ENTRY
131 041436 012600          MOV      (SP)+,RO      ;;POP STACK INTO RO
132
133          ;REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
134          ;ATA IS NOT SET AND SHOULD BE SET.
135 041440 013737 044566 001140      MOV      500$,$GDDAT  ;GET EXPECTED ATA STATUS
136 041446 032737 040000 001346      BIT      #ERR,RMDSI   ;IS COMPOSITE ERROR SET ??
137 041454 001403          BEQ      50$         ;NO !!
138 041456 052737 100000 001140      BIS      #ATA,$GDDAT ;EXPECT AN ATTENTION
139 041464 042737 077777 001140 50$: BIC      #^CATA,$GDDAT ;STRIP DONT CARES
140 041472 013737 001346 001142      MOV      RMDSI,$BDDAT ;GET RECEIVED ATA
141 041500 042737 077777 001142      BIC      #^CATA,$BDDAT ;STRIP DONT CARES
142 041506 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS ATA OK ??
143 041514 001413          BEQ      55$         ;YES !!
144 041516 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
145 041522 112776 000006 000000      MOVB    #6,@(SP)     ;LOAD ERROR # IN CALL
146 041530 162716 000002          SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
147 041534 004736          JSR      PC,@(SP)+   ;REPORT ERROR
148 041536 162716 000010          SUB      #10,(SP)    ;RESTORE SP
149 041542 000240          NOP
150 041544          55$:
151
152          ;REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
153          ;WITH FUNCTION CODE TABLE
154 041544 013737 044566 001140      MOV      500$,$GDDAT ;GET EXPECTED ILF
155 041552 042737 177776 001140      BIC      #^CILF,$GDDAT ;CLEAR ALL OTHER BITS
156 041560 013737 001350 001142      MOV      RMER11,$BDDAT ;GET RECEIVED ILF
157 041566 042737 177776 001142      BIC      #^CILF,$BDDAT ;CLEAR ALL OTHER BITS
158 041574 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS ILF OK ??
159 041602 001412          BEQ      60$         ;YES !!
160 041604 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
161 041610 112776 000254 000000      MOVB    #254,@(SP)   ;WRITE ERROR NUMBER IN CALL
162 041616 162716 000002          SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
163 041622 004736          JSR      PC,@(SP)+   ;REPORT ERROR AND RETURN
164 041624 162716 000010          SUB      #10,(SP)    ;MOVE SP TO NO ERROR
165 041630 005037 001140 60$: CLR      $GDDAT      ;CLEAR EXPECTED STATUS
166
167          ;REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
168 041634 013746 044566          MOV      500$,-(SP) ;GET WCE STATUS ENABLE
169 041640 052716 137777          BIS      #^CWCE,(SP) ;SET ALL OTHER BITS
170 041644 013737 001344 001142      MOV      RMCS21,$BDDAT ;RECEIVED STATUS
171 041652 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR WCE IF ENABLED
    
```

```

172 041656 001412          BEQ      90$          ;BRANCH IF WCE OK
173 041660 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
174 041664 112776 000026 000000          MOVVB   #26,@(SP)    ;WRITE ERROR NUMBER
175 041672 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
176 041676 004736          JSR      PC,@(SP)+    ;REPORT ERROR
177 041700 162716 000010          SUB      #10,(SP)     ;RESTORE ERROR
178 041704          90$:
179
180          ;REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
181 041704 013746 044566          MOV      500$,-(SP)   ;GET OPI STATUS ENABLE
182 041710 052716 157777          BIS      #^COPI,(SP) ;SET ALL OTHER BITS
183 041714 013737 001350 001142          MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
184 041722 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR OPI IF ENABLED
185 041726 001412          BEQ      100$         ;BRANCH IF OPI OK
186 041730 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
187 041734 112776 000164 000000          MOVVB   #164,@(SP)   ;WRITE ERROR NUMBER IN CALL
188 041742 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
189 041746 004736          JSR      PC,@(SP)+    ;REPORT ERROR
190 041750 162716 000010          SUB      #10,(SP)     ;RESTORE SP
191 041754          100$:
192
193          ;REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
194          ;SET AND VV IS NOT RESET
195 041754 013746 044566          MOV      500$,-(SP)   ;GET IVC STATUS ENABLE
196 041760 032737 000100 001346          BIT      #VV,RMDSI    ;IS VV SET
197 041766 001402          BEQ      105$         ;NO !!
198 041770 042716 010000          BIC      #IVC,(SP)    ;YES - IVC SHOULD BE 0
199 041774 052716 167777          BIS      #^CIVC,(SP) ;SET ALL OTHER BITS
200 042000 013737 001376 001142          MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
201 042006 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR IVC IF ENABLED
202 042012 001412          BEQ      110$         ;BRANCH IF IVC OK
203 042014 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USERS ERROR CALL
204 042020 112776 000165 000000          MOVVB   #165,@(SP)   ;WRITE ERROR NUMBER IN CALL
205 042026 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
206 042032 004736          JSR      PC,@(SP)+    ;REPORT ERROR
207 042034 162716 000010          SUB      #10,(SP)     ;RESTORE SP TO NO ERROR
208 042040          110$:
209
210          ;BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
211          ; ALL WRITE ERRORS, I.E.,
212          ;     RMER1 - WLE, WCF
213          ;     RMER2 - DPE
214          ;     RMCS2 - UPE.
215          ;EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
216          ;WRITE ERROR ENABLE BIT IS RESET.
217
218          ;REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
219          ;THE DRIVE IS NOT WRITE PROTECTED
220 042040 012746 177777          MOV      #-1,-(SP)    ;ASSUME WRITE ERRORS ENABLED
221 042044 032737 004000 044566          BIT      #WLE,500$    ;ARE WRITE ERRORS ENABLED ??
222 042052 001404          BEQ      115$         ;NO !!
223 042054 032737 004000 001346          BIT      #WRL,RMDSI   ;IS THE DRIVE WRITE PROTECTED ??
224 042062 001002          BNE      120$         ;YES !!
225 042064 042716 004000          BIC      #WLE,(SP)    ;RESET WLE ENABLE
226 042070 013737 001350 001142          MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
227 042076 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR WLE IF ENABLED
228 042102 001412          BEQ      125$         ;BRANCH IF WLE OK
    
```

```

229 042104 062716 000004          ADD    #4,(SP)      ;MOVE SP TO USERS ERROR CALL
230 042110 112776 000023 000000    MOV    #23,@(SP)   ;WRITE ERROR NUMBER IN CALL
231 042116 162716 000002          SUB    #2,(SP)     ;MOVE SP TO ERROR RETURN
232 042122 004736          JSR    PC,@(SP)+   ;REPORT ERROR AND RETURN
233 042124 162716 000010          SUB    #10,(SP)    ;RESTORE SP TO NO ERROR
234 042130          125$:
235
236          ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
237 042130 012746 177777          MOV    #-1,-(SP)  ;ASSUME WRITE ERRORS ENABLED
238 042134 032737 004000 044566    BIT    #WLE,500$  ;ARE WRITE ERRORS ENABLED ??
239 042142 001002          BNE    130$       ;YES !!
240 042144 042716 000040          BIC    #WCF,(SP)  ;DISABLE WCF ERROR
241 042150 013737 001350 001142 130$:  MOV    RMER11,$BDDAT ;GET RECEIVED STATUS
242 042156 042637 001142          BIC    (SP)+,$BDDAT ;RESET WCF IF ENABLED
243 042162 001412          BEQ    135$       ;BRANCH IF WCF OK
244 042164 062716 000004          ADD    #4,(SP)    ;MOVE SP TO USERS ERROR CALL
245 042170 112776 000025 000000    MOV    #25,@(SP)  ;WRITE ERROR NUMBER IN CALL
246 042176 162716 000002          SUB    #2,(SP)    ;MOVE SP TO ERROR RETURN
247 042202 004736          JSR    PC,@(SP)+  ;REPORT ERROR
248 042204 162716 000010          SUB    #10,(SP)   ;RESTORE SP TO NO ERROR
249 042210          135$:
250
251          ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
252 042210 012746 177777          MOV    #-1,-(SP)  ;ASSUME WRITE ERRORS ARE ENABLED
253 042214 032737 004000 044566    BIT    #WLE,500$  ;ARE WRITE ERRORS ENABLED ??
254 042222 001002          BNE    140$       ;YES !!
255 042224 042716 000010          BIC    #DPE,(SP)  ;RESET DPE ENABLE
256 042230 013737 001376 001142 140$:  MOV    RMER21,$BDDAT ;GET RECEIVED STATUS
257 042236 042637 001142          BIC    (SP)+,$BDDAT ;RESET DPE IF ENABLED
258 042242 001412          BEQ    145$       ;BRANCH IF DPE OK
259 042244 062716 000004          ADD    #4,(SP)    ;MOVE SP TO USERS ERROR CALL
260 042250 112776 000040 000000    MOV    #40,@(SP)  ;WRITE ERROR NUMBER IN CALL
261 042256 162716 000002          SUB    #2,(SP)    ;MOVE SP TO ERROR RETURN
262 042262 004736          JSR    PC,@(SP)+  ;REPORT ERROR
263 042264 162716 000010          SUB    #10,(SP)   ;RESTORE SP TO NO ERROR
264 042270          145$:
265
266          ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
267 042270 012746 177777          MOV    #-1,-(SP)  ;ASSUME WRITE ERRORS ARE ENABLED
268 042274 032737 004000 044566    BIT    #WLE,500$  ;ARE WRITE ERRORS ENABLED ??
269 042302 001002          BNE    150$       ;YES !!
270 042304 042716 020000          BIC    #UPE,(SP)  ;DISABLE UPE ERROR
271 042310 013737 001344 001142 150$:  MOV    RMCS21,$BDDAT ;GET RECEIVED STATUS
272 042316 042637 001142          BIC    (SP)+,$BDDAT ;RESET UPE IF ENABLED
273 042322 001412          BEQ    155$       ;BRANCH IF UPE OK
274 042324 062716 000004          ADD    #4,(SP)    ;MOVE SP TO USERS ERROR CALL
275 042330 112776 000024 000000    MOV    #24,@(SP)  ;WRITE ERROR NUMBER IN CALL
276 042336 162716 000002          SUB    #2,(SP)    ;MOVE SP TO ERROR RETURN
277 042342 004736          JSR    PC,@(SP)+  ;REPORT ERROR AND RETURN
278 042344 162716 000010          SUB    #10,(SP)   ;MOVE SP TO NO ERROR
279 042350          155$:
280
281          ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
282 042350 013746 044566          MOV    500$,-(SP) ;GET IAE ENABLE
283 042354 052716 175777          BIS    #^CIAE,(SP) ;SET ALL OTHER BITS
284 042360 013737 001350 001142    MOV    RMER11,$BDDAT ;GET RECEIVED STATUS
285 042366 042637 001142          BIC    (SP)+,$BDDAT ;CLEAR IAE IF ENABLED
    
```

```

286 042372 001412          BEQ      160$          ;BRANCH IF IAE IS OK
287 042374 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
288 042400 112776 000166 000000  MOVB    #166,@(SP)    ;WRITE ERROR NUMBER
289 042406 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
290 042412 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
291 042414 162716 000010    SUB      #10,(SP)     ;MOVE SP TO NO ERROR
292 042420          160$:
293
294          ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
295          ; ALL READ/WRITE ERRORS, I.E.,
296          ;
297          ; RMCS1 - TRE
298          ; RMCS2 - DLT,NEM,MXF
299          ; RMDS - LBT
300          ; RMER1 - AOE
301          ;NOTE:
302          ; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
303          ;CYLINDER REGISTER IS WRITTEN
304          ;NOTE:
305          ; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
306          ;NOTE:
307          ; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
308
309          ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
310 042420 012746 177777    MOV      #-1,-(SP)    ;ASSUME ERRORS ARE ENABLED
311 042424 032737 001000 044566  BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
312 042432 001002          BNE      165$          ;YES !!
313 042434 042716 100000    BIC      #DLT,(SP)    ;RESET DLT ENABLE
314 042440 013737 001344 001142 165$:  MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
315 042446 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
316 042452 001412          BEQ      170$          ;BRANCH IF DLT IS OK
317 042454 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
318 042460 112776 000032 000000  MOVB    #32,@(SP)    ;WRITE ERROR NUMBER IN CALL
319 042466 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
320 042472 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
321 042474 162716 000010    SUB      #10,(SP)     ;MOVE SP TO NO ERROR
322 042500          170$:
323
324          ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
325 042500 012746 177777    MOV      #-1,-(SP)    ;ASSUME ERRORS ARE ENABLED
326 042504 032737 001000 044566  BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
327 042512 001002          BNE      175$          ;YES !!
328 042514 042716 004000    BIC      #NEM,(SP)    ;DISABLE NEM
329 042520 013737 001344 001142 175$:  MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
330 042526 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
331 042532 001412          BEQ      180$          ;BRANCH IF NEM IS OK
332 042534 062716 000004    ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
333 042540 112776 000167 000000  MOVB    #167,@(SP)    ;WRITE ERROR NUMBER IN CALL
334 042546 162716 000002    SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
335 042552 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
336 042554 162716 000010    SUB      #10,(SP)     ;MOVE SP TO NO ERROR
337 042560          180$:
338
339          ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
340 042560 012746 177777    MOV      #-1,-(SP)    ;ASSUME ERRORS ARE ENABLED
341 042564 032737 001000 044566  BIT      #AOE,500$    ;ARE DATA ERRORS ENABLED ??
342 042572 001002          BNE      185$          ;YES !!
    
```

```

343 042574 042716 001000          BIC      #MXF,(SP)      ;DISABLE MXF ERROR
344 042600 013737 001344 001142 185$: MOV      RMCS21,$BDDAT ;GET RECEIVED STATUS
345 042606 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR MXF IF ENABLED
346 042612 001412          BEQ      190$          ;BRANCH IF MXF IS OK
347 042614 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
348 042620 112776 000033 000000  MOVB    #33,@(SP)     ;WRITE ERROR NUMBER IN CALL
349 042626 162716 000002          SUB      #2,(SP)      ;MOVE SP TO ERROR RETURN
350 042632 004736          JSR     PC,@(SP)+    ;REPORT ERROR AND RETURN
351 042634 162716 000010          SUB      #10,(SP)     ;MOVE SP TO NO ERROR
352 042640          190$:
353
354          ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
355 042640 012746 177777          MOV      #-1,-(SP)   ;ASSUME DATA ERRORS ARE ENABLED
356 042644 032737 001000 044566  BIT      #AOE,500$   ;ARE DATA ERRORS EANBLED ??
357 042652 001404          BEQ      191$          ;NO !!
358 042654 032737 002000 001346  BIT      #LBT,RMDSI  ;IS LBT ALSO SET ??
359 042662 001002          BNE      195$          ;YES !!
360 042664 042716 001000          BIC      #AOE,(SP)   ;DISABLE AOE
361 042670 013737 001350 001142 195$: MOV      RMER11,$BDDAT ;GET RECEIVED STATUS
362 042676 042637 001142          BIC      (SP)+,$BDDAT ;CLEAR AOE IF ENABLED
363 042702 001412          BEQ      200$          ;BRANCH IF AOE IS OK
364 042704 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
365 042710 112776 000020 000000  MOVB    #20,@(SP)    ;WRITE ERROR NUMBER
366 042716 162716 000002          SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
367 042722 004736          JSR     PC,@(SP)+    ;REPORT ERROR AND RETURN
368 042724 162716 000010          SUB      #10,(SP)    ;MOVE SP TO NO ERROR
369 042730          200$:
370
371          ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
372          ;HEADER ERRORS, I.E.,
373          ;
374          ;   RMER1 - HCRC,HCE,FER
375          ;   RMER2 - BSE
376          ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
377 042730 032737 002000 001366  BIT      #HCI,RMOFI  ;IS HCI SET ??
378 042736 001403          BEQ      201$          ;NO !!
379 042740 042737 000200 044566  BIC      #HCE,500$   ;YES - DISABLE ALL HEADER ERRORS
380 042746          201$:
381
382          ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
383 042746 012746 177777          MOV      #-1,-(SP)   ;ASSUME ERRORS ENABLED
384 042752 032737 000200 044566  BIT      #HCE,500$   ;ARE HEADER ERRORS ENABLED ??
385 042760 001002          BNE      205$          ;YES !!
386 042762 042716 000400          BIC      #HCRC,(SP)  ;DISABLE HCRC
387 042766 013737 001350 001142 205$: MOV      RMER11,$BDDAT ;GET RECEIVED STATUS
388 042774 042637 001142          BIC      (SP)+,$BDDAT ;RESET HCRC IF ENABLED
389 043000 001412          BEQ      210$          ;BRANCH IF HCRC IS OK
390 043002 062716 000004          ADD      #4,(SP)     ;MOVE SP TO USERS ERROR CALL
391 043006 112776 000035 000000  MOVB    #35,@(SP)    ;WRITE ERROR NUMBER IN CALL
392 043014 162716 000002          SUB      #2,(SP)     ;MOVE SP TO ERROR RETURN
393 043020 004736          JSR     PC,@(SP)+    ;REPORT ERROR AND RETURN
394 043022 162716 000010          SUB      #10,(SP)    ;MOVE SP TO NO ERROR
395 043026          210$:
396
397          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
398 043026 012746 177777          MOV      #-1,-(SP)   ;ASSUME ERRORS ENABLED
399 043032 032737 000200 044566  BIT      #HCE,500$   ;ARE ERRORS ENABLED ??

```



```

400 043040 001002          BNE      215$      ;YES !!
401 043042 042716 000200    BIC      #HCE,(SP) ;DISABLE HCE
402 043046 013737 001350 001142 215$:  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
403 043054 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR HCE IF ENABLED
404 043060 001412          BEQ      220$      ;BRANCH IF HCE IS OK
405 043062 062716 000004    ADD      #4,(SP)    ;MOVE SP TO USERS ERROR CALL
406 043066 112776 000036 000000    MOV      #36,@(SP) ;WRITE ERROR NUMBER IN CALL
407 043074 162716 000002    SUB      #2,(SP)   ;MOVE SP TO ERROR RETURN
408 043100 004736          JSR      PC,@(SP)+ ;REPORT ERROR AND RETURN
409 043102 162716 000010    SUB      #10,(SP)  ;MOVE SP TO NO ERROR
410 043106          220$:
411
412          ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
413 043106 012746 177777    MOV      #-1,-(SP) ;ASSUME FER IS ENABLED
414 043112 032737 000200 044566    BIT      #HCE,500$ ;ARE HEADER ERRORS ENABLED ??
415 043120 001002          BNE      225$      ;YES !!
416 043122 042716 000020    BIC      #FER,(SP) ;DISABLE FER
417 043126 013737 001350 001142 225$:  MOV      RMER1I,$BDDAT ;GET RECEIVED STATUS
418 043134 042637 001142    BIC      (SP)+,$BDDAT ;RESET FER IF ENABLED
419 043140 001412          BEQ      230$      ;BRANCH IF FER OK
420 043142 062716 000004    ADD      #4,(SP)   ;MOVE SP TO USERS ERROR CALL
421 043146 112776 000037 000000    MOV      #37,@(SP) ;WRITE ERROR NUMBER IN CALL
422 043154 162716 000002    SUB      #2,(SP)   ;MOVE SP TO ERROR RETURN
423 043160 004736          JSR      PC,@(SP)+ ;REPORT ERROR AND RETURN
424 043162 162716 000010    SUB      #10,(SP)  ;MOVE SP TO NO ERROR
425 043166          230$:
426
427          ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
428 043166 012746 177777    MOV      #-1,-(SP) ;ASSUME ERRORS ENABLED
429 043172 032737 000200 044566    BIT      #HCE,500$ ;ARE THEY ENABLED ??
430 043200 001002          BNE      235$      ;YES !!
431 043202 042716 100000    BIC      #BSE,(SP) ;DISABLE BSE
432 043206 013737 001376 001142 235$:  MOV      RMER2I,$BDDAT ;GET RECEIVED STATUS
433 043214 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR BSE IF ENABLED
434 043220 001412          BEQ      240$      ;BRANCH IF BSE OK
435 043222 062716 000004    ADD      #4,(SP)   ;MOVE SP TO USERS ERROR CALL
436 043226 112776 000354 000000    MOV      #354,@(SP) ;WRITE ERROR NUMBER
437 043234 162716 000002    SUB      #2,(SP)   ;MOVE SP TO ERROR RETURN
438 043240 004736          JSR      PC,@(SP)+ ;REPORT ERROR AND RETURN
439 043242 162716 000010    SUB      #10,(SP)  ;MOVE SP TO NO ERROR
440 043246          240$:
441
442          ;BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
443          ;FIELD ERRORS, I.E.,
444          ;
445          ;   RMCS2 - MDPE
446          ;   RMER1 - DCK,ECH
447          ;NOTE:
448          ;   ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
449          ;   DCK IS SET.
450          ;REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
451 043246 012746 177777    MOV      #-1,-(SP) ;ASSUME ENABLED
452 043252 032737 000100 044566    BIT      #ECH,500$ ;ARE DATA FIELD ERRORS ENABLED ??
453 043260 001002          BNE      245$      ;YES !!
454 043262 042716 000400    BIC      #MDPE,(SP) ;DISBALE MDPE
455 043266 013737 001344 001142 245$:  MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
456 043274 042637 001142    BIC      (SP)+,$BDDAT ;CLEAR MDPE IF ENABLED
    
```

SECONDARY ERROR CHECK SUBROUTINE

```

457 043300 001412          BEQ      250$      ;BRANCH IF MDPE OK
458 043302 062716 000004    ADD      #4,(SP)    ;MOVE SP TO USERS ERROR CALL
459 043306 112776 000027 000000  MOVVB   #27,@(SP)  ;WRITE ERROR NUMBER IN CALL
460 043314 162716 000002    SUB      #2,(SP)    ;MOVE SP TO ERROR RETURN
461 043320 004736          JSR     PC,@(SP)+  ;REPORT ERROR AND RETURN
462 043322 162716 000010    SUB      #10,(SP)   ;MOVE SP TO NO ERROR
463 043326          250$:
464
465          ;REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
466 043326 012746 177777    MOV     #-1,-(SP)   ;ASSUME ENABLED
467 043332 032737 000100 044566  BIT     #ECH,500$   ;ARE THEY ENABLED ??
468 043340 001002          BNE     255$        ;YES !!
469 043342 042716 100000    BIC     #DCK,(SP)   ;DISABLE DCK
470 043346 013737 001350 001142 255$:  MOV     RMER11,$BDDAT ;GET RECEIVED STATUS
471 043354 042637 001142    BIC     (SP)+,$BDDAT ;CLEAR DCK IF ENABLED
472 043360 001412          BEQ     260$        ;BRANCH IF DCK IS OK
473 043362 062716 000004    ADD     #4,(SP)    ;MOVE SP TO USERS ERROR CALL
474 043366 112776 000030 000000  MOVVB   #30,@(SP)  ;WRITE ERROR NUMBER IN CALL
475 043374 162716 000002    SUB     #2,(SP)    ;MOVE SP TO ERROR RETURN
476 043400 004736          JSR     PC,@(SP)+  ;REPORT ERROR AND RETURN
477 043402 162716 000010    SUB     #10,(SP)   ;MOVE SP TO NO ERROR
478 043406          260$:
479
480          ;REPORT ERROR IF ECH IS SET AND,
481          ;DATA FIELD ERRORS ARE NOT ENABLED, OR
482          ;ECI IS SET, OR
483          ;DCK IS NOT SET.
484 043406 012746 177777    MOV     #-1,-(SP)   ;ASSUME ENABLED
485 043412 032737 000100 044566  BIT     #ECH,500$   ;ARE ERRORS ENABLED ??
486 043420 001410          BEQ     265$        ;NO !!
487 043422 032737 004000 001366  BIT     #ECI,RMOF1  ;IS ECI SET ??
488 043430 001004          BNE     265$        ;YES !!
489 043432 032737 100000 001350  BIT     #DCK,RMER11 ;IS DCK ALSO SET ??
490 043440 001002          BNE     270$        ;YES !!
491 043442 042716 000100 265$:  BIC     #ECH,(SP)   ;DISABLE ECH
492 043446 013737 001350 001142 270$:  MOV     RMER11,$BDDAT ;GET RECEIVED STATUS
493 043454 042637 001142    BIC     (SP)+,$BDDAT ;CLEAR ECH IF ENABLED
494 043460 001412          BEQ     275$        ;BRANCH IF ECH IS OK
495 043462 062716 000004    ADD     #4,(SP)    ;MOVE SP TO USERS ERROR CALL
496 043466 112776 000031 000000  MOVVB   #31,@(SP)  ;WRITE ERROR NUMBER IN CALL
497 043474 162716 000002    SUB     #2,(SP)    ;MOVE SP TO ERROR RETURN
498 043500 004736          JSR     PC,@(SP)+  ;REPORT ERROR AND RETURN
499 043502 162716 000010    SUB     #10,(SP)   ;MOVE SP TO NO ERROR
500 043506          275$:
501

```

```

1
2
3
4
5 043506 022737 000030 044574          CMP    #SEARCH,515$    ;WAS DATA TRANSFERRED ?
6 043514 103402          BLO    280$           ;BR IF YES
7 043516 000137 044540          JMP    355$           ;NO - EXIT
8
9
10 043522 013737 001336 001142 280$:  ;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
11 043530 001421          MOV    RMWCI,$BDDAT   ;WORD COUNT ZERO??
12 043532 032737 040000 001334          BEQ    285$           ;YES
13 043540 001015          BIT    #TRE,RMCS1I   ;TRANSFER ERROR DETECTED??
14 043542 062716 000004          BNE    285$           ;YES!!
15 043546 112776 000015 000000          ADD    #4,(SP)
16 043554 005037 001140          MOVB  #15,@(SP)      ;ERROR NUMBER
17 043560 162716 000002          CLR    $GDDAT        ;GOOD DATA FOR TYPEOUT
18 043564 004736          SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
19 043566 162716 000010          JSR   PC,@(SP)+      ;REPORT WORD COUNT NOT ZERO
20 043572 000240          SUB    #10,(SP)     ;RESTORE (SP)
21
22
23 043574 013737 001336 001140 285$:  ;REPORT ERROR IF RMBA IS NOT CORRECT
24 043602 163737 001412 001140          MOV    RMWCI,$GDDAT  ;GET WORD COUNT AT END OF TRANSFER AND
25 043610 006337 001140          SUB    RMWCO,$GDDAT  ;SUBTRACT STARTING WORD COUNT.
26 043614 063737 001414 001140          ASL   $GDDAT        ;* 2
27
28 043622 032737 000010 001344          ADD    RMBAO,$GDDAT  ;ADD STARTING BUS ADDRESS
29 043630 001403          BIT    #BAI,RMCS2I  ;WAS BUS ADDRESS INHIBIT (BAI) SET ??
30 043632 013737 001414 001140          BEQ    290$           ;NO !!
31
32 043640 023737 001140 001340 290$:  ;ADDRESS SHOULD NOT HAVE CHANGED
33 043646 001416          MOV    RMBAI,$BDDAT  ;BUS ADDRESS OK??
34 043650 013737 001340 001142          BEQ    295$           ;YES!!
35 043656 062716 000004          MOV    RMBAI,$BDDAT  ;BAD DATA FOR TYPEOUT
36 043662 112776 000016 000000          ADD    #4,(SP)
37 043670 162716 000002          MOVB  #16,@(SP)     ;ERROR NUMBER
38 043674 004736          SUB    #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
39 043676 162716 000010          JSR   PC,@(SP)+      ;REPORT UNEXPECTED ADDRESS
40 043702 000240          SUB    #10,(SP)     ;RESTORE (SP)
41
42
43 043704 005046          ;COMPUTE NUMBER OF SECTORS TRANSFERRED FROM WORD COUNT
44 043706 013746 001336 295$:  CLR    -(SP)         ;NUMBER OF SECTORS TRANSFERRED
45 043712 163716 001412          MOV    RMWCI,-(SP)   ;GET WORD COUNT AT END OF TRANSFER AND
46
47 043716 012746 000400          SUB    RMWCO,(SP)   ;SUBTRACT STARTING WORD COUNT.
48 043722 032737 000002 001410          MOV    #256,-(SP)   ;ASSUME 256. WORDS PER SECTOR
49 043730 001402          BIT    #BIT1,RMCS10 ;HEADER & DATA COMMAND ??
50 043732 062716 000002          BEQ    300$           ;NO !!
51
52 043736 005266 000004 300$:  INC    4(SP)         ;CHANGE TO 258. WORDS PER SECTOR
53 043742 161666 000002          ;INCREMENT SECTOR COUNT
54 043746 003373          SUB    (SP),2(SP)   ;SUBTRACT ONE SECTOR'S WORTH
55 043750 022626          BGT    300$         ;CONTINUE IF NOT DONE
56
57
;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM

```

```

58                                     ;NUMBER OF SECTORS
59 043752 013737 001444 044566      MOV     RMDCO,500$      ;STORE ORIGINAL CYLINDER
60 043760 013737 001416 044570      MOV     RMDAO,505$     ;STORE ORIGINAL TRACK
61 043766 013737 001416 044572      MOV     RMDAO,510$     ;STORE ORIGINAL SECTOR
62 043774 013737 001332 044576      MOV     LSTRK,520$    ;STORE LAST TRACK,
63 044002 000337 044576              SWAB    520$          ;GET TRACK ADDRESS TO LO BYTE AND
64 044006 005237 044576              INC     520$          ;INCREMENT TO GET TOTL # OF TRACKS.
65
66 044012 042737 000377 044570      BIC     #^C<TADMSK>,505$ ;SAVE TRACK ADDRESS BITS AND
67 044020 000337 044570              SWAB    505$          ;SWAP TRACK ADDRESS TO LOW BYTE.
68 044024 042737 177400 044572      BIC     #^C<SADMSK>,510$ ;SAVE SECTOR ADDRESS BITS
69 044032 062637 044572              ADD     (SP)+,510$
70
71 044036 023727 044572 000040 310$:  CMP     510$,#32.      ;SECTOR OVEFLOWED??
72 044044 103406                    BLO    315$          ;NO!!
73 044046 005237 044570              INC     505$          ;INCREMENT TRACK
74 044052 162737 000040 044572      SUB     #32.,510$     ;ADJUST SECTOR
75 044060 000766                    BR     310$          ;TRY AGAIN
76
77 044062 023737 044570 044576 315$:  CMP     505$,520$     ;TRACK OVERFLOWED??
78 044070 103407                    BLO    320$          ;NO!!
79 044072 005237 044566              INC     500$          ;INCREMENT CYLINDER
80 044076 163737 044576 044570      SUB     520$,505$     ;ADJUST TRACK
81 044104 000766                    BR     315$          ;TRY AGAIN
82 044106 000240                    NOP
83
84                                     ;REPORT ERROR IF "LBT" IS NOT CORRECT
85 044110 005037 001140 001466 320$:  CLR     $GDDAT        ;SET GOOD DATA FOR LBT = 0
86 044110 005037 001140 001466      CMP     500$,#822.    ;SHOULD LBT BE SET??
87 044114 023727 044566 001466      BLOS   325$          ;NO!!
88 044122 101407                    BIT     #IAE,RMER11   ;WAS IAE SET ??
89 044124 032737 002000 001350      BNE    325$          ;YES - LBT SHOULD NOT BE SET
90 044132 001003                    MOV     #LBT,$GDDAT   ;SET GOOD DATA FOR LBT = 1
91 044134 012737 002000 001140 325$:  MOV     RMDSI,$BDDAT  ;BAD DATA FOR TYPEOUT
92 044142 013737 001346 001142      BIC     #^CLBT,$BDDAT
93 044150 042737 175777 001142      CMP     $GDDAT,$BDDAT ;IS LBT CORRECT??
94 044156 023737 001140 001142      BEQ    330$          ;YES!!
95 044164 001413                    ADD     #4,(SP)
96 044166 062716 000004              MOVVB  #17,@(SP)      ;ERROR NUMBER
97 044172 112776 000017 000000      SUB     #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
98 044200 162716 000002              JSR    PC,@(SP)+     ;REPORT LBT IS WRONG
99 044204 004736                    SUB     #10,(SP)     ;RESTORE (SP)
100 044206 162716 000010              NOP
101 044212 000240
102
103                                     ;REPORT ERROR IF "AOE" IS INCORRECT
104 044214 005037 001140 001350 330$:  CLR     $GDDAT        ;SET FOR AOE = 0
105 044220 032737 002000 001350      BIT     #IAE,RMER11   ;WAS "IAE" DETECTED??
106 044226 001031                    BNE    340$          ;YES-"AOE" SHOULD BE ZERO
107 044230 023727 044566 001466      CMP     500$,#822.    ;SHOULD AOE BE SET??
108 044236 101425                    BLOS   340$          ;NO!!
109 044240 005737 044570              TST    505$          ;MAYBE
110 044244 001012                    BNE    335$          ;YES
111 044246 005737 044572              TST    510$
112 044252 001007                    BNE    335$          ;YES !!
113 044254 032737 000010 044574      BIT     #F2,515$     ;WAS THIS READ OR WRITE CHECK ??
114 044262 001413                    BEQ    340$          ;NO !!
    
```

```

115 044264 005737 001336          TST    RMWCI          ;WAS ALL DATA TRANSFERRED ??
116 044270 001410          BEQ    340$          ;YES !!
117 044272 012737 001000 001140 335$: MOV    #AOE,$GDDAT   ;SET FOR AOE = 1
118 044300 005037 044570          CLR    505$         ;CLEAR EXPECTED TRACK
119 044304 012737 000001 044572 340$: MOV    #1,510$      ;EXPECT SECTOR = 1
120 044312 013737 001350 001142 340$: MOV    RMER11,$BDDAT ;BAD DATA FOR TYPEOUT
121 044320 042737 176777 001142 340$: BIC    #^CAOE,$BDDAT
122 044326 023737 001140 001142 340$: CMP    $GDDAT,$BDDAT ;IS AOE CORRECTY??
123 044334 001413          BEQ    345$         ;YES!!
124 044336 062716 000004          ADD    #4,(SP)
125 044342 112776 000020 000000 340$: MOVB  #20,@(SP)   ;ERROR NUMBER
126 044350 162716 000002          SUB    #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
127 044354 004736          JSR   PC,@(SP)+    ;REPORT AOE IS WRONG
128 044356 162716 000010          SUB    #10,(SP)   ;RESTORE (SP)
129 044362 000240          NOP
130
131          ;REPORT ERROR IF RMDA IS NOT CORRECT
132 044364 032737 002000 001350 345$: BIT    #IAE,RMER11 ;WAS THERE AN IAE ERROR ??
133 044372 001062          BNE    355$         ;YES - DONT CHECK RMDA,RMDC
134 044374 013737 044570 001140 345$: MOV    505$,$GDDAT ;SETUP EXPECTED DISK ADDRESS
135 044402 000337 001140          SWAB  $GDDAT
136 044406 113737 044572 001140 345$: MOVB  510$,$GDDAT
137 044414 013737 001342 001142 345$: MOV    RMDA1,$BDDAT ;SETUP RECEIVED DISK ADDRESS
138 044422 023737 001140 001142 345$: CMP    $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
139 044430 001413          BEQ    350$         ;BRANCH IF EQUAL
140 044432 062716 000004          ADD    #4,(SP)
141 044436 112776 000021 000000 345$: MOVB  #21,@(SP)   ;ERROR NUMBER
142 044444 162716 000002          SUB    #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
143 044450 004736          JSR   PC,@(SP)+    ;REPORT BAD DISK ADDRESS
144 044452 162716 000010          SUB    #10,(SP)   ;RESTORE (SP)
145 044456 000240          NOP
146
147          ;REPORT ERROR IF RMDC IS INCORRECT
148 044460 013737 044566 001140 350$: MOV    500$,$GDDAT ;SETUP EXPECTED CYLINDER
149 044466 042737 176000 001140 350$: BIC    #^C1777,$GDDAT
150 044474 013737 001370 001142 350$: MOV    RMDC1,$BDDAT ;SETUP RECEIVED CYLINDER
151 044502 023737 001140 001142 350$: CMP    $GDDAT,$BDDAT ;COMPAPE CYLINDERS
152 044510 001413          BEQ    355$         ;BRANCH IF EQUAL
153 044512 062716 000004          ADD    #4,(SP)
154 044516 112776 000022 000000 350$: MOVB  #22,@(SP)   ;ERROR NUMBER
155 044524 162716 000002          SUB    #2,(SP)     ;MOVE SP TO RETURN FOR ERROR
156 044530 004736          JSR   PC,@(SP)+    ;REPORT BAD CYLINDER
157 044532 162716 000010          SUB    #10,(SP)   ;RESTORE (SP)
158 044536 000240          NOP
159
160 044540 062716 000004          355$: ADD    #4,(SP) ;MOVE (SP) TO ERROR CALL
161 044544 105776 000000          TSTB  @(SP)        ;WAS ERROR FOUND??
162 044550 001403          BEQ    360$
163 044552 062716 000004          ADD    #4,(SP) ;MOVE (SP) TO ERROR RETURN
164 044556 000402          BR    365$
165 044560 162716 000004          360$: SUB    #4,(SP) ;MOVE (SP) TO NO ERROR RETURN
166 044564 000207          365$: RTS    PC
167
168 044566 000000          500$: .WORD 0 ;CYLINDER
169 044570 000000          505$: .WORD 0 ;TRACK
170 044572 000000          510$: .WORD 0 ;SECTOR
171 044574 000000          515$: .WORD 0 ;FUNCTION CODE
    
```

172 044576 000000
173
174

520\$: .WORD 0

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

```

1      .SBTTL  COMPOSITE ERROR CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE CHECKS THE STORED CONTENTS OF RMER1 AND
4      ;RMER2 AFTER MASKING EACH REGISTER WORD WITH THE USER'S STATUS
5      ;MASKS AND REPORTS AN ERROR IF ANY BITS ARE LEFT ON AFTER
6      ;THE MASKS ARE APPLIED.
7
8      ;CALL:
9      ;(1)  JSR      PC,CMPERRSTS
10     ;      .WORD   MASK FOR ERROR REGISTER 1
11     ;      .WORD   MASK FOR ERROR REGISTER 2
12     ;      BR      ???      RETURN HERE IF NO ERROR
13     ;      NOP     RETURN HERE TO REPORT AN ERROR
14     ;      ERROR   ERROR NUMBER DEFINED BY SUB
15     ;      JSR     PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
16     ;      ???     RETURN HERE IF NO MORE ERRORS
17
18     ;NOTE: BITS TO BE MASKED SHOULD BE ONE; BITS TO BE TESTED SHOULD
19     ;BE ZERO
20
21     044600
22
23     ;MASK AND STORE THE CONTENTS OF RMER1 AND RMER2
24     044600 013737 001350 001176      MOV     RMER1I,$TMP1      ;STORE RMER1 AT TEMP STORAGE
25     044606 047637 000000 001176      BIC     @($P),$TMP1      ;MASK RMER1
26     044614 062716 000002             ADD     #2,$(SP)         ;MOVE SP TO NEXT MASK
27     044620 013737 001376 001200      MOV     RMER2I,$TMP2      ;STORE RMER2 AT TEMP STORAGE
28     044626 047637 000000 001200      BIC     @($P),$TMP2      ;MASK RMER2
29
30
31     ;CLEAR USER'S ERROR CALL
32     044634 062716 000006             ADD     #6,$(SP)         ;MOVE SP TO USER'S ERROR CALL
33     044640 105076 000000             CLRB   @($P)            ;CLEAR ERROR NUMBER
34     044644 162716 000004             SUB     #4,$(SP)         ;LEAVE SP AT NO ERROR RETURN
35
36     ;SEE IF THERE WERE ANY ERRORS IN RMER1, I.E., $TMP1
37     044650 005737 001176             TST    $TMP1            ;ANY ERRORS TO REPORT??
38     044654 001420                     BEQ    5$                ;NO !!
39     044656 013737 001176 001142      MOV     $TMP1,$BDDAT     ;RECEIVED STATUS FOR TYPEOUT
40     044664 005037 001140             CLR    $GDDAT           ;EXPECTED STATUS FOR TYPEOUT
41     044670 062716 000004             ADD     #4,$(SP)         ;MOVE SP TO USER'S ERROR CALL
42     044674 112776 000066 000000      MOV     #66,@($P)        ;CORRECTABLE DATA CHECK ERROR #
43     044702 162716 000002             SUB     #2,$(SP)         ;MOVE SP TO RETURN FOR ERROR
44     044706 004736                     JSR    PC,@($P)+         ;REPORT ERROR VIA USER
45     044710 162716 000010             SUB     #10,$(SP)        ;MOVE SP BACK TO BRANCH
46     044714 000240                     NOP
47     044716
48
49     5$:
50     ;SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 ($TMP2)
51     044716 005737 001200             TST    $TMP2            ;ANY ERRORS IN RMER2?
52     044722 001420                     BEQ    10$               ;NO!!
53     044724 013737 001200 001142      MOV     $TMP2,$BDDAT     ;RECEIVED STATUS FOR TYPEOUT
54     044732 005037 001140             CLR    $GDDAT           ;EXPECTED STATUS FOR TYPEOUT
55     044736 062716 000004             ADD     #4,$(SP)         ;MOVE SP TO USER'S ERROR CALL
56     044742 112776 000067 000000      MOV     #67,@($P)        ;WRITE ERROR NUMBER IN USER'S CALL
57     044750 162716 000002             SUB     #2,$(SP)         ;MOVE SP TO RETURN FOR ERROR
    
```

```

58 044754 004736          JSR   PC,@(SP)+      ;REPORT ERROR VIA USER
59 044756 162716 000010   SUB   #10,(SP)         ;MOVE SP TO NO ERROR RETURN
60 044762 000240          NOP
61 044764          10$:
62
63          ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
64 044764 062716 000004   ADD   #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
65 044770 105776 000000   TSTB @ (SP)          ;WAS THERE AN ERROR CALLED??
66 044774 001403          BEQ   20$             ;NO!!
67 044776 062716 000004   ADD   #4,(SP)         ;YES - MOVE SP TO ERROR RETURN
68 045002 000402          BR   30$
69 045004 162716 000004   20$: SUB   #4,(SP)         ;MOVE SP TO NO ERROR RETURN
70 045010 000207          30$: RTS   PC         ;RETURN TO USER
71

```



```

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 045012  DEVSEL:
13
14     ;CLEAR USER'S ERROR CALL
15 045012 062716 000004      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR
16 045016 105076 000000      CLRB    @10(SP)         ;CLEAR LOW ORDER BYTE OF CALL
17 045022 162716 000004      SUB      #4,(SP)          ;MOVE SP BACK
18
19     ;SAVE USER'S INFORMATION AND SETUP REGISTERS
20 045026 013746 000004      MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
21 045032 013746 000006      MOV      ERRVEC+2,-(SP)  ;;PUSH ERRVEC+2 ON STACK
22 045036 010046              MOV      R0,-(SP)         ;;PUSH R0 ON STACK
23 045040 010146              MOV      R1,-(SP)         ;;PUSH R1 ON STACK
24 045042 012737 045162 000004  MOV      #20$,ERRVEC      ;SETUP FOR BUS TIMEOUT
25 045050 012737 000300 000006  MOV      #PR6,ERRVEC+2
26 045056 013700 001276              MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
27 045062 013701 001464              MOV      TSTQUE,R1       ;R1 POINTS TO DEVICE NUMBER
28
29     ;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
30
31 045066 111160 000010              MOVVB   (R1),RMCS2(R0)    ;WRITE UNIT SELECT BITS
32 045072 016037 000000 001176      MOV      RMCS1(R0),$TMP1  ;GET 'DVA' STATUS
33 045100 016037 000010 001174      MOV      RMCS2(R0),$TMP0  ;GET 'NED' STATUS
34
35 045106 032737 010000 001174      BIT      #NED,$TMP0      ;IS DEVICE NONEXISTENT ?
36 045114 001407              BEQ     10$              ;NO!!
37 045116 062766 000004 000010      ADD      #4,10(SP)       ;MOVE SP TO USERS ERROR CALL
38 045124 112776 000111 000010      MOVVB   #111,@10(SP)    ;WRITE ERROR NUMBER
39 045132 000422              BR      30$
40
41 045134 032737 004000 001176 10$:  BIT      #DVA,$TMP1      ;IS DEVICE AVAILABLE ?
42 045142 001021              BNE     35$              ;YES!!
43 045144 062766 000004 000010      ADD      #4,10(SP)       ;MOVE SP TO USERS ERROR CALL
44 045152 112776 000112 000010      MOVVB   #112,@10(SP)    ;WRITE ERROR NUMBER
45 045160 000407              BR      30$
46
47     ;HANDLE BUS TIMEOUT
48 20$:  CMP      (SP)+,(SP)+    ;ADJUST SP
49 045162 022626              ADD      #4,10(SP)       ;MOVE SP TO USERS ERROR CALL
50 045164 062766 000004 000010      MOVVB   #113,@10(SP)    ;WRITE BUS TIMEOUT ERROR NUMBER
51 045172 112776 000113 000010 30$:  SUB      #2,10(SP)       ;ADJUST RETURN TO 'NOP' PRECEDING
52 045200 162766 000002 000010              ;THE ERROR CALL
53
54     ;RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
55 35$:  MOV      (SP)+,R1        ;;POP STACK INTO R1
56 045206 012601              MOV      (SP)+,R0        ;;POP STACK INTO R0
57 045210 012600              MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
58 045212 012637 000006

```

```
52 045216 012637 000004      MOV      (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC  
53 045222 000207      RTS      PC                ;EXIT
```

```

1      .SBTTL  SEEK STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE VERIFIES THE RESULTS OF SEEK TESTS USING STATUS
4      ;STORED IN THE GET BUFFER AND TEST PARAMETERS STORED IN THE PUT BUFFER.
5
6
7      ;
8      ;THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS DETECTED
9      ;AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN THE "ERROR" TRAP
10     ;OF THE CALLING ROUTINE.  SEEK STATUS IS CHECKED AS FOLLOWS:
11
12     ;CALL:
13     ;(1)  JSR    PC,SEKSTS
14           BR      ???          RETURN HERE IF NO ERROR
15           NOP
16           ERROR          RETURN HERE TO REPORT AN ERROR
17           JSR    PC,@(SP)+    ERROR NUMBER DEFINED BY SUB
18           ???          GO BACK TO SUB FOR MORE ERROR CHECKS
19           RETURN HERE IF NO MORE ERRORS
20
21     SEKSTS:
22
23     ;CLEAR USERS' ERROR CALL
24     NOP
25     ADD    #4,(SP)          ;MOVE (SP) TO ERROR CALL
26     CLRB  @ (SP)          ;CLEAR ERROR NUMBER
27     SUB    #4,(SP)          ;MOVE (SP) TO NO ERROR RETURN
28     CLR    300$          ;CLEAR ERROR FLAGS
29
30     ;TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
31     ;LOCAL REGISTERS, I.E., "PAR" = 1 AND "DPE" = 0
32     BIT    #PAR,RMER11    ;WAS PARITY ERROR DETECTED??
33     BEQ    1$            ;NO!!
34     BIT    #DPE,RMER21    ;WAS IT DUE TO CONTROL BUS??
35     BNE    1$            ;NOT SURE!!
36
37     ;REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
38     CLR    $GDDAT          ;EXPECTED STATUS
39     MOV    RMER11,$BDDAT   ;RECEIVED STATUS
40     ADD    #4,(SP)          ;MOVE STACK TO USER'S ERROR
41     MOVB  #50,@(SP)        ;ERROR #50
42     SUB    #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
43     JSR    PC,@(SP)+
44     SUB    #10,(SP)         ;RESTORE STACK
45     BR     3$            ;IAE SHOULD BE ZERO
46
47     ;DETERMINE THE VALUE OF "IAE" STATUS BASED ON TRACK, SECTOR AND CYLINDER
48     ;ALSO, SET "SKI" IF CYLINDER ADDRESS IS TOO LARGE.
49     1$: MOV    #IAE,$GDDAT  ;SETUP FOR IAE = 1
50         BIS    #SKI,300$    ;SETUP FOR SKI = 1
51         CMP    RMDCO,#822.  ;GREATER THAN LAST CYLINDER ?
52         BHI    3$          ;YES - CYLINDER IS INVALID
53         BIC    #SKI,300$    ;CLEAR SKI ERROR FLAG
54
55     CMPB  RMDAO+1,LSTRK+1  ;GREATER THAN LAST TRACK ?
56     BHI  3$          ;YES - TRACK IS INVALID
57
58     CMPB  RMDAO,#29.      ;SECTOR > 29. ?
59     BLOS  2$          ;BR IF NO

```

```

58 045400 032737 010000 001442      BIT      #FMT16,RMOFO      ;18 BIT FORMAT ?
59 045406 001406                BEQ      3$              ;YES - SECTOR IS INVALID FOR 18 BIT MODE
60 045410 123727 001416 000037      CMPB    RMDAO,#31.      ;SECTOR > 31. ?
61 045416 101002                BHI      3$              ;YES - SECTOR IS INVALID
62
63 045420 005037 001140      2$:     CLR      $GDDAT      ;"IAE" SHOULD = 0
64
65                                ;COMPARE EXPECTED AND RECIEVED "IAE" STATUS
66 045424 013737 001350 001142      3$:     MOV      RMER1I,$BDDAT ;IS IAE OK??
67 045432 042737 175777 001142      BIC     #^CIAE,$BDDAT    ;SAVE IAE BIT FOR COMPARE
68 045440 023737 001140 001142      CMP     $GDDAT,$BDDAT   ;CORRECT "IAE" STATUS ?
69 045446 001004                BNE     35$             ;BR IF NO
70 045450 042737 040000 046462      BIC     #SKI,300$       ;CLEAR SKI FLAG
71 045456 000413                BR      5$              ;GO CHECK NEXT ERROR
72 045460
73                                35$:
74 045460 062716 000004      ;REPORT INCORRECT "IAE" STATUS VIA USER'S ERROR CALL
75 045464 112776 000051 000000      ADD     #4,(SP)
76 045472 162716 000002      MOVB   #51,@(SP)       ;ERROR 51
77 045476 004736                SUB     #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
78 045500 162716 000010      JSR    PC,@(SP)+       ;REPORT INCORRECT IAE
79 045504 000240                SUB     #10,(SP)        ;RESTORE (SP)
80 045506
81                                5$:
82                                ;REPORT ANY IVC ERROR AS
83                                ; IVC ERROR WITH VOLUME VALID ZERO
84                                ; ERRONEOUS IVC ERROR, VOLUME VALID IS SET
85 045506 032737 010000 001376      BIT     #IVC,RMER2I     ;IVC ERROR??
86 045514 001427                BEQ     52$             ;NO!!
87 045516 005037 001140      CLR     $GDDAT         ;EXPECTED STATUS
88 045522 013737 001376 001142      MOV     RMER2I,$BDDAT  ;RECEIVED STATUS
89 045530 062716 000004      ADD     #4,(SP)        ;MOVE SP TO USER'S ERROR
90 045534 112776 000060 000000      MOVB   #60,@(SP)       ;ERROR 60 IF VV = 0
91 045542 032737 000100 001346      BIT     #VV,RMDSI
92 045550 001403                BEQ     51$             ;ERROR 61 IF VV = 1
93 045552 112776 000061 000000      MOVB   #61,@(SP)
94 045560 162716 000002      51$:   SUB     #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
95 045564 004736                JSR    PC,@(SP)+       ;REPORT ERROR VIA USER
96 045566 162716 000010      SUB     #10,(SP)        ;RESTORE SP
97 045572 000240                NOP
98
99 045574 013737 001376 001142      52$:   MOV     RMER2I,$BDDAT  ;RECEIVED STATUS
100 045602 042737 137777 001142      BIC     #^CSKI,$BDDAT  ;CLEAR DONT CARES
101 045610 013737 046462 001140      MOV     300$,$GDDAT    ;GET EXPECTED SKI STATUS
102 045616 042737 137777 001140      BIC     #^CSKI,$GDDAT  ;CLEAR DONT CARES
103 045624 001417                BEQ     53$             ;BRANCH IF 0 EXPECTED
104
105                                ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
106 045626 032737 040000 001142      BIT     #SKI,$BDDAT    ;WAS SKI DETECTED ??
107 045634 001032                BNE     54$             ;YES !!
108 045636 062716 000004      ADD     #4,(SP)        ;MOVE SP TO USERS ERROR CALL
109 045642 112776 000267 000000      MOVB   #267,@(SP)     ;WRITE ERROR NUMBER
110 045650 162716 000002      SUB     #2,(SP)        ;MOVE SP TO ERROR RETURN
111 045654 004736                JSR    PC,@(SP)+       ;REPORT ERROR, AND RETURN
112 045656 162716 000010      SUB     #10,(SP)        ;MOVE SP TO NO ERROR
113 045662 000443                BR      6$              ;GO TO NEXT ERROR CHECK
114 045664                                53$:

```

```

115
116 ;REPORT ERROR IF SKI IS SET
117 045664 032737 040000 001142 BIT #SKI,$BDDAT ;IS SKI SET ??
118 045672 001413 BEQ 54$ ;NO - SKI IS OK
119 045674 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
120 045700 112776 000054 000000 MOVB #54,@(SP) ;LOAD ERROR NUMBER
121 045706 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
122 045712 004736 JSR PC,@(SP)+ ;REPORT SEEK ERROR
123 045714 162716 000010 SUB #10,(SP) ;RESTORE (SP)
124 045720 000240 NOP
125
126 ;REPORT ANY DEVICE CHECK
127 045722 032737 000200 001376 54$: BIT #DVC,RMER21 ;WAS THERE DVC DURING SEEK??
128 045730 001420 BEQ 6$ ;NO!!
129 045732 005037 001140 CLR $GDDAT ;EXPECTED STATUS
130 045736 013737 001376 001142 MOV RMER21,$BDDAT ;RECEIVED STATUS
131 045744 062716 000004 ADD #4,(SP)
132 045750 112776 000055 000000 MOVB #55,@(SP) ;ERROR #55
133 045756 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
134 045762 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
135 045764 162716 000010 SUB #10,(SP) ;RESTORE SP
136 045770 000240 NOP
137
138 ;REPORT ANY "OPI" ERROR AS OPI WITH MOL = 0, OR OPI
139 ;BECAUSE ON CYLINDER LATCH DIDN'T RESET
140 045772 032737 020000 001350 6$: BIT #OPI,RMER11 ;"OPI" ERROR??
141 046000 001427 BEQ 8$ ;NO!!
142 046002 005037 001140 CLR $GDDAT ;EXPECTED STATUS
143 046006 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
144 046014 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
145 046020 112776 000052 000000 MOVB #52,@(SP) ;LOAD ERROR NUMBER
146 046026 032737 010000 001346 BIT #MOL,RMDSI ;WAS MEDIUM ON LINE??
147 046034 001403 BEQ 7$ ;NO!!
148 046036 112776 000053 000000 MOVB #53,@(SP) ;YES - CHANGE ERROR NUMBER
149 046044 162716 000002 7$: SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
150 046050 004736 JSR PC,@(SP)+ ;REPORT "OPI" ERROR
151 046052 162716 000010 SUB #10,(SP) ;RESTORE (SP)
152 046056 000240 NOP
153
154 ;SEE IF "PIP" = 0, AND "ATA", "MOL" AND "VV" = 1
155 046060 013746 001346 8$: MOV RMDSI,-(SP)
156 046064 042716 047677 BIC #^C<ATA!PIP!MOL!VV>,(SP)
157 046070 022726 110100 CMP #ATA!MOL!VV,(SP)+
158 046074 001002 BNE 9$ ;ERROR IN RMDS
159 046076 000137 046432 JMP 14$ ;RMDS IS OK
160
161 ;REPORT ERROR IF MOL = 0 AND OPI = 0
162 046102 032737 010000 001346 9$: BIT #MOL,RMDSI ;IS MOL RESET??
163 046110 001030 BNE 10$ ;NO - MOL IS SET
164 046112 032737 020000 001350 BIT #OPI,RMER11 ;WAS OPI SET
165 046120 001024 BNE 10$ ;YES - DONT REPORT ERROR
166 046122 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
167 046130 052737 010000 001140 BIS #MOL,$GDDAT
168 046136 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
169 046144 062716 000004 ADD #4,(SP)
170 046150 112776 000062 000000 MOVB #62,@(SP)
171 046156 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR

```

```

172 046162 004736          JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
173 046164 162716 000010  SUB    #10,(SP)
174 046170 000240          NOP
175
176          ;REPORT AN ERROR IF "PIP" IS STIL SET AND SKI NOT SET
177 046172 032737 020000 001346 10$: BIT    #PIP,RMDSI      ;IS "PIP" STILL SET??
178 046200 001430          BEQ    11$              ;NO!!
179 046202 032737 040000 001376  BIT    #SKI,RMER2I     ;WAS "SKI"SET??
180 046210 001024          BNE    11$              ;YES-DONT REPORT PIP
181 046212 013737 001346 001140  MOV    RMDSI,$GDDAT    ;EXPECTED STATUS
182 046220 042737 020000 001142  BIC    #PIP,$BDDAT
183 046226 013737 001346 001142  MOV    RMDSI,$BDDAT    ;RECEIVED STATUS
184 046234 062716 000004          ADD    #4,(SP)         ;MOVE (SP) TO ERROR
185 046240 112776 000056 000000  MOVB  #56,@(SP)        ;LOAD ERROR NUMBER
186 046246 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
187 046252 004736          JSR    PC,@(SP)+      ;REPORT "PIP" SET AFTER SEEK
188 046254 162716 000010  SUB    #10,(SP)        ;RESTORE (SP)
189 046260 000240          NOP
190
191          ;REPORT AN ERROR IF "ATA" IS NOT SET
192 046262 032737 100000 001346 11$: BIT    #ATA,RMDSI     ;WAS "ATA" SET ??
193 046270 001024          BNE    13$              ;YES!!
194 046272 013737 001346 001140  MOV    RMDSI,$GDDAT    ;EXPECTED STATUS
195 046300 052737 110600 001140  BIS    #ATA!MOL!DPR!DRY,$GDDAT
196 046306 013737 001346 001142  MOV    RMDSI,$BDDAT    ;RECEIVED STATUS
197 046314 062716 000004          ADD    #4,(SP)         ;MOVE (SP) TO ERROR
198 046320 112776 000057 000000  MOVB  #57,@(SP)        ;LOAD ERROR NUMBER
199 046326 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
200 046332 004736          JSR    PC,@(SP)+      ;REPORT ATTENTION NOT SET DURING
201          ;SEEK TEST
202 046334 162716 000010  SUB    #10,(SP)        ;RESTORE (SP)
203 046340 000240          NOP
204
205          ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
206 046342 032737 000100 001346 13$: BIT    #VV,RMDSI      ;IS VV = 0 ??
207 046350 001030          BNE    14$              ;NO!!
208 046352 032737 010000 001376  BIT    #IVC,RMER2I     ;IS IVC ALSO 0 ??
209 046360 001024          BNE    14$              ;NO - IVC IS SET
210 046362 013737 001346 001140  MOV    RMDSI,$GDDAT    ;EXPECTED STATUS
211 046370 052737 000100 001140  BIS    #VV,$GDDAT
212 046376 013737 001346 001142  MOV    RMDSI,$BDDAT    ;RECEIVED STATUS
213 046404 062716 000004          ADD    #4,(SP)
214 046410 112776 000064 000000  MOVB  #64,@(SP)        ;ERROR #64
215 046416 162716 000002          SUB    #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
216 046422 004736          JSR    PC,@(SP)+
217 046424 162716 000010  SUB    #10,(SP)
218 046430 000240          NOP
219 046432          14$:
220
221          ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
222 046432 000240          NOP
223 046434 062716 000004          ADD    #4,(SP)         ;MOVE (SP) TO ERROR CALL
224 046440 105776 000000          TSTB  @(SP)           ;WAS ERROR CALLED??
225 046444 001403          BEQ    15$              ;NO!!
226 046446 062716 000004          ADD    #4,(SP)         ;MOVE TO ERROR RETURN
227 046452 000402          BR    16$

```

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

MACRO V03.01 11-APR-80 13:17:48 PAGE 29-4

C 16

SEQ 0197

229 046454 162716 000004
230 046460 000207
231
232 046462 000000
233

15\$: SUB #4,(SP)
16\$: RTS PC
300\$: .WORD 0

;MOVE (SP) TO NO ERROR RETURN
;RETURN
;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      046464 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
14      046466 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
15      046470 013746 000004  MOV      ERRVEC,-(SP)  ;;PUSH ERRVEC ON STACK
16      046474 013746 000006  MOV      ERRVEC+2,-(SP) ;;PUSH ERRVEC+2 ON STACK
17      046500 012737 046540 000004  MOV      #10$,ERRVEC   ;SETUP FOR BUS TIMEOUT
18      046506 012737 000300 000006  MOV      #PR6,ERRVEC+2
19      046514 013700 001276      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
20      046520 012760 000040 000010  MOV      #CLR,RMCS2(R0) ;CLEAR MASSBUS
21      046526 013701 001464      MOV      TSTQUE,R1     ;GET DEVICE UNDER TEST
22      046532 111160 000010      MOVVB   (R1),RMCS2(R0) ;SELECT DEVICE
23      046536 000412      BR      20$
24
25      10$:  CMP      (SP)+,(SP)+  ;ADJUST STACK
26      ADD      #4,10(SP)      ;MOVE SP TO USER'S ERROR CALL
27      MOVVB   #7,@10(SP)     ;WRITE THE ERROR NUMBER
28      SUB      #2,10(SP)     ;ADJUST SP TO RETURN TO ERROR
29
30      20$:  MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
31      MOV      (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
32      MOV      (SP)+,R1       ;;POP STACK INTO R1
33      MOV      (SP)+,R0       ;;POP STACK INTO R0
34      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 STATUS CHECK SUBROUTINES
:*****
.SBTTL CONTROLLER CLEAR STATUS CHECK SUBROUTINE

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

:STATUS PERTINENT TO THE DEVICE IS NOT CHECKED. IN PARTICULAR, THE
:FOLLOWING STATUS BITS ARE NOT CHECKED:
:
:   ATA,ERR,PIP,MOL,WRL,LBT,PGM,VV,OM,UNS,SKI,DVC
:
:CALL:
:(1) JSR    PC,CLRSTS          RETURN HERE IF NO ERROR
:     BR     ???              RETURN HERE TO REPORT AN ERROR
:     NOP
:     ERROR  ERROR NUMBER DEFINED BY SUB
:     JSR    PC,@(SP)+        GO BACK TO SUB FOR MORE ERROR CHECKS
:     ???
:                               RETURN HERE IF NO MORE ERRORS

CLPSTS:

: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
4$:    MOV     RMCS1I,$BDDAT    ;VERIFY RMCS1
BIC     #SC,$BDDAT            ;IGNORE SPECIAL CONDITION
MOV     #DVA!RDY,$GDDAT      ;EXPECT DVA & RDY
CMP     $GDDAT,$BDDAT        ;COMPARE EXPECTED, RECEIVED
BEQ     5$                    ;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
5$:    CLR     $GDDAT          ;VERIFY RMBA IS ZERO
MOV     RMBAI,$BDDAT
BEQ     7$                    ;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
7$:    MOV     RMCS2I,$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)

```

046602				
046602	062716	000004		
046606	105076	000000		
046612	162716	000004		
046616	013737	001334	001142	
046624	042737	100000	001142	
046632	012737	004200	001140	
046640	023737	001140	001142	
046646	001413			
046650	062716	000004		
046654	112776	000126	000000	
046662	162716	000002		
046666	004736			
046670	162716	000010		
046674	000240			
046676	005037	001140		
046702	013737	001340	001142	
046710	001413			
046712	062716	000004		
046716	112776	000127	000000	
046724	162716	000002		
046730	004736			
046732	162716	000010		
046736	000240			
046740	013737	001344	001142	
046746	010146			
046750	005046			
046752	013701	001464		
046756	111116			

```

58 046760 052716 000100      BIS      #IR,(SP)
59 046764 012637 001140      MOV      (SP)+,$GDDAT
60 046770 012601 001140      MOV      (SP)+,R1          ;;POP STACK INTO R1
61 046772 023737 001140 001142    CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED & RECEIVED
62 047000 001413 000004      BEQ      9$              ;;BRANCH IF EQUAL
63 047002 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
64 047006 112776 000130 000000    MOVB    #130,@(SP)       ;;WRITE ERROR NUMBER IN CALL
65 047014 162716 000002      SUB      #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
66 047020 004736 000010      JSR     PC,@(SP)+        ;;REPORT ERROR VIA USER
67 047022 162716 000010      SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
68 047026 000240      NOP
69                                ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
70 047030 005037 001140 9$:      CLR      $GDDAT          ;;VERIFY RMER1
71 047034 013737 001350 001142    MOV      RMR1I,$BDDAT
72 047042 042737 040000 001142    BIC     #UNS,$BDDAT      ;;IGNORE UNSAFE
73 047050 001413 000004      BEQ      13$            ;;BRANCH IF ZERO
74 047052 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
75 047056 112776 000131 000000    MOVB    #131,@(SP)       ;;WRITE ERROR NUMBER IN CALL
76 047064 162716 000002      SUB      #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
77 047070 004736 000010      JSR     PC,@(SP)+        ;;REPORT ERROR VIA USER
78 047072 162716 000010      SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
79 047076 000240      NOP
80                                ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
81 047100 013737 001360 001142    13$:    MOV      RMMR1I,$BDDAT    ;;VERIFY RMMR
82 047106 042737 000046 001142    BIC     #WC:LS:LST,$BDDAT ;;IGNORE WORD CLOCK, SCT, TRK
83 047114 012737 000010 001140    MOV      #MWD,$GDDAT     ;;EXPECT WRITE DATA BIT
84 047122 023737 001140 001142    CMP      $GDDAT,$BDDAT    ;;COMPARE EXPECTED AND RECEIVED
85 047130 001413 000004      BEQ      17$            ;;BRANCH IF 0
86 047132 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
87 047136 112776 000133 000000    MOVB    #133,@(SP)       ;;WRITE ERROR NUMBER IN CALL
88 047144 162716 000002      SUB      #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
89 047150 004736 000010      JSR     PC,@(SP)+        ;;REPORT ERROR VIA USER
90 047152 162716 000010      SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
91 047156 000240      NOP
92                                ;REPORT AN ERROR IF RMEC2 IS NOT RESET
93 047160 005037 001140 17$:    CLR      $GDDAT          ;;EXPECT ZEROS
94 047164 013737 001402 001142    MOV      RMEC2I,$BDDAT    ;;VERIFY RMEC2=0
95 047172 001413 000004      BEQ      19$
96 047174 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
97 047200 112776 000135 000000    MOVB    #135,@(SP)       ;;WRITE ERROR NUMBER IN CALL
98 047206 162716 000002      SUB      #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
99 047212 004736 000010      JSR     PC,@(SP)+        ;;REPORT ERROR VIA USER
100 047214 162716 000010      SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
101 047220 000240      NOP
102                                ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
103 047222 013737 001374 001142    19$:    MOV      RMMR2I,$BDDAT    ;;VERIFY RMMR2
104 047230 042737 140000 001142    BIC     #RQA!RQB,$BDDAT
105 047236 012737 011777 001140    MOV      #TST!1777,$GDDAT ;;EXPECT TEST BIT ON
106 047244 023737 001140 001142    CMP      $GDDAT,$BDDAT
107 047252 001413 000004      BEQ      21$
108 047254 062716 000004      ADD      #4,(SP)         ;;MOVE SP TO USER'S ERROR CALL
109 047260 112776 000136 000000    MOVB    #136,@(SP)       ;;WRITE ERROR NUMBER IN CALL
110 047266 162716 000002      SUB      #2,(SP)         ;;MOVE SP TO RETURN FOR ERROR
111 047272 004736 000010      JSR     PC,@(SP)+        ;;REPORT ERROR VIA USER
112 047274 162716 000010      SUB      #10,(SP)        ;;MOVE SP BACK TO NO ERROR
113 047300 000240      NOP
114                                ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC
    
```

```

115 047302 005037 001140      21$: CLR    $GDDAT      ;EXPECT ALL ZEROS
116 047306 013737 001376 001142  MOV    RMER21,$BDDAT ;VERIFY RMER2
117 047314 042737 040200 001142  BIC    #SKI!DVC,$BDDAT ;IGNORE DEVICE ERRORS
118 047322 001413                BEQ    215$         ;BRANCH IF OTHER BITS 0
119 047324 062716 000004                ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
120 047330 112776 000174 000000  MOVB   #174,@(SP)   ;WRITE ERROR NUMBER IN CALL
121 047336 162716 000002                SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
122 047342 004736                JSR    PC,@(SP)+    ;REPORT ERROR VIA USER
123 047344 162716 000010                SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR
124 047350 000240                NOP
125                                ;REPORT ERROR IF RMDS NOT INITIALIZED
126 047352 013737 001346 001142  215$: MOV    RMDSI,$BDDAT ;TEST DRIVE STATUS REGISTER
127 047360 042737 177177 001142  BIC    #^C<DRY!DPR>,$BDDAT
128 047366 012737 000600 001140  MOV    #DPR!DRY,$GDDAT ;EXPECTED DRIVE STATUS
129 047374 023737 001140 001142  CMP    $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
130 047402 001413                BEQ    22$         ;BRANCH IF EQUAL
131 047404 062716 000004                ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
132 047410 112776 000134 000000  MOVB   #134,@(SP)   ;WRITE ERROR NUMBER
133 047416 162716 000002                SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
134 047422 004736                JSR    PC,@(SP)+    ;REPORT ERROR TO USER
135 047424 162716 000010                SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR
136 047430 000240                NOP
137 047432 062716 000004                22$: ADD    #4,(SP)      ;MOVE SP TO ERROR CALL
138 047436 105776 000000                TSTB   @(SP)        ;WAS AN ERROE DETECTED??
139 047442 001403                BEQ    23$         ;NO!!
140 047444 062716 000004                ADD    #4,(SP)      ;YES - MOVE TO ERROR RETURN
141 047450 000402                BR     24$
142 047452 162716 000004                23$: SUB    #4,(SP)      ;MOVE SP TO NO ERROR RETURN
143 047456 000240                24$: NOP
144 047460 000207                RTS    PC
145

```

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

047462

047462 062716 000004
 047466 105076 000000
 047472 162716 000004
 047476 032737 000100 001346
 047504 001024
 047506 013737 001346 001140
 047514 052737 000100 001140
 047522 013737 001346 001142
 047530 062716 000004
 047534 112776 000155 000000
 047542 162716 000002
 047546 004736
 047550 162716 000010
 047554 000240
 047556
 047556 032737 010000 001346
 047564 001024
 047566 013737 001346 001140
 047574 052737 010000 001140
 047602 013737 001346 001142
 047610 062716 000004
 047614 112776 000041 000000
 047622 162716 000002
 047626 004736
 047630 162716 000010
 047634 000240
 047636
 047636 032737 060007 001350
 047644 001570
 047646 032737 040000 001350
 047654 001424
 047656 013737 001350 001142

```

.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,@(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  @ (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,@(SP)
      SUB    #2,(SP)
      JSR    PC,@(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,@(SP)
      SUB    #2,(SP)
      JSR    PC,@(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,RMER11
      BEQ    7$

:REPORT AN ERROR IF 'UNS' IS SET
      BIT    #UNS,RMER11
      BEQ    3$
      MOV    RMER11,$BDDAT

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

58	047664	013737	001350	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
59	047672	042737	040000	001140	BIC	#UNS,\$GDDAT	
60	047700	062716	000004		ADD	#4,(SP)	:MOVE SP TO ERROR CALL
61	047704	112776	000042	000000	MOVB	#42,@(SP)	:WRITE NUMBER OF ERROR IN CALL
62	047712	162716	000002		SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
63	047716	004736			JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
64	047720	162716	000010		SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
65	047724	000240			NOP		
66	047726						
67							
68							
69	047726	032737	020000	001350	:REPORT ANY OPI ERROR		
70	047734	001424			BIT	#OPI,RMER11	:WAS OPI SET ??
71	047736	013737	001350	001142	BEQ	4\$:NO!!
72	047744	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
73	047752	042737	020000	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
74	047760	062716	000004		BIC	#OPI,\$GDDAT	
75	047764	112776	000043	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
76	047772	162716	000002		MOVB	#43,@(SP)	:WRITE NUMBER OF ERROR IN CALL
77	047776	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
78	050000	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
79	050004	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
80	050006				NOP		
81							
82							
83	050006	032737	000004	001350	:REPORT ANY RMR ERROR		
84	050014	001424			BIT	#RMR,RMER11	:WAS RMR SET??
85	050016	013737	001350	001142	BEQ	5\$:NO!!
86	050024	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
87	050032	042737	000004	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
88	050040	062716	000004		BIC	#RMR,\$GDDAT	
89	050044	112776	000044	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
90	050052	162716	000002		MOVB	#44,@(SP)	:WRITE NUMBER OF ERROR IN CALL
91	050056	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
92	050060	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
93	050064	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
94	050066				NOP		
95							
96							
97	050066	032737	000002	001350	:REPORT ANY ILR ERROR		
98	050074	001424			BIT	#ILR,RMER11	:WAS ILR SET??
99	050076	013737	001350	001142	BEQ	6\$:NO!!
100	050104	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
101	050112	042737	000002	001140	MOV	RMER11,\$GDDAT	:EXPECTED STATUS
102	050120	062716	000004		BIC	#ILR,\$GDDAT	
103	050124	112776	000045	000000	ADD	#4,(SP)	:MOVE SP TO ERROR CALL
104	050132	162716	000002		MOVB	#45,@(SP)	:WRITE NUMBER OF ERROR IN CALL
105	050136	004736			SUB	#2,(SP)	:MOVE SP TO RETURN FOR ERROR
106	050140	162716	000010		JSR	PC,@(SP)+	:REPORT THE ERROR VIA USER
107	050144	000240			SUB	#10,(SP)	:MOVE SP TO NO ERROR RETURN
108	050146				NOP		
109							
110							
111	050146	032737	000001	001350	:REPORT ANY ILF ERROR		
112	050154	001424			BIT	#ILF,RMER11	:WAS ILF SET??
113	050156	013737	001350	001142	BEQ	7\$:NO!!
114	050164	013737	001350	001140	MOV	RMER11,\$BDDAT	:RECEIVED STATUS
					MOV	RMER11,\$GDDAT	:EXPECTED STATUS

```
115 050172 042737 000001 001140      BIC      #1LF,$GDDAT
116 050200 062716 000004          ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
117 050204 112776 000046 000000      MOVB     #46,@(SP)       ;WRITE NUMBER OF ERROR IN CALL
118 050212 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
119 050216 004736          JSR      PC,@(SP)+      ;REPORT THE ERROR VIA USER
120 050220 162716 000010          SUB      #10,(SP)       ;MOVE SP TO NO ERROR RETURN
121 050224 000240
122 050226
123
124
125 050226 062716 000004          ;AUGMENT RETURN ADDRESS IF ERROR WAS FOUND
126 050232 105776 000000      ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
127 050236 001403          TSTB     @ (SP)          ;WAS ERROR FOUND??
128 050240 062716 000004          BEQ      8$              ;NO!!
129 050244 000402          ADD      #4,(SP)          ;YES - MOVE TO ERROR RETURN
130 050246 162716 000004          BR       9$              ;MOVE SP TO NO ERROR RETURN
131 050252 000240      8$: SUB      #4,(SP)
132 050254 000207      9$: NOP
133      RTS      PC
```

```

1          .SBTTL RECALIBRATE STATUS CHECK SUBROUTINE
2
3          ;THIS SUBROUTINE CHECKS THE RESULTS OF A RECALIBRATE OPERATION
4          ;USING THE STATUS STORED IN THE GET BUFFER.
5
6          ;CALL:
7
8          ;(1) JSR PC,RCLSTS ;CALL SUBROUTINE
9          BR ??? ;RETURN HERE IF NO ERROR
10         NOP ;RETURN HERE TO REPORT AN ERROR
11         ERROR ;ERROR NUMBER DEFINED BY SUB
12         JSR PC,@(SP)+ ;GO BACK TO SUB FOR MORE ERROR CHECKS
13         ??? ;RETURN HERE IF NO MORE ERRORS
14
15 050256 RCLSTS:
16
17         ;CLEAR USER'S ERROR NUMBER
18 050256 062716 000004 ADD #4,(SP)
19 050262 105076 000000 CLRB @(SP) ;CLEAR USER'S ERROR CALL
20 050266 162716 000004 SUB #4,(SP) ;MOVE SP BACK TO BRANCH
21
22
23         ;SEE IF "PAR" OR "ILF" OR "OPI" OR "IAE" IS SET
24 050272 032737 022011 001350 BIT #OPI!PAR!ILF!IAE,RMER1I
25 050300 001553 BEQ 4$ ;NONE ARE SET - GO TO NEXT CHECK
26
27         ;REPORT ANY MASSBUS CONTROL BUS PARITY ERROR, I.E.,
28         ;"PAR" = 1 AND "DPE" = 0
29 050302 032737 000010 001350 BIT #PAR,RMER1I ;WAS "PAR" SET??
30 050310 001430 BEQ 1$ ;NO!!
31 050312 032737 000010 001376 BIT #DPE,RMER2I ;WAS "DPE" SET??
32 050320 001024 BNE 1$ ;YES - NOT A REGISTER ERROR
33 050322 013737 001350 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
34 050330 042737 000010 001140 BIC #PAR,$GDDAT
35 050336 013737 001350 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
36 050344 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
37 050350 112776 000050 000000 MOVB #50,@(SP) ;WRITE ERROR NUMBER IN CALL
38 050356 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
39 050362 004736 JSR PC,@(SP)+ ;GO REPORT ERROR
40 050364 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO BRANCH
41 050370 000240 NOP
42 050372
43
44         1$:
45 050372 032737 000001 001350 ;REPORT ANY "ILF" ERROR
46 050400 001424 BIT #ILF,RMER1I ;WAS "ILF" SET??
47 050402 013737 001350 001140 BEQ 2$ ;NO!!
48 050410 042737 000001 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
49 050416 013737 001350 001142 BIC #ILF,$GDDAT
50 050424 062716 000004 MOV RMER1I,$BDDAT ;RECEIVED STATUS
51 050430 112776 000071 000000 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
52 050436 162716 000002 MOVB #71,@(SP) ;WRITE ERROR NUMBER IN CALL
53 050442 004736 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
54 050444 162716 000010 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
55 050450 000240 SUB #10,(SP) ;MOVE SP BACK TO BRANCH
56 050452
57         2$:
    
```

```

58
59
60
61 050452 032737 020000 001350
62 050460 001433
63 050462 013737 001350 001140
64 050470 042737 020000 001140
65 050476 013737 001350 001142
66 050504 062716 000004
67 050510 112776 000072 000000
68 050516 032737 010000 001346
69 050524 001403
70 050526 112776 000073 000000
71 050534 162716 000002
72 050540 004736
73 050542 162716 000010
74 050546 000240
75 050550
76
77
78 050550 032737 002000 001350
79 050556 001424
80 050560 013737 001350 001140
81 050566 042737 002000 001140
82 050574 013737 001350 001142
83 050602 062716 000004
84 050606 112776 000070 000000
85 050614 162716 000002
86 050620 004736
87 050622 162716 000010
88 050626 000240
89 050630
90
91
92 050630 032737 050200 001376
93 050636 001517
94
95
96
97
98
99 050640 032737 010000 001376
100 050646 001433
101 050650 013737 001376 001140
102 050656 042737 010000 001140
103 050664 013737 001376 001142
104 050672 062716 000004
105 050676 112776 000074 000000
106 050704 032737 000100 001346
107 050712 001403
108 050714 112776 000075 000000
109 050722 162716 000002
110 050726 004736
111 050730 162716 000010
112 050734 000240
113 050736
114

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

31$:
3$:
3$:

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

4$:

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

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

5$:
6$:

```



```

115                                     ;REPORT ANY "SKI" ERROR
116 050736 032737 040000 001376      BIT    #SKI,RMER2I      ;WAS SKI SET??
117 050744 001424                                     BEQ    7$              ;NO!!
118 050746 013737 001376 001140      MOV    RMER2I,$GDDAT   ;EXPECTED STATUS
119 050754 042737 040000 001140      BIC    #SKI,$GDDAT
120 050762 013737 001376 001142      MOV    RMER2I,$BDDAT   ;RECEIVED STATUS
121 050770 062716 000004                                     ADD    #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
122 050774 112776 000076 000000      MOV    #76,@(SP)      ;WRITE ERROR NUMBER
123 051002 162716 000002                                     SUB    #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
124 051006 004736                                     JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
125 051010 162716 000010                                     SUB    #10,(SP)       ;MOVE SP TO BRANCH
126 051014 000240                                     NOP
127 051016                                     7$:
128
129                                     ;REPORT ANY "DVC" ERROR
130 051016 032737 000200 001376      BIT    #DVC,RMER2I    ;WAS "DVC" SET??
131 051024 001424                                     BEQ    8$              ;NO!!
132 051026 013737 001376 001140      MOV    RMER2I,$GDDAT   ;EXPECTED STATUS
133 051034 042737 000200 001140      BIC    #DVC,$GDDAT
134 051042 013737 001376 001142      MOV    RMER2I,$BDDAT   ;RECEIVED STATUS
135 051050 062716 000004                                     ADD    #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
136 051054 112776 000077 000000      MOV    #77,@(SP)      ;WRITE ERROR NUMBER
137 051062 162716 000002                                     SUB    #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
138 051066 004736                                     JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
139 051070 162716 000010                                     SUB    #10,(SP)       ;MOVE SP TO USER'S BRANCH
140 051074 000240                                     NOP
141 051076                                     8$:
142
143                                     ;SEE IF "PIP" AND "OM" ARE 0, AND "ATA","MOL" AND "VV" ARE 1
144 051076 013746 001346                                     MOV    RMDSI,-(SP)    ;PUT RMDS ON STACK
145 051102 042716 047676                                     BIC    #^C<PIP!MOL!VV!OM!ATA>,(SP)
146 051106 022726 110100                                     CMP    #ATA!MOL!VV,(SP)+
147 051112 001002                                     BNE    85$
148 051114 000137 051530                                     JMP    13$
149 051120                                     85$:
150
151                                     ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
152                                     ;LINE AFTER RECALIBRATE WAS INITIATED
153 051120 032737 010000 001346      BIT    #MOL,RMDSI     ;DID MOL DROP??
154 051126 001030                                     BNE    9$              ;NO!!
155 051130 032737 020000 001350      BIT    #OPI,RMER1I    ;WAS OPI ERROR REPORTED??
156 051136 001024                                     BNE    9$              ;YES - DON'T REPORT MOL=0
157 051140 013737 001346 001140      MOV    RMDSI,$GDDAT   ;EXPECTED STATUS
158 051146 052737 010000 001140      BIS    #MOL,$GDDAT
159 051154 013737 001346 001142      MOV    RMDSI,$BDDAT   ;RECEIVED STATUS
160 051162 062716 000004                                     ADD    #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
161 051166 112776 000100 000000      MOV    #100,@(SP)     ;WRITE ERROR NUMBER
162 051174 162716 000002                                     SUB    #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
163 051200 004736                                     JSR    PC,@(SP)+      ;REPORT ERROR VIA USER
164 051202 162716 000010                                     SUB    #10,(SP)       ;MOVE SP BACK TO USER'S BRANCH
165 051206 000240                                     NOP
166 051210                                     9$:
167
168                                     ;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0
169 051210 032737 000100 001346      BIT    #VV,RMDSI      ;DID "VV" DROP??
170 051216 001030                                     BNE    10$             ;NO!!
171 051220 032737 010000 001376      BIT    #IVC,RMER2I    ;WAS THERE A IVC ERROR??
    
```

```

172 051226 001024          BNE      10$          ;YES - DONT REPORT VV=0
173 051230 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
174 051236 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
175 051244 052737 000100 001140  BIS      #VV,$GDDAT
176 051252 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
177 051256 112776 000101 000000  MOVB    #101,@(SP)   ;WRITE ERROR NUMBER IN CALL
178 051264 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
179 051270 004736          JSR      PC,@(SP)+
180 051272 162716 000010          SUB      #10,(SP)     ;MOVE SP BACK TO USER'S BRANCH
181 051276 000240          NOP
182 051300          10$:
183
184          ;REPORT AN ERROR IF ATA IS NOT SET
185 051300 032737 100000 001346  BIT      #ATA,RMDSI   ;WAS ATA SET DURING RECALIBRATE??
186 051306 001024          BNE      11$          ;YES!!
187 051310 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
188 051316 052737 100000 001140  BIS      #ATA,$GDDAT
189 051324 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
190 051332 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
191 051336 112776 000102 000000  MOVB    #102,@(SP)   ;WRITE ERROR NUMBER IN CALL
192 051344 162716 000002          SUB      #2,(SP)      ;MOVE SP TO USER'S BRANCH
193 051350 004736          JSR      PC,@(SP)+
194 051352 162716 000010          SUB      #10,(SP)
195 051356 000240          NOP
196
197 051360          11$:
198
199          ;REPORT AN ERROR IF "OM" IS NOT ZERO BECAUSE RECALIBRATE SHOULD
200          ;ALWAYS CLEAR OFFSET MODE
201 051360 032737 000001 001346  BIT      #OM,RMDSI    ;WAS "OM" RESET??
202 051366 001424          BEQ      12$          ;YES!!
203 051370 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
204 051376 042737 000001 001140  BIC      #OM,$GDDAT
205 051404 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
206 051412 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
207 051416 112776 000103 000000  MOVB    #103,@(SP)   ;WRITE ERROR NUMBER
208 051424 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
209 051430 004736          JSR      PC,@(SP)+   ;REPORT ERROR VIA USER
210 051432 162716 000010          SUB      #10,(SP)     ;MOVE SP TO USER'S BRANCH
211 051436 000240          NOP
212 051440          12$:
213
214          ;REPORT AN ERROR IF "PIP" IS STIL ON, I.E., DRIVE NOT ON
215          ;CYLINDER
216 051440 032737 020000 001346  BIT      #PIP,RMDSI   ;IS DRIVE OFF CYLINDER??
217 051446 001430          BEQ      13$          ;NO!!
218 051450 032737 040000 001376  BIT      #SKI,RMER2I  ;WAS "SKI" DETECTED??
219 051456 001024          BNE      13$          ;YES-DONT REPORT "PIP"
220 051460 013737 001346 001140  MOV      RMDSI,$GDDAT ;EXPECTED STATUS
221 051466 042737 020000 001140  BIC      #PIP,$GDDAT
222 051474 013737 001346 001142  MOV      RMDSI,$BDDAT ;RECEIVED STATUS
223 051502 062716 000004          ADD      #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
224 051506 112776 000104 000000  MOVB    #104,@(SP)   ;WRITE ERROR NUMBER
225 051514 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
226 051520 004736          JSR      PC,@(SP)+
227 051522 162716 000010          SUB      #10,(SP)     ;MOVE SP BACK TO USER'S BRANCH
228 051526 000240          NOP
    
```

```

229 051530
230
231
232 051530 032737 040006 001350 ;SEE IF "ILR" OR "RMR" OR "UNS" IS SET
233 051536 001514 BIT #ILR!RMR!UNS,RMER11
;BEQ 16$
234
235 ;REPORT AN ERROR IF "ILR" IS SET
236 051540 032737 000002 001350 BIT #ILR,RMER11 ;WAS ILR SET DURING RECALIBRATE??
237 051546 001424 BEQ 14$ ;NO!!
238 051550 013737 001350 001140 MOV RMER11,$GDDAT ;EXPECTED STATUS
239 051556 042737 000002 001140 BIC #ILR,$GDDAT
240 051564 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
241 051572 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
242 051576 112776 000105 000000 MOVB #105,@(SP) ;WRITE ERROR NUMBER IN CALL
243 051604 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
244 051610 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
245 051612 162716 000010 SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
246 051616 000240 NOP
247 051620
248
249 ;REPORT AN ERROR IF "RMR" IS SET
250 051620 032737 000004 001350 BIT #RMR,RMER11 ;WAS RMR SET??
251 051626 001424 BEQ 15$ ;NO!!
252 051630 013737 001350 001140 MOV RMER11,$GDDAT ;EXPECTED STATUS
253 051636 042737 000004 001140 BIC #RMR,$GDDAT
254 051644 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
255 051652 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
256 051656 112776 000106 000000 MOVB #106,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
257 051664 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
258 051670 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
259 051672 162716 000010 SUB #10,(SP) ;MOVE SP TO USER'S BRANCH
260 051676 000240 NOP
261 051700
262
263 ;REPORT AN ERROR IF "UNS" IS SET AND "DVC" IS 0
264 051700 032737 040000 001350 BIT #UNS,RMER11 ;WAS UNSAFE ON??
265 051706 001430 BEQ 16$ ;NO!!
266 051710 032737 000200 001376 BIT #DVC,RMER2I ;WAS THERE A DEVICE CHECK??
267 051716 001024 BNE 16$ ;YES - DON'T REPORT UNSAFE
268 051720 013737 001350 001140 MOV RMER11,$GDDAT ;EXPECTED STATUS
269 051726 042737 040000 001140 BIC #UNS,$GDDAT
270 051734 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
271 051742 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
272 051746 112776 000107 000000 MOVB #107,@(SP) ;WRITE ERROR NUMBER
273 051754 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
274 051760 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
275 051762 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO USER'S BRANCH
276 051766 000240 NOP
277 051770
278
279 ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
280 051770 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
281 051774 105776 000000 TSTB @(SP) ;WAS AN ERROR REPORTED??
282 052000 001403 BEQ 17$ ;NO!!
283 052002 062716 000004 ADD #4,(SP) ;YES - AUGMENT SP RETURN
284 052006 000402 BR 18$
285 052010 162716 000004 17$: SUB #4,(SP) ;NO ERROR - RETURN SP TO BRANCH
    
```

286 052014 000240
287 052016 000207
288

18\$: NOP
RTS PC

;STATUS CECK IS COMPLETE

```

1          .SBTTL  DRIVE CLEAR STATUS CHECK SUBROUTINE
2
3          :
4          :   BR      ???          RETURN HERE IF NO ERROR
5          :   NOP
6          :   ERROR          RETURN HERE TO REPORT AN ERROR
7          :   JSR      PC,@(SP)+  ERROR NUMBER DEFINED BY SUB
8          :   ???          GO BACK TO SUB FOR MORE ERROR CHECKS
9          :   ???          RETURN HERE IF NO MORE ERRORS
10
11         DRVSTS:
12         ;CLEAR USER'S ERROR CALL
13         ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
14         CLRB    @(SP)          ;CLEAR ERROR CALL
15         SUB      #4,(SP)          ;MOVE SP TO USER'S BRANCH
16         ;REPORT ERROR IF RMCS1 NOT INITIALIZED
17         4$:  MOV      RMCS1I,$BDDAT ;CHECK RMCS1
18         BIC     #^C<DVA!FNCMSK>,$BDDAT ;CLEAR DONT CARES
19         MOV     #DVA!DRVCLR,$GDDAT ;EXPECT DVA
20         CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
21         BEQ     6$              ;BRANCH IF EQUAL
22         ADD     #4,(SP)          ;MOVE SP TO ERROR CALL
23         MOV     #141,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
24         SUB     #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
25         JSR     PC,@(SP)+      ;REPORT THE ERROR VIA USER
26         SUB     #10,(SP)        ;MOVE SP TO NO ERROR RETURN
27         NOP
28         ;REPORT ERROR IF RMDS NOT INITIALIZED
29         5$:  MOV      RMDSI,$BDDAT ;CHECK RMDS
30         BIC     #PGM!OM!VV!PIP,$BDDAT ;CLEAR DONT CARES
31         MOV     #MOL!DPR!DRY,$GDDAT ;EXPECT DRY & DPR
32         CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED & RECEIVED
33         BEQ     6$              ;BRANCH IF EQUAL
34         ADD     #4,(SP)          ;MOVE SP TO ERROR CALL
35         MOV     #142,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
36         SUB     #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
37         JSR     PC,@(SP)+      ;REPORT THE ERROR VIA USER
38         SUB     #10,(SP)        ;MOVE SP TO NO ERROR RETURN
39         NOP
40         ;REPORT ERROR IF RMER1 NOT INITIALIZED
41         6$:  CLR      $GDDAT      ;EXPECT 0'S
42         MOV     RMER1I,$BDDAT ;CHECK RMER1
43         BEQ     8$              ;BRANCH IF EQUAL
44         ADD     #4,(SP)          ;MOVE SP TO ERROR CALL
45         MOV     #143,@(SP)      ;WRITE NUMBER OF ERROR IN CALL
46         SUB     #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
47         JSR     PC,@(SP)+      ;REPORT THE ERROR VIA USER
48         SUB     #10,(SP)        ;MOVE SP TO NO ERROR RETURN
49         NOP
50         ;REPORT ERROR IF ATA NOT INITIALIZED
51         8$:  MOV      RMASI,$BDDAT ;CHECK ATTENTION BIT
52         MOV     R1,-(SP)         ;;PUSH R1 ON STACK
53         MOV     R2,-(SP)         ;;PUSH R2 ON STACK
54         MOV     TSTQUE,R1
55         MOV     1(R1),R2
56         BIC     #^CATNMSK,R2
57         COM     R2
58         BIC     R2,$BDDAT
    
```

```

58 052272 012602          MOV      (SP)+,R2          ;;POP STACK INTO R2
59 052274 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
60 052276 005737 001142    TST      $BDDAT           ;IS ATTENTION CLEARED??
61 052302 001413          BEQ      9$               ;BRANCH IF ATTENTION CLEARED
62 052304 062716 000004    ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
63 052310 112776 000144 000000    MOVVB   #144,@(SP)        ;WRITE NUMBER OF ERROR IN CALL
64 052316 162716 000002    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
65 052322 004736          JSR      PC,@(SP)+        ;REPORT THE ERROR VIA USER
66 052324 162716 000010    SUB      #10,(SP)         ;MOVE SP TO NO ERROR RETURN
67 052330 000240          NOP
68                                ;REPORT ERROR IF RMMR1 NOT INITIALIZED
69 052332 013737 001360 001142 9$: MOV      RMMR1I,$BDDAT     ;CHECK RMMR
70 052340 042737 000046 001142    BIC      #WC!LS!LST,$BDDAT ;CLEAR DONT CARES
71 052346 012737 000010 001140    MOV      #MWD,$GDDAT      ;EXPECT WRITE DATA ON
72 052354 023737 001140 001142    CMP      $GDDAT,$BDDAT    ;COMPARE EXPECTED AND RECEIVED
73 052362 001413          BEQ      11$             ;BRANCH IF ZERO
74 052364 062716 000004    ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
75 052370 112776 000145 000000    MOVVB   #145,@(SP)        ;WRITE NUMBER OF ERROR IN CALL
76 052376 162716 000002    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
77 052402 004736          JSR      PC,@(SP)+        ;REPORT THE ERROR VIA USER
78 052404 162716 000010    SUB      #10,(SP)         ;MOVE SP TO NO ERROR RETURN
79 052410 000240          NOP
80                                ;REPORT ERROR IF RMMR2 NOT INITIALIZED
81 052412 013737 001374 001142 11$: MOV      RMMR2I,$BDDAT     ;CHECK RMMR2
82 052420 042737 140000 001142    BIC      #RQA!RQB,$BDDAT  ;CLEAR REQA, REQB
83 052426 012737 011777 001140    MOV      #TST!1777,$GDDAT ;EXPECT TEST BIT ON
84 052434 023737 001140 001142    CMP      $GDDAT,$BDDAT    ;COMPARE EXPECTED & RECEIVED
85 052442 001413          BEQ      15$             ;BRANCH IF EQUAL
86 052444 062716 000004    ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
87 052450 112776 000146 000000    MOVVB   #146,@(SP)        ;WRITE NUMBER OF ERROR IN CALL
88 052456 162716 000002    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
89 052462 004736          JSR      PC,@(SP)+        ;REPORT THE ERROR VIA USER
90 052464 162716 000010    SUB      #10,(SP)         ;MOVE SP TO NO ERROR RETURN
91 052470 000240          NOP
92 052472 005037 001140 15$: CLR      $GDDAT           ;EXPECT ZEROS
93                                ;REPORT ERROR IF RMEC2 NOT RESET
94 052476 013737 001402 001142    MOV      RMEC2I,$BDDAT    ;CHECK RMEC2
95 052504 001413          BEQ      17$             ;BRANCH IF 0
96 052506 062716 000004    ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
97 052512 112776 000150 000000    MOVVB   #150,@(SP)        ;WRITE NUMBER OF ERROR IN CALL
98 052520 162716 000002    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
99 052524 004736          JSR      PC,@(SP)+        ;REPORT THE ERROR VIA USER
100 052526 162716 000010    SUB      #10,(SP)         ;MOVE SP TO NO ERROR RETURN
101 052532 000240          NOP
102                                ;REPORT ERROR IF RMER2 NOT RESET
103 052534 013737 001376 001142 17$: MOV      RMER2I,$BDDAT    ;CHECK RMER2
104 052542 001413          BEQ      18$             ;BRANCH IF NO ERROR
105 052544 062716 000004    ADD      #4,(SP)          ;MOVE SP TO ERROR CALL
106 052550 112776 000147 000000    MOVVB   #147,@(SP)        ;WRITE NUMBER OF ERROR IN CALL
107 052556 162716 000002    SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
108 052562 004736          JSR      PC,@(SP)+        ;REPORT THE ERROR VIA USER
109 052564 162716 000010    SUB      #10,(SP)         ;MOVE SP TO NO ERROR RETURN
110 052570 000240          NOP
111 052572          18$:
112
113 052572          19$:
114

```

```
115  
116 052572 062716 000004  
117 052576 105776 000000  
118 052602 001403  
119 052604 062716 000004  
120 052610 000402  
121 052612 162716 000004  
122 052616 000240  
123 052620 000207  
124  
  
;AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND  
ADD #4,(SP) ;MOVE SP TO ERROR CALL  
TSTB @ (SP) ;WAS AN ERROR DETECTED??  
BEQ 21$ ;NO!!  
ADD #4,(SP) ;YES - MOVE SP TO ERROR RETURN  
BR 23$  
21$: SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN  
23$: NOP  
RTS PC ;RETURN TO USER
```

```

1      .SBTTL DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE VERIFIES THE RESULTS OF ALL DATA TRANSFER COMMANDS
4      ;USING STATUS STORED IN THE GET BUFFER AND TEST PARAMETERS
5      ;STORED IN THE PUT BUFFER.  ERRORS ARE REPORTED BY WRITING
6      ;THE ERROR NUMBER IN THE USERS ERROR CALL.
7
8      ;USER'S SUBROUTINE CALL:
9      ;(1) JSR PC,DTASTS
10     ;(2) BR ?? RETURN HERE IF NO DATA ERRORS
11     ;(3) NOP RETURN HERE TO REPORT AN ERROR
12     ;(4) ERROR SUB WRITES ERROR NUMBER
13     ;(5) JSR PC,@(SP)+ USER RETURNS FOR MORE CHECKS
14     ;(6) ?? SUB RETURNS HERE AFTER ALL
15     ; ERRORS ARE REPORTED
16
17 052622 DTASTS:
18
19     ;CLEAR USER'S ERROR CALL AND ERROR FLAGS
20 052622 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
21 052626 105076 000000 CLRB @(SP) ;CLEAR LOW ORDER BYTE OF TRAP
22 052632 162716 000004 SUB #4,(SP) ;RESTORE SP TO NO ERROR
23 052636 005037 056216 CLR 500$ ;CLEAR ERROR FLAGS
24
25     ;REPORT ANY CONTROL BUS PARITY ERROR WHILE READING REMOTE REGISTERS,
26     ;I.E., MCPE = 1
27 052642 032737 020000 001334 BIT #MCPE,RMCS11 ;WAS THERE A PARITY ERROR??
28 052650 001422 BEQ 10$ ;NO!!
29 052652 013737 001334 001140 MOV RMCS11,$GDDAT ;EXPECTED STATUS
30 052660 042737 020000 001140 BIC #MCPE,$GDDAT
31 052666 013737 001334 001142 MOV RMCS11,$BDDAT ;RECEIVED STATUS
32 052674 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
33 052700 112776 000013 000000 MOVB #13,@(SP) ;WRITE ERROR NUMBER
34 052706 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
35 052712 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
36 052714 000466 BR 30$
37 052716
38
39     ;REPORT ANY CONTROL BUS PARITY ERROR WHILE WRITING REMOTE REGISTERS,
40     ;I.E., PAR = 1 AND DPE = 0
41 052716 032737 000010 001350 BIT #PAR,RMER11 ;WAS THERE A PARITY ERROR??
42 052724 001435 BEQ 20$ ;NO!!
43 052726 032737 000010 001376 BIT #DPE,RMER21 ;DATA PARITY ERROR ?
44 052734 001031 BNE 20$ ;YES!!
45 052736 013737 001350 001140 MOV RMER11,$GDDAT ;EXPECTED STATUS
46 052744 042737 000010 001140 BIC #PAR,$GDDAT
47 052752 013737 001350 001142 MOV RMER11,$BDDAT ;RECEIVED STATUS
48 052760 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
49 052764 112776 000050 000000 MOVB #50,@(SP) ;WRITE ERROR NUMBER
50 052772 032737 001000 001344 BIT #MXF,RMCS21 ;DID MXF GET SET??
51 053000 001003 BNE 15$ ;YES!!
52 053002 112776 000274 000000 MOVB #274,@(SP) ;NO - CHANGE ERROR NUMBER
53 053010 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
54 053014 004736 JSR PC,@(SP)+ ;REPORT ERROR AND RETURN
55 053016 000425 BR 30$
56
57     ;LOOK FOR ANY ERRORS WHICH MAY HAVE OCCURRED DURING COMMAND INITIATION OR
    
```



```

58                                     ;MECHANICAL POSITIONING
59
60                                     ;FIRST TEST MXF WHICH WOULD INDICATE COMPOSITE ERROR SET WHEN FUNCTION
61                                     ;CODE AND GO BIT WERE LOADED
62 053020                               20$:
63 053020 032737 001000 001344         BIT    #MXF,RMCS2I    ;WAS 'MISSED TRANSFER' SET??
64 053026 001425                               BEQ    40$          ;NO!!
65 053030 013737 001344 001140         MOV    RMCS2I,$GDDAT ;EXPECTED STATUS
66 053036 042737 001000 001140         BIC    #MXF,$GDDAT
67 053044 013737 001344 001142         MOV    RMCS2I,$BDDAT ;RECEIVED STATUS
68 053052 062716 000004                               ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
69 053056 112776 000275 000000         MOVB  #275,@(SP)   ;WRITE ERROR NUMBER
70 053064 162716 000002                               SUB    #2,(SP)     ;MOVE SP TO RETURN IF ERROR
71 053070 004736                               JSR    PC,@(SP)+   ;REPORT ERROR AND RETURN
72 053072
73
74                                     ;RESTORE SP TO NO ERROR RETURN AND BYPASS FURHTER STATUS CHECKING
75 053072 162716 000010                               SUB    #10,(SP)   ;MOVE SP TO NO ERROR
76 053076 000137 056170                               JMP    380$       ;SKIP TO END OF SUB
77
78 053102                               40$:
79
80                                     ;REPORT AN ERROR IF 'OPI' ERROR OCCURRED DUE TO 'MOL' = 0, OR IF 'OPI'
81                                     ;AND 'MOL' ARE SET, BUT 'VV' IS RESET, INDICATING AN INTERMITTENT
82                                     ;'MOL'
83 053102 032737 020000 001350         BIT    #OPI,RMER1I ;IS 'OPI' SET??
84 053110 001447                               BEQ    60$          ;NO!!
85 053112 013737 001350 001140         MOV    RMER1I,$GDDAT ;EXPECTED STATUS
86 053120 042737 020000 001140         BIC    #OPI,$GDDAT
87 053126 013737 001350 001142         MOV    RMER1I,$BDDAT ;RECEIVED STATUS
88 053134 032737 010000 001346         BIT    #MOL,RMDSI  ;WAS MEDIUM OFF LINE??
89 053142 001404                               BEQ    45$          ;YES!!
90 053144 032737 000100 001346         BIT    #VV,RMDSI   ;WAS 'MOL' INTERMITTENT??
91 053152 001013                               BNE    50$          ;NO!!
92 053154 062716 000004                               45$: ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
93 053160 112776 000276 000000         MOVB  #276,@(SP)   ;WRITE ERROR NUMBER IN CALL
94 053166 162716 000002                               SUB    #2,(SP)     ;MOVE SP TO RETURN IF ERROR
95 053172 004736                               JSR    PC,@(SP)+   ;REPORT ERROR AND RETURN
96 053174 162716 000010                               SUB    #10,(SP)   ;RESTORE SP TO NO ERROR
97 053200 000413
98 053202                               50$:
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 053202 062716 000004                               ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
103 053206 112776 000277 000000         MOVB  #277,@(SP)   ;WRITE ERROR NUMBER IN CALL
104 053214 162716 000002                               SUB    #2,(SP)     ;MOVE SP TO RETURN IF ERROR
105 053220 004736                               JSR    PC,@(SP)+   ;REPORT ERROR AND RETURN
106 053222 162716 000010                               SUB    #10,(SP)   ;RESTORE SP TO NO ERROR
107 053226 000240
108 053230                               60$:
109
110                                     ;LOOK FOR 'IVC' ERROR DURING COMMAND INITIATION
111 053230 032737 010000 001376         BIT    #IVC,RMER2I ;WAS THERE AN 'IVC' ERROR??
112 053236 001432                               BEQ    70$          ;NO!!
113                                     ;REPORT 'IVC' ERROR DUE TO 'VV' = 0, OR REPORT ERRONEOUS 'IVC' ERROR
114 053240 013737 001376 001140         MOV    RMER2I,$GDDAT ;EXPECTED STATUS
    
```

```

115 053246 042737 010000 001140      BIC      #IVC,$GDDAT
116 053254 013737 001376 001142      MOV      RMER21,$BDDAT      ;RECEIVED STATUS
117 053262 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR
118 053266 112776 000300 000000      MOVVB   #300,@(SP)        ;WRITE ERROR NUMBER IN CALL
119 053274 032737 000100 001346      BIT      #VV,RMDSI        ;WAS VOLUME VALID??
120 053302 001403                BEQ      65$              ;NO!!
121 053304 112776 000301 000000      MOVVB   #301,@(SP)        ;CHANGE ERROR NUMBER
122 053312 162716 000002      65$:    SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
123 053316 004736                JSR      PC,@(SP)+        ;REPORT 'IVC' ERROR AND RETURN
124 053320 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
125 053324                70$:
126
127                ;SEE IF 'ILF' OR 'RMR' IS SET
128 053324 032737 000007 001350      BIT      #ILR!ILF!RMR, RMER11
129 053332 001510                BEQ      100$            ;NO ERRORS DETECTED
130                ;REPORT AN ERROR IF 'ILR' IS SET
131 053334 032737 000002 001350      BIT      #ILR,RMER11      ;WAS 'ILR' DETECTED??
132 053342 001424                BEQ      80$            ;NO!!
133 053344 013737 001350 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
134 053352 042737 000002 001140      BIC      #ILR,$GDDAT
135 053360 013737 001350 001142      MOV      RMER11,$BDDAT    ;RECEIVED STATUS
136 053366 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
137 053372 112776 000302 000000      MOVVB   #302,@(SP)        ;WRITE ERROR NUMBER IN CALL
138 053400 162716 000002                SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
139 053404 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
140 053406 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
141 053412 000240                NOP
142 053414                80$:
143
144                ;REPORT AN ERROR IF 'ILF' IS SET
145 053414 032737 000001 001350      BIT      #ILF,RMER11      ;WAS 'ILF' DETECTED??
146 053422 001424                BEQ      90$            ;NO!!
147 053424 013737 001350 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
148 053432 042737 000001 001140      BIC      #ILF,$GDDAT
149 053440 013737 001350 001142      MOV      RMER11,$BDDAT    ;RECEIVED STATUS
150 053446 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
151 053452 112776 000303 000000      MOVVB   #303,@(SP)        ;WRITE ERROR NUMBER IN CALL
152 053460 162716 000002                SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
153 053464 004736                JSR      PC,@(SP)+        ;REPORT ERORR AND RETURN
154 053466 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
155 053472 000240                NOP
156 053474                90$:
157
158                ;REPORT AN ERROR IF 'RMR' IS SET
159 053474 032737 000004 001350      BIT      #RMR,RMER11      ;WAS 'RMR' DETECTED??
160 053502 001424                BEQ      100$           ;NO!!
161 053504 013737 001350 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
162 053512 042737 000004 001140      BIC      #RMR,$GDDAT
163 053520 013737 001350 001142      MOV      RMER11,$BDDAT    ;RECEIVED STATUS
164 053526 062716 000004                ADD      #4,(SP)           ;MOVE SP TO USER'S ERROR CALL
165 053532 112776 000304 000000      MOVVB   #304,@(SP)        ;WRITE ERROR NUMBER IN CALL
166 053540 162716 000002                SUB      #2,(SP)           ;MOVE SP TO RETURN IF ERROR
167 053544 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
168 053546 162716 000010                SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
169 053552 000240                NOP
170 053554                100$:
171                ;DETERMINE WHETHER OR NOT 'IAE' SHOULD BE SET AND CHECK FOR ERROR
    
```

```

172 053554 012737 002000 001140      MOV      #IAE,$GDDAT      ;SETUP FOR "IAE" = 1
173 053562 052737 040000 056216      BIS      #SKI,500$       ;SETUP FOR "SKI" = 1
174 053570 023727 001444 001466      CMP      RMDCO,#822.     ;GREATER THAN LAST CYLINDER ?
175 053576 101025                BHI      110$           ;YES - CYLINDER IS INVALID
176 053600 042737 040000 056216      BIC      #SKI,500$       ;RESET SKI FLAG
177
178 053606 123737 001417 001333      CMPB     RMDAO+1,LSTRK+1 ;GREATER THAN LAST TRACK ?
179 053614 101016                BHI      110$           ;YES - TRACK IS INVALID
180
181 053616 123727 001416 000035      CMPB     RMDAO,#29.     ;IS SECTOR > 29. ?
182 053624 101410                BLOS     105$           ;NO
183 053626 032737 010000 001442      BIT      #FMT16,RMOFO   ;18 BIT FORMAT ?
184 053634 001406                BEQ      110$           ;YES - SECTOR IS INVALID FOR 18 BIT MODE
185 053636 123727 001416 000037      CMPB     RMDAO,#31.     ;IS SECTOR > 31. ?
186 053644 101002                BHI      110$           ;YES - SECTOR IS INVALID
187 053646 005037 001140          105$:    CLR      $GDDAT      ;"IAE" SHOULD = 0
188
189 053652 013737 001350 001142      110$:    MOV      RMER1I,$BDDAT   ;GET RECEIVED STATUS
190 053660 042737 175777 001142      BIC      #^CIAE,$BDDAT
191 053666 023737 001140 001142      CMP      $GDDAT,$BDDAT  ;IS "IAE" STATUS OK??
192 053674 001004                BNE      115$           ;NO!!
193 053676 042737 040000 056216      BIC      #SKI,500$       ;IAE OK - SKI SHOULD BE 0
194 053704 000412                BR       120$
195 053706 062716 000004          115$:    ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
196 053712 112776 000305 000000      MOV      #305,@(SP)     ;WRITE ERROR NUMBER
197 053720 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
198 053724 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
199 053726 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
200 053732          120$:
201
202          ;REPORT AN ERROR IF "SKI" IS SET AND "IAE" STATUS WAS OK
203 053732 013737 001376 001142      MOV      RMER2I,$BDDAT  ;RECEIVED STATUS
204 053740 042737 137777 001142      BIC      #^CSKI,$BDDAT
205 053746 013737 056216 001140      MOV      500$,$GDDAT    ;EXPECTED STATUS
206 053754 042737 137777 001140      BIC      #^CSKI,$GDDAT
207 053762 032737 040000 001376      BIT      #SKI,RMER2I    ;WAS "SKI" SET??
208 053770 001417                BEQ      140$           ;NO!!
209 053772 032737 040000 056216      BIT      #SKI,500$      ;WAS SKI CAUSED BY IAE = 0??
210 054000 001032                BNE      150$           ;YES - DON'T REPORT SKI
211 054002 062716 000004          ADD      #4,(SP)         ;MOVE SP TO USERS ERROR CALL
212 054006 112776 000306 000000      MOV      #306,@(SP)     ;WRITE ERROR NUMBER
213 054014 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
214 054020 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
215 054022 162716 000010          SUB      #10,(SP)        ;MOVE SP TO NO ERROR
216 054026 000417                BR       150$
217
218 054030          140$:
219
220          ;REPORT AN ERROR IF SKI = 0 AND IAE WAS NOT DETECTED
221 054030 032737 040000 056216      BIT      #SKI,500$      ;SHOULD SKI BE SET??
222 054036 001413                BEQ      150$           ;NO!!
223 054040 062716 000004          ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
224 054044 112776 000307 000000      MOV      #307,@(SP)     ;WRITE ERROR NUMBER IN CALL
225 054052 162716 000002          SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
226 054056 004736          JSR      PC,@(SP)+      ;REPORT ERROR AND RETURN
227 054060 162716 000010          SUB      #10,(SP)        ;RESTORE SP TO NO ERROR
228 054064 000240                NOP

```

```

229 054066          150$:
230
231                ;LOOK FOR "LSC" OR "LBC" OR "DVC" IN ERROR REGISTER #2
232 054066 032737 006200 001376      BIT      #LSC!LBC!DVC,RMER2I
233 054074 001512                BEQ      180$          ;NO ERRORS SET
234
235                ;REPORT ANY DEVICE FAULT, I.E., "DVC" = 1
236 054076 032737 000200 001376      BIT      #DVC,RMER2I      ;IS "DVC" = 1??
237 054104 001424                BEQ      160$          ;NO!!
238 054106 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
239 054114 042737 000200 001140      BIC      #DVC,$GDDAT
240 054122 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
241 054130 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USERS ERROR
242 054134 112776 000310 000000      MOV      #310,@(SP)      ;WRITE ERROR NUMBER IN CALL
243 054142 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
244 054146 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
245 054150 162716 000010                SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
246 054154 000240                NOP
247 054156          160$:
248
249                ;REPORT LOSS OF BIT CLOCK, I.E.: "LBC" = 1, IF "MOL" = 1
250 054156 032737 002000 001376      BIT      #LBC,RMER2I      ;IS LBC SET??
251 054164 001430                BEQ      170$          ;NO!!
252 054166 032737 010000 001346      BIT      #MOL,RMDSI      ;WAS LBC ERROR BY MOL = 0
253 054174 001424                BEQ      170$          ;YES!!
254 054176 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
255 054204 042737 002000 001140      BIC      #LBC,$GDDAT
256 054212 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
257 054220 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
258 054224 112776 000311 000000      MOV      #311,@(SP)      ;WRITE ERROR NUMBER IN CALL
259 054232 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
260 054236 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
261 054240 162716 000010                SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
262 054244 000240                NOP
263 054246          170$:
264
265                ;REPORT LOS OF SYSTEM CLOCK, I.E., "LSC" = 1
266 054246 032737 004000 001376      BIT      #LSC,RMER2I      ;IS "LSC" = 1??
267 054254 001422                BEQ      180$          ;NO!!
268 054256 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
269 054264 042737 004000 001140      BIC      #LSC,$GDDAT
270 054272 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
271 054300 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
272 054304 112776 000312 000000      MOV      #312,@(SP)      ;WRITE ERROR NUMBER
273 054312 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
274 054314 162716 000010                SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
275 054320 000240                NOP
276 054322          180$:
277
278                ;LOOK FOR "UNS" OR "DTE" OR "WLE" IN ERROR REGISTER #1
279 054322 032737 054000 001350      BIT      #UNS!DTE!WLE,RMER1I
280 054330 001527                BEQ      220$          ;NO BITS SET
281                ;REPORT "UNS" ERROR IF "DVC" = 0
282 054332 032737 040000 001350      BIT      #UNS,RMER1I      ;IS "UNS" SET??
283 054340 001427                BEQ      190$          ;NO!!
284 054342 032737 000200 001376      BIT      #DVC,RMER2I      ;WAS "UNS" CAUSED BY "DVC"??
285 054350 001023                BNE     190$          ;YES!!
  
```

```
286 054352 013737 001350 001140      MOV      RMER11,$GDDAT      ;EXPECTED STATUS
287 054360 042737 040000 001140      BIC      #UNS,$GDDAT
288 054366 013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
289 054374 062716 000004              ADD      #4,(SP)             ;MOVE SP TO USERS ERROR CALL
290 054400 112776 000313 000000      MOV      #313,@(SP)         ;WRITE ERROR NUMBER
291 054406 162716 000002              SUB      #2,(SP)             ;MOVE SP TO RETURN IF ERROR
292 054412 004736 000000              JSR      PC,@(SP)+          ;REPORT ERROR AND RETURN
293 054414 162716 000010              SUB      #10,(SP)           ;RESTORE SP TO NO ERROR
294 054420
295
296
297 054420 032737 010000 001350      ;REPORT ANY DRIVE TIMING ERROR, I.E., 'DTE' = 1
298 054426 001423              BIT      #DTE,RMER11        ;IS DTE SET??
299 054430 013737 001350 001140      BEQ      200$               ;NO!!
300 054436 042737 010000 001140      MOV      RMER11,$GDDAT      ;EXPECTED STATUS
301 054444 013737 001350 001142      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
302 054452 062716 000004              ADD      #4,(SP)             ;MOVE SP TO USER'S ERROR CALL
303 054456 112776 000314 000000      MOV      #314,@(SP)         ;WRITE ERROR NUMBER IN CALL
304 054464 162716 000002              SUB      #2,(SP)             ;MOVE SP TO RETURN IF ERROR
305 054470 004736 000000              JSR      PC,@(SP)+          ;REPORT ERROR AND RETURN
306 054472 162716 000010              SUB      #10,(SP)           ;MOVE SP TO NO ERROR
307 054476
308
309
310
311 054476 032737 004000 001350      ;REPORT AN ERROR IF WRITE LOCK ERROR IS SET. SEE IF DRIVE IS NOT
312 054504 001441              ;WRITE PROTECTED, OR IF FUNCTION WAS NOT A WRITE
313 054506 013737 001350 001142      BIT      #WLE,RMER11        ;WAS 'WLE' SET??
314 054514 013737 001350 001140      BEQ      220$               ;NO!!
315 054522 052737 004000 001140      MOV      RMER11,$BDDAT      ;RECEIVED STATUS
316 054530 062716 000004              MOV      RMER11,$GDDAT      ;EXPECTED STATUS
317 054534 112776 000315 000000      BIS      #WLE,$GDDAT
318 054542 032737 004000 001346      ADD      #4,(SP)             ;MOVE SP TO USERS ERROR CALL
319 054550 001404              MOV      #315,@(SP)         ;WRITE ERROR NUMBER IN CALL
320 054552 032737 000010 001410      BIT      #WRL,RMDSI        ;WAS DRIVE WRITE PROTECTED??
321 054560 001406              BEQ      205$               ;NO!!
322 054562 112776 000316 000000      BIT      #BIT3,RMCS10       ;WAS COMMAND A WRITE??
323 054570 042737 004000 001140      BEQ      210$               ;YES!!
324 054576 162716 000002              MOV      #316,@(SP)         ;CHANGE ERROR NUMBER
325 054602 004736 000000      BIC      #WLE,$GDDAT
326 054604 162716 000010      SUB      #2,(SP)             ;MOVE SP TO RETURN IF ERROR
327
328 054610              JSR      PC,@(SP)+          ;REPORT ERROR AND RETURN
329
330
331 054610 062716 000004              SUB      #10,(SP)           ;MOVE SP TO NO ERROR
332 054614 105776 000000              ;OMIT DATA ERROR CHECKS IF ANY PREVIOUS ERRORS HAVE BEEN DETECTED
333 054620 001404              ADD      #4,(SP)             ;MOVE SP TO USER'S ERROR
334 054622 162716 000004              TSTB     @(SP)              ;WAS ERROR DETECTED??
335 054626 000137 055630              BEQ      225$               ;NO - DO DATA CHECKS
336 054632 162716 000004              SUB      #4,(SP)             ;RESTORE SP
337
338
339
340 054636 013737 001410 056220      JMP      340$               ;SKIP DATA CHECKS
341 054644 042737 177700 056220      ;CHECK HEADER ERRORS IF FUNCTION WAS NOT WRITE HEADER AND DATA, AND
342 054652 022737 000063 056220      ;IF HEADER COMPARE IS NOT INHIBITED
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
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
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
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
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
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
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```

343 054660 001512          BEQ      250$          ;YES - SKIP HEADER CHECKS
344 054662 032737 002000 001366  BIT      #HCI,RMOFI    ;WAS HCI SET??
345 054670 001106          BNE      250$          ;YES - SKIP HEADER CHECKS
346
347
348 054672 032737 000620 001350  ;SEE IF ANY HEADER ERRORS ARE SET, I.E., 'FER' OR 'HCRC' OR 'HCE'
349 054700 001533          BIT      #HCRC!FER!HCE,RMER1I
350          BEQ      270$          ;NO ERRORS SET
351
352 054702 032737 000400 001350  ;REPORT HEADER CRC ERROR IF SET
353 054710 001422          BIT      #HCRC,RMER1I  ;WAS HCRC SET??
354 054712 013737 001350 001140  BEQ      230$          ;NO!!
355 054720 042737 000400 001140  MOV      RMER1I,$GDDAT ;EXPECTED STATUS
356 054726 013737 001350 001142  BIC      #HCRC,$GDDAT
357 054734 062716 000004          MOV      RMER1I,$BDDAT ;RECEIVED STATUS
358 054740 112776 000317 000000  ADD      #4,(SP)       ;MOVE SP TO USERS ERROR
359 054746 162716 000002          MOV      #317,@(SP)    ;WRITE ERROR NUMBER
360 054752 004736          SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
361 054754 000501          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
362 054756
363
364          230$:
365 054756 032737 000020 001350  ;REPORT FORMAT ERROR IF SET
366 054764 001422          BIT      #FER,RMER1I  ;WAS 'FER' SET??
367 054766 013737 001350 001140  BEQ      240$          ;NO!!
368 054774 042737 000020 001140  MOV      RMER1I,$GDDAT ;EXPECTED STATUS
369 055002 013737 001350 001142  BIC      #FER,$GDDAT
370 055010 062716 000004          MOV      RMER1I,$BDDAT ;RECEIVED STATUS
371 055014 112776 000320 000000  ADD      #4,(SP)       ;MOVE SP TO USERS ERROR
372 055022 162716 000002          MOV      #320,@(SP)   ;WRITE ERROR NUMBER
373 055026 004736          SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
374 055030 000453          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
375 055032
376          240$:
377
378 055032 032737 000200 001350  ;REPORT HEADER COMPARE ERROR IF SET
379 055040 001453          BIT      #HCE,RMER1I  ;WAS 'HCE' SET??
380 055042 013737 001350 001140  BEQ      270$          ;NO!!
381 055050 042737 000200 001140  MOV      RMER1I,$GDDAT ;EXPECTED STATUS
382 055056 013737 001350 001142  BIC      #HCE,$GDDAT
383 055064 062716 000004          MOV      RMER1I,$BDDAT ;RECEIVED STATUS
384 055070 112776 000321 000000  ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR
385 055076 162716 000002          MOV      #321,@(SP)   ;WRITE ERROR NUMBER
386 055102 004736          SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
387 055104 000425          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN
388          BR      260$
389
390          ;THERE SHOULD BE NO HEADER ERRORS BECAUSE
391          ;.COMMAND WAS WRITE HEADER AND DATA, OR
392          ;.HEADER COMPARE INHIBIT WAS SET
393          250$:
394 055106 032737 000620 001350  BIT      #HCE!FER!HCRC,RMER1I
395 055114 001425          BEQ      270$          ;NO ERRORS WERE SET
396 055116 013737 001350 001140  MOV      RMER1I,$GDDAT ;EXPECTED STATUS
397 055124 042737 000620 001140  BIC      #HCE!FER!HCRC,$GDDAT
398 055132 013737 001350 001142  MOV      RMER1I,$BDDAT ;RECEIVED STATUS
399 055140 062716 000004          ADD      #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
400 055144 112776 000322 000000  MOV      #322,@(SP)   ;WRITE ERROR NUMBER
401 055152 162716 000002          SUB      #2,(SP)       ;MOVE SP TO RETURN IF ERROR
402 055156 004736          JSR      PC,@(SP)+    ;REPORT ERROR AND RETURN

```

```

400 055160 162716 000010      260$: SUB #10,(SP)      ;MOVE SP TO NO ERROR
401 055164 000137 055630      JMP 340$           ;OMIT FURTHER DATA CHECKS
402
403 055170      270$:
404
405      ;IF COMMAND WAS A WRITE COMMAND, GO DO WRITE ERROR CHECKS, OTHERWISE
406      ;DO READ ERROR CHECKS
407 055170 032737 000010 056220      BIT #BIT3,510$    ;WAS THIS A WRITE COMMAND?
408 055176 001002      BNE 275$         ;NO!!
409 055200 000137 055416      JMP 310$         ;GO DO WRITE STATUS CHECK
410 055204
411
412      ;REPORT DATA CHECK IF SET
413 055204 032737 100000 001350      BIT #DCK,RMER11  ;DATA CHECK ERROR??
414 055212 001450      BEQ 290$         ;NO!!
415 055214 013737 001350 001140      MOV RMER11,$GDDAT ;EXPECTED STATUS
416 055222 042737 100000 001140      BIC #DCK,$GDDAT
417 055230 013737 001350 001142      MOV RMER11,$BDDAT ;RECEIVED STATUS
418 055236 062716 000004      ADD #4,(SP)      ;MOVE SP TO USER'S ERROR
419 055242 112776 000323 000000      MOVB #323,@(SP)  ;WRITE ERROR NUMBER
420 055250 032737 004000 001366      BIT #EC1,RMOFI   ;WAS ECC CORRECTION DISABLED??
421 055256 001021      BNE 280$         ;YES!!
422 055260 112776 000324 000000      MOVB #324,@(SP)  ;CHANGE TO RECOVERABLE ERROR
423 055266 032737 000100 001350      BIT #ECH,RMER11  ;IS ERROR RECOVERABLE??
424 055274 001007      BNE 276$         ;NO !!
425
426 055276 032737 000020 056220      ;DO NOT REPORT RECOVERABLE ERROR IF READ COMMAND
427 055304 001406      BIT #BIT4,510$    ;WAS THIS A READ COMMAND ??
428 055306 162716 000004      BEQ 280$         ;NO !!
429 055312 000410      SUB #4,(SP)      ;RESTORE SP
430 055314 112776 000325 000000      BR 290$          ;SKIP ERROR - DATA WILL BE CORRECTED
431 055322 162716 000002      MOVB #325,@(SP)  ;CHANGE TO NON RECOVERABLE
432 055326 004736 000002      SUB #2,(SP)      ;MOVE SP TO RETURN IF ERROR
433 055330 162716 000010      JSR PC,@(SP)+    ;REPORT ERROR AND RETURN
434      SUB #10,(SP)   ;RESTORE SP TO NO ERROR
435 055334
436
437      ;REPORT DATA BUS PARITY ERROR IF SET, I.E., MDPE = 1
438 055334 032737 000400 001344      BIT #MDPE,RMCS2I ;PARITY ERROR SET??
439 055342 001423      BEQ 300$         ;NO!!
440 055344 013737 001344 001140      MOV RMCS2I,$GDDAT ;EXPECTED STATUS
441 055352 042737 000400 001140      BIC #MDPE,$GDDAT
442 055360 013737 001344 001142      MOV RMCS2I,$BDDAT ;RECEIVED STATUS
443 055366 062716 000004      ADD #4,(SP)      ;MOVE SP TO USER'S ERROR
444 055372 112776 000326 000000      MOVB #326,@(SP)  ;WRITE ERROR NUMBER
445 055400 162716 000002      SUB #2,(SP)      ;MOVE SP TO RETURN IF ERROR
446 055404 004736      JSR PC,@(SP)+    ;REPORT ERROR AND RETURN
447 055406 162716 000010      SUB #10,(SP)     ;MOVE SP TO NO ERROR
448 055412 000137 055630      300$: JMP 340$     ;SKIP WRITE STATUS CHECK
449
450 055416      310$:
451
452      ;TEST TO SEE THAT OFFSET MODE WAS RESET; REPORT ERROR IF "OM" = 1
453 055416 032737 000001 001346      BIT #OM,RMDSI    ;IS OFFSET ON??
454 055424 001423      BEQ 320$         ;NO
455 055426 013737 001346 001140      MOV RMDSI,$GDDAT ;EXPECTED STATUS
456 055434 042737 000001 001140      BIC #OM,$GDDAT
    
```

```

457 055442 013737 001346 001142      MOV      RMDSI,$BDDAT      ;RECEIVED STATUS
458 055450 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
459 055454 112776 000327 000000      MOVVB   #327,@(SP)       ;WRITE ERROR NUMBER IN CALL
460 055462 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
461 055466 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
462 055470 162716 000010                SUB      #10,(SP)        ;MOVE SP TO NO ERROR
463 055474                320$:
464
465                ;TEST FOR DATA BUS PARITY ERROR; REPORT ERROR IF 'DPE' = 1
466 055474 032737 000010 001376      BIT      #DPE,RMER2I     ;DATA PARITY ERROR??
467 055502 001423                BEQ      330$            ;NO!!
468 055504 013737 001376 001140      MOV      RMER2I,$GDDAT   ;EXPECTED STATUS
469 055512 042737 000010 001140      BIC      #DPE,$GDDAT
470 055520 013737 001376 001142      MOV      RMER2I,$BDDAT   ;RECEIVED STATUS
471 055526 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
472 055532 112776 000330 000000      MOVVB   #330,@(SP)       ;WRITE ERROR NUMBER
473 055540 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
474 055544 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
475 055546 162716 000010                SUB      #10,(SP)        ;MOVE SP TO NO ERROR
476 055552                330$:
477
478                ;TEST FOR WRITE CLOCK FAILURE; REPORT ERROR IF 'WCF' = 1
479 055552 032737 000040 001350      BIT      #WCF,RMER1I     ;IS 'WCF' SET??
480 055560 001423                BEQ      340$            ;NO!!
481 055562 013737 001350 001140      MOV      RMER1I,$GDDAT   ;EXPECTED STATUS
482 055570 042737 000040 001140      BIC      #WCF,$GDDAT
483 055576 013737 001350 001142      MOV      RMER1I,$BDDAT   ;RECEIVED STATUS
484 055604 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
485 055610 112776 000331 000000      MOVVB   #331,@(SP)       ;WRITE ERROR NUMBER
486 055616 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
487 055622 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
488 055624 162716 000010                SUB      #10,(SP)        ;MOVE SP TO NO ERROR
489 055630                340$:
490
491                ;REPORT 'DATA LATE' ERROR IF 'DLT' = 1
492 055630 032737 100000 001344      BIT      #DLT,RMCS2I     ;IS 'DLT' SET??
493 055636 001423                BEQ      350$            ;NO!!
494 055640 013737 001344 001140      MOV      RMCS2I,$GDDAT   ;EXPECTED STATUS
495 055646 042737 100000 001140      BIC      #DLT,$GDDAT
496 055654 013737 001344 001142      MOV      RMCS2I,$BDDAT   ;RECEIVED STATUS
497 055662 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USERS ERROR CALL
498 055666 112776 000332 000000      MOVVB   #332,@(SP)       ;WRITE ERROR NUMBER
499 055674 162716 000002                SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
500 055700 004736                JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
501 055702 162716 000010                SUB      #10,(SP)        ;MOVE SP TO NO ERROR
502 055706                350$:
503                ;LOOK FOR UNEXPECTED CHANGES IN DRIVE STATUS
504 055706 013746 001346                MOV      RMDSI,-(SP)      ;STACK DRIVE STATUS
505 055712 042716 147677                BIC      #^C<PIP!MOL!VV>,(SP) ;CLEAR DONT CARES
506 055716 022726 010100                CMP      #MOL!VV,(SP)+   ;IS DRIVE STATUS OK??
507 055722 001522                BEQ      380$            ;YES!!
508
509                ;REPORT ERROR IF POSITIONING IN PROGRESS AND NO SEEK INCOMPLETE ERROR,
510                ;I.E. PIP = 1 AND SKI = 0
511 055724 032737 020000 001346      BIT      #PIP,RMDSI      ;IS 'PIP' SET??
512 055732 001430                BEQ      360$            ;NO!!
513 055734 032737 040000 001376      BIT      #SKI,RMER2I     ;WAS 'SKI' ERROR REPORTED??
    
```



```

514 055742 001024          BNE      360$          :YES-DONT REPORT PIP
515 055744 013737 001346 001140  MOV      RMDSI,$GDDAT  :EXPECTED STATUS
516 055752 042737 020000 001140  BIC      #PIP,$GDDAT
517 055760 013737 001346 001142  MOV      RMDSI,$BDDAT  :RECEIVED STATUS
518 055766 062716 000004          ADD      #4,(SP)       :MOVE SP TO USERS ERROR CALL
519 055772 112776 000333 000000  MOVVB   #333,@(SP)    :WRITE ERROR NUMBER
520 056000 162716 000002          SUB      #2,(SP)       :MOVE SP TO RETURN IF ERROR
521 056004 004736          JSR      PC,@(SP)+    :REPORT ERROR AND RETURN
522 056006 162716 000010          SUB      #10,(SP)     :MOVE SP TO NO ERROR
523 056012 000240
524 056014          360$:
525
526          :REPORT ERROR IF MEDIUM IS NOT ON LINE AND OPI ERROR WAS NOT
527          :REPORTED, I.E., MOL = OPI = 0
528 056014 032737 010000 001346  BIT      #MOL,RMDSI   :IS MEDIUM ON LINE??
529 056022 001027          BNE      370$          :YES!!
530 056024 032737 020000 001350  BIT      #OPI,RMER1I  :WAS OPI ERROR REPORTED??
531 056032 001023          BNE      370$          :YES!!
532 056034 013737 001346 001140  MOV      RMDSI,$GDDAT  :EXPECTED STATUS
533 056042 052737 010000 001140  BIS      #MOL,$GDDAT
534 056050 013737 001346 001142  MOV      RMDSI,$BDDAT  :RECEIVED STATUS
535 056056 062716 000004          ADD      #4,(SP)       :MOVE SP TO USER'S ERROR
536 056062 112776 000334 000000  MOVVB   #334,@(SP)    :WRITE ERROR NUMBER
537 056070 162716 000002          SUB      #2,(SP)       :MOVE SP TO RETURN IF ERROR
538 056074 004736          JSR      PC,@(SP)+    :REPORT ERROR AND RETURN
539 056076 162716 000010          SUB      #10,(SP)     :MOVE SP TO NO ERROR
540 056102          370$:
541
542          :REPORT ERROR IF VOLUME IS NOT VALID AND "IVC" ERROR WAS NOT
543          :REPORTED, I.E., VV = IVC = 0
544 056102 032737 000100 001346  BIT      #VV,RMDSI    :IS VOLUME VALID??
545 056110 001027          BNE      380$          :YES!!
546 056112 032737 010000 001376  BIT      #IVC,RMER2I  :WAS IVC ERROR REPORTED??
547 056120 001033          BNE      390$          :YES!!
548 056122 013737 001346 001140  MOV      RMDSI,$GDDAT  :EXPECTED STATUS
549 056130 052737 000100 001140  BIS      #VV,$GDDAT
550 056136 013737 001346 001142  MOV      RMDSI,$BDDAT  :RECEIVED STATUS
551 056144 062716 000004          ADD      #4,(SP)       :MOVE SP TO USERS ERROR CALL
552 056150 112776 000335 000000  MOVVB   #335,@(SP)    :WRITE ERROR NUMBER
553 056156 162716 000002          SUB      #2,(SP)       :MOVE SP TO RETURN IF ERROR
554 056162 004736          JSR      PC,@(SP)+    :REPORT ERROR AND RETURN
555 056164 162716 000010          SUB      #10,(SP)     :MOVE SP TO NO ERROR
556 056170          380$:
557
558          :AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS FOUND
559 056170 062716 000004          ADD      #4,(SP)       :MOVE SP TO ERROR CALL
560 056174 105776 000000          TSTB   @ (SP)         :ANY ERROR??
561 056200 001403          BEQ     390$          :NO!!
562 056202 062716 000004          ADD      #4,(SP)       :YES - MOVE SP TO ERROR RETURN
563 056206 000402          BR     400$
564 056210 162716 000004          390$: SUB      #4,(SP)     :MOVE SP TO NO ERROR RETURN
565
566 056214 000207          400$: RTS      PC       :RETURN TO USER
567
568 056216 000000          500$: .WORD          :ERROR FLAGS
569 056220 000000          510$: .WORD          :TEMPORARY STORAGE
570

```

```

1      .SBTTL  STATIC DRIVE STATUS CHECK SUBROUTINE
2
3      ;THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
4      ;STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID.  THE SUBROUTINE
5      ;CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
6      ;SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.
7
8      ;THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
9      ;IF TRUE:
10
11      ;      .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
12      ;      THAT MOL IS ASSUMED TO HAVE BEEN SET
13      ;      .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
14      ;      .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
15      ;      .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
16      ;      .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT
17
18      ;THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.
19
20      ;(1)  JSR      PC,STCDRVSTS
21      ;      BR      ???          RETURN HERE IF NO ERROR
22      ;      NOP          RETURN HERE TO REPORT AN ERROR
23      ;      ERROR      ERROR NUMBER DEFINED BY SUB
24      ;      JSR      PC,@(SP)+   GO BACK TO SUB FOR MORE ERROR CHECKS
25      ;      ???          RETURN HERE IF NO MORE ERRORS
26
27 056222 STCDRVSTS:
28
29      ;CLEAR USER'S ERROR CALL
30 056222 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
31 056226 105076 000000 CLR B @ (SP) ;CLEAR ERROR NUMBER
32 056232 162716 000004 SUB #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
33
34      ;SEE IF 'MOL' = 'VV' = 1, AND 'PIP' = 0
35 056236 013746 001346 MOV RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
36 056242 042716 147677 BIC #^C<PIP!MOL!VV>,(SP)
37 056246 022726 010100 CMP #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
38 056252 001524 BEQ 30$ ;YES!!
39
40      ;REPORT AN ERROR IF MOL = 0 AND 'OPI' = 0
41 056254 032737 010000 001346 BIT #MOL,RMDSI ;IS MOL ON ??
42 056262 001030 BNE 10$ ;YES!!
43 056264 032737 020000 001350 BIT #OPI,RMER11 ;WAS 'OPI' SET??
44 056272 001024 BNE 10$ ;YES-DONT REPORT 'MOL' = 0
45 056274 013737 001346 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
46 056302 052737 010000 001140 BIS #MOL,$GDDAT
47 056310 013737 001346 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
48 056316 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
49 056322 112776 000207 000000 MOV B #207,@(SP) ;WRITE ERROR NUMBER IN CALL
50 056330 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
51 056334 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
52 056336 162716 000010 SUB #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN
53 056342 000240 NOP
54
55      10$:
56      ;REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND 'IVC' = 0
57 056344 032737 000100 001346 BIT #VV,RMDSI ;IS 'VV' = 0??
58 056352 001030 BNE 20$ ;NO!!
    
```

```

58 056354 032737 010000 001376      BIT      #IVC,RMER2I      ;WAS "IVC" SET??
59 056362 001024                      BNE      20$             ;YES-DONT REPORT "VV" = 0
60 056364 013737 001346 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
61 056372 052737 000100 001346      BIS      #VV,RMDSI
62 056400 013737 001346 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
63 056406 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
64 056412 112776 000210 000000      MOVVB   #210,a(SP)      ;WRITE ERROR NUMBER IN CALL
65 056420 162716 000002                      SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
66 056424 004736                      JSR      PC,a(SP)+       ;REPORT ERROR VIA USER
67 056426 162716 000010                      SUB      #10,(SP)        ;MOVE SP BACK TO NO ERROR
68 056432 000240
69 056434                                20$:
70
71                                ;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND "SKI" = 0
72 056434 032737 020000 001346      BIT      #PIP,RMDSI      ;IS DRIVE OFF CYLINDER??
73 056442 001430                      BEQ      30$             ;NO!!
74 056444 032737 040000 001376      BIT      #SKI,RMER2I     ;WAS "SKI" SET??
75 056452 001024                      BNE      30$             ;YES-DONT REPORT "PIP" = 1
76 056454 013737 001346 001140      MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
77 056462 042737 020000 001140      BIC      #PIP,$GDDAT
78 056470 013737 001346 001142      MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
79 056476 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
80 056502 112776 000211 000000      MOVVB   #211,a(SP)      ;WRITE ERROR NUMBER IN USER'S CALL
81 056510 162716 000002                      SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
82 056514 004736                      JSR      PC,a(SP)+       ;REPORT ERROR VIA USER
83 056516 162716 000010                      SUB      #10,(SP)        ;MOVE SP TO NO ERROR RETURN
84 056522 000240
85 056524                                30$:
86
87                                ;SEE IF "SKI" = "DVC" = 0
88 056524 013746 001376                      MOV      RMER2I,-(SP)    ;PUT ERROR REG 2 ON STACK
89 056530 042726 137577                      BIC      #^C<SKI!DVC>,(SP)+
90 056534 001460                      BEQ      60$             ;BRANCH IF NO ERROR
91 056536                                40$:
92
93                                ;REPORT AN ERROR IF THERE IS A DEVICE FAULT
94 056536 032737 000200 001376      BIT      #DVC,RMER2I     ;ANY DEVICE FAULT??
95 056544 001424                      BEQ      50$             ;NO!!
96 056546 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
97 056554 042737 000200 001140      BIC      #DVC,$GDDAT
98 056562 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
99 056570 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S CALL
100 056574 112776 000212 000000      MOVVB   #212,a(SP)      ;WRITE NUMBER OF ERROR IN CALL
101 056602 162716 000002                      SUB      #2,(SP)         ;MOVE SP TO RETURN FOR ERROR
102 056606 004736                      JSR      PC,a(SP)+       ;REPORT ERROR VIA USER
103 056610 162716 000010                      SUB      #10,(SP)        ;MOVE SP BACK TO NO ERROR
104 056614 000240
105 056616                                50$:
106
107                                ;REPORT AN ERROR IF "SKI" = 1
108 056616 032737 040000 001376      BIT      #SKI,RMER2I     ;IS THERE A SEEK INCOMPLETE ERROR
109 056624 001424                      BEQ      60$             ;NO!!
110 056626 013737 001376 001140      MOV      RMER2I,$GDDAT    ;EXPECTED STATUS
111 056634 042737 040000 001140      BIC      #SKI,$GDDAT
112 056642 013737 001376 001142      MOV      RMER2I,$BDDAT    ;RECEIVED STATUS
113 056650 062716 000004                      ADD      #4,(SP)         ;MOVE SP TO USER'S ERROR CALL
114 056654 112776 000213 000000      MOVVB   #213,a(SP)      ;WRITE ERROR NUMBER IN USER'S ERROR CALL

```

115	056662	162716	000002		SUB	#2,(SP)		;MOVE SP TO RETURN FOR ERROR
116	056666	004736			JSR	PC,@(SP)+		;REPORT ERROR VIA USER
117	056670	162716	000010		SUB	#10,(SP)		;MOVE SP BACK TO NO ERROR
118	056674	000240			NOP			
119	056676			60\$:				
120								
121								
122	056676	062716	000004					
123	056702	105776	000000		ADD	#4,(SP)		;MOVE SP TO USER'S ERROR CALL
124	056706	001403			TSTB	@(SP)		;WAS AN ERROR DETECTED??
125	056710	062716	000004		BEQ	70\$;NO!!
126	056714	000402			ADD	#4,(SP)		;YES - MOVE SP TO USER'S ERROR RETURN
127	056716	162716	000004		BR	80\$		
128	056722	000240		70\$:	SUB	#4,(SP)		;NO - MOVE SP TO NO ERROR RETURN
129	056724	000207		80\$:	NOP			
130					RTS	PC		;RETURN TO USER

```

1      .SBTTL  STOP AND SHUTDOWN SUBROUTINES
2
3 056726 STOP:
4
5      ;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT
6 056726 012746 000140      MOV      #PR3,-(SP)      ;;PUT NEW PS ON STACK
   056732 012746 056740      MOV      #64$,-(SP)     ;;PUT NEW PC ON STACK
   056736 000002      RTI          ;;POP NEW PC AND PS
   056740
7 056740 000240      64$:      NOP
8
9      ;RAISE PRIORITY TO INHIBIT CONSOLE INTERRUPT
10 056742 012746 000300     MOV      #PR6,-(SP)     ;;PUT NEW PS ON STACK
   056746 012746 056754     MOV      #65$,-(SP)    ;;PUT NEW PC ON STACK
   056752 000002      RTI          ;;POP NEW PC AND PS
   056754
11 056754 000207      65$:      RTS      PC          ;CONTINUE
12
13 056756 005737 001326     SHUT:   TST      CTLFG      ;WAS CONTROL C FLAGGED ?
14 056762 001002      BNE      5$             ;BR IF YES
15 056764 000137 007764     JMP      READY          ;CONTINUE
16 056770 005737 000042     5$:    TST      @#42      ;ANY MONITOR PRESENT ?
17 056774 001015      BNE      10$           ;BR IF YES
18 056776 104401 057004     TYPE    ,65$          ;;TYPE ASCIZ STRING
   057002 000410      BR       64$          ;;GET OVER THE ASCIZ
   057024      ;;65$: .ASCIZ <CRLF><07>/TEST HALTED/<CRLF>
19 057024 000137 005420     64$:   JMP      START        ;GO TO START
20 057030 000137 032340     10$:   JMP      $EOP        ;RETURN CONTROL TO MONITOR
21
22 057034 012737 177777 001326 SHUT2:  MOV      #-1,CTLFG     ;SET THE CONTROL-C FLAG
23 057042 000002      RTI          ;EXIT FROM INTERRUPT
24

```

1

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

057044
057044 010046
057046 010146
057050 010246
057052 010346
057054 010446
057056 010546
057060 016646 000022
057064 016646 000022
057070 016646 000022
057074 016646 000022
057100 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
    
```

```

057102
057102 012666 000022
057106 012666 000022
057112 012666 000022
057116 012666 000022
057122 012605
057124 012604
057126 012603
057130 012602
057132 012601
057134 012600
057136 000002
    
```

```

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

2

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
*BINARY-ASCII NUMBER AND TYPE IT.
*CALL:
*   MOV     NUMBER,-(SP)   ;;NUMBER TO BE TYPED
*   TYPBN
*   ;;TYPE IT
    
```

```

057140 010146
057142 016601 000006
057146 000261
    
```

```

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

```

057150 112737 000060 057212 1$:   MOVB   #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
057156 006101                    ROL    R1             ;;GET THIS BIT
057160 001406                    BEQ    2$             ;;DONE?
057162 105537 057212            ADCB   $BIN           ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
057166 104401 057212            TYPE  , $BIN         ;;GO TYPE THIS BIT
057172 000241                    CLC                     ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
057174 000765                    BR     1$             ;;GO DO THE NEXT BIT
057176 012601                    MOV    (SP)+,R1       ;;POP THE STACK INTO R1
057200 016666 000002 000004    2$:   MOV    2(SP),4(SP)   ;;ADJUST THE STACK
057206 012616                    MOV    (SP)+,(SP)
057210 000002                    RTI                     ;;RETURN TO USER
057212 000      000          $BIN:  .BYTE  0,0           ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
3                                     .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
    
```

```

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

```

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

```

057214 010046          $TYPDS:  MOV    R0,-(SP)      ;;PUSH R0 ON STACK
057214 010146          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
057220 010246          MOV    R2,-(SP)      ;;PUSH R2 ON STACK
057222 010346          MOV    R3,-(SP)      ;;PUSH R3 ON STACK
057224 010546          MOV    R5,-(SP)      ;;PUSH R5 ON STACK
057226 012746 020200    MOV    #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
057232 016605 000020    MOV    20(SP),R5      ;;GET THE INPUT NUMBER
057236 100004          BPL    1$             ;;BR IF INPUT IS POS.
057240 005405          NEG    R5             ;;MAKE THE BINARY NUMBER POS.
057242 112766 000055 000001 1$:   MOVB   #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
057250 005000          CLR    R0             ;;ZERO THE CONSTANTS INDEX
057252 012703 057430    MOV    #$DBLK,R3     ;;SETUP THE OUTPUT POINTER
057256 112723 000040    MOVB   #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
057262 005002          CLR    R2             ;;CLEAR THE BCD NUMBER
057264 016001 057420    MOV    $DTBL(R0),R1  ;;GET THE CONSTANT
057270 160105          3$:   SUB    R1,R5          ;;FORM THIS BCD DIGIT
057272 002402          BLT    4$             ;;BR IF DONE
057274 005202          INC    R2             ;;INCREASE THE BCD DIGIT BY 1
057276 000774          BR     3$
057300 060105          4$:   ADD    R1,R5          ;;ADD BACK THE CONSTANT
057302 005702          TST    R2             ;;CHECK IF BCD DIGIT=0
057304 001002          BNE    5$             ;;FALL THROUGH IF 0
057306 105716          TSTB   (SP)          ;;STILL DOING LEADING 0'S?
057310 100407          BMI    7$             ;;BR IF YES
057312 106316          5$:   ASLB   (SP)          ;;MSD?
057314 103003          BCC    6$             ;;BR IF NO
057316 116663 000001 177777 6$:   MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
057324 052702 000060    7$:   BIS    #'0,R2       ;;MAKE THE BCD DIGIT ASCII
057330 052702 000040    BIS    #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
057334 110223          MOVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
057336 005720          TST    (R0)+       ;;JUST INCREMENTING
057340 020027 000010    CMP    R0,#10        ;;CHECK THE TABLE INDEX
    
```

```

057344 002746          BLT      2$          ;;GO DO THE NEXT DIGIT
057346 003002          BGT      8$          ;;GO TO EXIT
057350 010502          MOV      R5,R2        ;;GET THE LSD
057352 000764          BR       6$          ;;GO CHANGE TO ASCII
057354 105726          8$:    TSTB   (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
057356 100003          BPL      9$          ;;BR IF NO
057360 116663 177777 177776 9$:    MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
057366 105013          CLRB   (R3)          ;;SET THE TERMINATOR
057370 012605          MOV     (SP)+,R5      ;;POP STACK INTO R5
057372 012603          MOV     (SP)+,R3      ;;POP STACK INTO R3
057374 012602          MOV     (SP)+,R2      ;;POP STACK INTO R2
057376 012601          MOV     (SP)+,R1      ;;POP STACK INTO R1
057400 012600          MOV     (SP)+,R0      ;;POP STACK INTO R0
057402 104401 057430  TYPE     ,SDBLK      ;;NOW TYPE THE NUMBER
057406 016666 000002 000004  MOV     2(SP),4(SP)    ;;ADJUST THE STACK
057414 012616          MOV     (SP)+,(SP)
057416 000002          RTI
057420 023420          SDTBL: 10000.
057422 001750          1000.
057424 000144          100.
057426 000012          10.
057430          SDBLK: .BLKW 4
          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
    
```

4

```

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

```

057440 017646 000000          $TYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
057444 116637 000001 057663  MOVB   1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
057452 112637 057665          MOVB   (SP)+, $OMODE+1   ;;NUMBER OF DIGITS TO TYPE
057456 062716 000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
057462 000406          BR     $TYPON
057464 112737 000001 057663  $TYPOC: MOVB   #1, $OFILL      ;;SET THE ZERO FILL SWITCH
057472 112737 000006 057665  MOVB   #6, $OMODE+1     ;;SET FOR SIX(6) DIGITS
057500 112737 000005 057662  $TYPON: MOVB   #5, $OCNT      ;;SET THE ITERATION COUNT
057506 010346          MOV     R3,-(SP)        ;;SAVE R3
057510 010446          MOV     R4,-(SP)        ;;SAVE R4
    
```


057512	010546		MOV	R5,-(SP)	::SAVE R5
057514	113704	057665	MOVB	\$OMODE+1,R4	::GET THE NUMBER OF DIGITS TO TYPE
057520	005404		NEG	R4	
057522	062704	000006	ADD	#6,R4	::SUBTRACT IT FOR MAX. ALLOWED
057526	110437	057664	MOVB	R4,\$OMODE	::SAVE IT FOR USE
057532	113704	057663	MOVB	\$OFILL,R4	::GET THE ZERO FILL SWITCH
057536	016605	000012	MOV	12(SP),R5	::PICKUP THE INPUT NUMBER
057542	005003		CLR	R3	::CLEAR THE OUTPUT WORD
057544	006105	1\$:	ROL	R5	::ROTATE MSB INTO 'C'
057546	000404		BR	3\$::GO DO MSB
057550	006105	2\$:	ROL	R5	::FORM THIS DIGIT
057552	006105		ROL	R5	
057554	006105		ROL	R5	
057556	010503		MOV	R5,R3	
057560	006103	3\$:	ROL	R3	::GET LSB OF THIS DIGIT
057562	105337	057664	DECB	\$OMODE	::TYPE THIS DIGIT?
057566	100016		BPL	7\$::BR IF NO
057570	042703	177770	BIC	#177770,R3	::GET RID OF JUNK
057574	001002		BNE	4\$::TEST FOR 0
057576	005704		TST	R4	::SUPPRESS THIS 0?
057600	001403		BEQ	5\$::BR IF YES
057602	005204	4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
057604	052703	000060	BIS	#'0,R3	::MAKE THIS DIGIT ASCII
057610	052703	000040	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
057614	110337	057660	MOVB	R3,8\$::SAVE FOR TYPING
057620	104401	057660	TYPE	,8\$::GO TYPE THIS DIGIT
057624	105337	057662	DECB	\$OCNT	::COUNT BY 1
057630	003347		BGT	2\$::BR IF MORE TO DO
057632	002402		BLT	6\$::BR IF DONE
057634	005204		INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
057636	000744		BR	2\$::GO DO THE LAST DIGIT
057640	012605	6\$:	MOV	(SP)+,R5	::RESTORE R5
057642	012604		MOV	(SP)+,R4	::RESTORE R4
057644	012603		MOV	(SP)+,R3	::RESTORE R3
057646	016666	000002 000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
057654	012616		MOV	(SP)+,(SP)	
057656	000002		RTI		::RETURN
057660	000	8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
057661	000		.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
057662	000	\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
057663	000	\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
057664	000000	\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE
		.SBTTL	TYPE ROUTINE		

5

```

*****
*ROUTINE TO TYPE ASCII 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 ASCII STRING
*OR
*   TYPE
*   MESADR
  
```

```

;*
057666 105737 001173 $TYPE: TSTB $TPFLG ::IS THERE A TERMINAL?
057672 100002 BPL 1$ ::BR IF YES
057674 000000 HALT ::HALT HERE IF NO TERMINAL
057676 000430 BR 3$ ::LEAVE
057700 010046 1$: MOV R0,-(SP) ::SAVE R0
057702 017600 000002 MOV @2(SP),R0 ::GET ADDRESS OF ASCIZ STRING
057706 122737 000001 001242 CMPB #APTENV,$ENV ::RUNNING IN APT MODE
057714 001011 BNE 62$ ::NO,GO CHECK FOR APT CONSOLE
057716 132737 000100 001243 BITB #APTSPOOL,$ENVM ::SPOOL MESSAGE TO APT
057724 001405 BEQ 62$ ::NO,GO CHECK FOR CONSOLE
057726 010037 057736 MOV R0,61$ ::SETUP MESSAGE ADDRESS FOR APT
057732 004737 062660 JSR PC,$ATY3 ::SPOOL MESSAGE TO APT
057736 000000 61$: .WORD 0 ::MESSAGE ADDRESS
057740 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ::APT CONSOLE SUPPRESSED
057746 001003 BNE 60$ ::YES,SKIP TYPE OUT
057750 112046 2$: MOV (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
057752 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
057754 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
057756 012600 60$: MOV (SP)+,R0 ::RESTORE R0
057760 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
057764 000002 RTI ::RETURN
057766 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
057772 001430 BEQ 8$
057774 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
060000 001006 BNE 5$
060002 005726 TST (SP)+ ::POP <CR><LF> EQUIV
060004 104401 TYPE ::TYPE A CR AND LF
060006 001217 $CRLF
060010 105037 060216 CLR B $CHARCNT ::CLEAR CHARACTER COUNT
060014 000755 BR 2$ ::GET NEXT CHARACTER
060016 004737 060100 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
060022 123726 001172 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
060026 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
060030 013746 001170 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
::AND THE NULL CHAR.
060034 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
060040 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
060042 004737 060100 JSR PC,$TYPEC ::GO TYPE A NULL
060046 105337 060216 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
060052 000770 BR 7$ ::LOOP

:HORIZONTAL TAB PROCESSOR
060054 112716 000040 8$: MOV B #' (SP) ::REPLACE TAB WITH SPACE
060060 004737 060100 9$: JSR PC,$TYPEC ::TYPE A SPACE
060064 132737 000007 060216 BITB #7,$CHARCNT ::BRANCH IF NOT AT
060072 001372 BNE 9$ ::TAB STOP
060074 005726 TST (SP)+ ::POP SPACE OFF STACK
060076 000724 BR 2$ ::GET NEXT CHARACTER
060100 $TYPEC:
060100 105777 121054 TSTB @$TKS ::CHAR IN KYBD BUFFER?
060104 100022 BPL 10$ ::BR IF NOT
060106 017746 121050 MOV @$TKB,-(SP) ::GET CHAR
060112 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS
060116 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF

```

```

060122 001012
060124 105777 121030
060130 100375
060132 117716 121024
060136 042716 177600
060142 122716 000021
060146 001366
060150
060150 005726
060152
060152 105777 121006
060156 100375
060160 116677 000002 121000
060166 122766 000015 000002
060174 001003
060176 105037 060216
060202 000406
060204 122766 000012 000002
060212 001402
060214 105227
060216 000000
060220 000207

```

```

101$: BNE 102$ ;;BR IF NOT
TSTB @STKS ;;WAIT FOR CHAR
BPL 101$
MOVB @STKB,(SP) ;;GET CHAR
BIC #177600,(SP) ;;STRIP IT
CMPB #SXON,(SP) ;;WAS IT XON?
BNE 101$ ;;BR IF NOT

102$: TST (SP)+ ;;FIX STACK

10$: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
BPL 10$
MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
BNE 1$ ;;BRANCH IF NO
CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
BR $TYPEX ;;EXIT
1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
BEQ $TYPEX ;;BRANCH IF YES
INCB (PC)+ ;;COUNT THE CHARACTER
$CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
$TYPEX: RTS PC

```

5

.SBTTL SCOPE HANDLER ROUTINE

```

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

```

```

060222
060222 104410
060224 004737 056726
060230 032777 040000 120716
060236 001131
060240 000416
060242 013746 000004
060246 012737 060266 000004
060254 005737 177060
060260 012637 000004
060264 000500
060266 022626
060270 012637 000004
060274 000440
060276
060276 032777 000400 120650
060304 001421

```

```

$SCOPE:
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
JSR PC,STOP
1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
BNE $OVER ;;YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
$XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
;;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5,@#ERRVEC ;;SET FOR TIMEOUT
TST @#177060 ;;TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
BR $SVLAD ;;GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
BR 7$ ;;LOOP ON THE PRESENT TEST
6$:;#####END OF CODE FOR THE XOR TESTER#####
BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
BEQ 2$ ;;BR IF NO

```

```

060306 005046          CLR      -(SP)          ;;CLEAR A TEMP. LOCATION
060310 117716 120640  MOVB     @SWR,(SP)     ;;PICKUP THE DESIRED TEST NUMBER
060314 001414          BEQ      8$           ;;BRANCH IF BAD TEST NUMBER IN SWR
060316 022716 000033  CMP      #33,(SP)     ;;CHECK THE NUMBER IN THE SWR
060322 002411          BLT      8$           ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
060324 011637 001116  MOV      (SP), $STSTM  ;;UPDATE THE TEST NUMBER
060330 005316          DEC      (SP)        ;;BACKUP BY ONE
060332 006316          ASL      (SP)        ;;SCALE THE TEST NUMBER AS AN INDEX
060334 062716 060540  ADD      $$SW08TBL,(SP) ;;FORM THE ADDRESS OF TEST POINTER
060340 013637 001122  MOV      @($)+,$LPADR  ;;SET LOOP ADDRESS TO DESIRED TEST
060344 000466          BR       $OVER       ;;GO LOOP ON THE TEST
060346 005726          8$: TST      (SP)+     ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
060350 105737 001117  2$: TSTB   $ERFLG     ;;HAS AN ERROR OCCURRED?
060354 001421          BEQ      3$           ;;BR IF NO
060356 123737 001131 001117  CMPB   $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
060364 101015          BHI      3$           ;;BR IF NO
060366 032777 001000 120560  BIT    #BIT09,@SWR   ;;LOOP ON ERROR?
060374 001404          BEQ      4$           ;;BR IF NO
060376 013737 001124 001122  7$: MOV    $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
060404 000446          BR       $OVER
060406 105037 001117          4$: CLRB   $ERFLG     ;;ZERO THE ERROR FLAG
060412 005037 001206          CLR    $TIMES     ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
060416 000415          BR     1$           ;;ESCAPE TO THE NEXT TEST
060420 032777 004000 120526  3$: BIT    #BIT11,@SWR  ;;INHIBIT ITERATIONS?
060426 001011          BNE     1$           ;;BR IF YES
060430 005737 001230          TST    $PASS      ;;IF FIRST PASS OF PROGRAM
060434 001406          BEQ     1$           ;;INHIBIT ITERATIONS
060436 005237 001120          INC    $ICNT     ;;INCREMENT ITERATION COUNT
060442 023737 001206 001120  CMP    $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
060450 002024          BGE    $OVER     ;;BR IF MORE ITERATION REQUIRED
060452 012737 000001 001120  1$: MOV    #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
060460 013737 060536 001206  MOV    $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
060466 105237 001116          $SVLAD: INCB   $STSTM  ;;COUNT TEST NUMBERS
060472 113737 001116 001226  MOVB   $STSTM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
060500 011637 001122          MOV    (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
060504 011637 001124          MOV    (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
060510 005037 001210          CLR    $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
060514 112737 000001 001131  MOVB   #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
060522 013777 001116 120426  $OVER: MOV    $STSTM,@DISPLAY ;;DISPLAY TEST NUMBER
060530 013716 001122          MOV    $LPADR,(SP) ;;FUDGE RETURN ADDRESS
060534 000002          RTI           ;;FIXES PS
060536 000012          $MXCNT: 10.    ;;MAX. NUMBER OF ITERATIONS
060540          $SW08TBL:
060540 010132          .WORD  TST1+2     ;;STARTING ADDRESS OF TEST 1
060542 010316          .WORD  TST2+2     ;;STARTING ADDRESS OF TEST 2
060544 010502          .WORD  TST3+2     ;;STARTING ADDRESS OF TEST 3
060546 010704          .WORD  TST4+2     ;;STARTING ADDRESS OF TEST 4
060550 011546          .WORD  TST5+2     ;;STARTING ADDRESS OF TEST 5
060552 012402          .WORD  TST6+2     ;;STARTING ADDRESS OF TEST 6
060554 013172          .WORD  TST7+2     ;;STARTING ADDRESS OF TEST 7
060556 014132          .WORD  TST10+2    ;;STARTING ADDRESS OF TEST 10
060560 014752          .WORD  TST11+2    ;;STARTING ADDRESS OF TEST 11
060562 015542          .WORD  TST12+2    ;;STARTING ADDRESS OF TEST 12
060564 016500          .WORD  TST13+2    ;;STARTING ADDRESS OF TEST 13
060566 017274          .WORD  TST14+2    ;;STARTING ADDRESS OF TEST 14
060570 020106          .WORD  TST15+2    ;;STARTING ADDRESS OF TEST 15
060572 020720          .WORD  TST16+2    ;;STARTING ADDRESS OF TEST 16

```

060574 021542
060576 022304
060600 023046
060602 023610
060604 024114
060606 024420
060610 024762
060612 025304
060614 025622
060616 026534
060620 027356
060622 030140
060624 031310

.WORD TST17+2 ;; STARTING ADDRESS OF TEST 17
.WORD TST20+2 ;; STARTING ADDRESS OF TEST 20
.WORD TST21+2 ;; STARTING ADDRESS OF TEST 21
.WORD TST22+2 ;; STARTING ADDRESS OF TEST 22
.WORD TST23+2 ;; STARTING ADDRESS OF TEST 23
.WORD TST24+2 ;; STARTING ADDRESS OF TEST 24
.WORD TST25+2 ;; STARTING ADDRESS OF TEST 25
.WORD TST26+2 ;; STARTING ADDRESS OF TEST 26
.WORD TST27+2 ;; STARTING ADDRESS OF TEST 27
.WORD TST30+2 ;; STARTING ADDRESS OF TEST 30
.WORD TST31+2 ;; STARTING ADDRESS OF TEST 31
.WORD TST32+2 ;; STARTING ADDRESS OF TEST 32
.WORD TST33+2 ;; STARTING ADDRESS OF TEST 33

7

.SBTTL ERROR HANDLER ROUTINE

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

060626
060626 104410
060630 105237 001117
060634 001775
060636 013777 001116 120312
060644 032777 002000 120302
060652 001402
060654 104401 001212
060660 005237 001126
060664 011637 001132
060670 162737 000002 001132
060676 117737 120230 001130
060704 032777 020000 120242
060712 001004
060714 004737 032550
060720 104401 001217
060724
060724 122737 000001 001242
060732 001007
060734 113737 001130 060746
060742 004737 062670
060746 000
060747 000
060750 000777
060752 005777 120176
060756 100002
060760 000000
060762 104410
060764 032777 001000 120162
060772 001402

\$ERROR:
7\$: CKSWR ;; TEST FOR CHANGE IN SOFT-SWR
INCB \$ERFLG ;; SET THE ERROR FLAG
BEQ 7\$;; DON'T LET THE FLAG GO TO ZERO
MOV \$STNM,@DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,@SWR ;; BELL ON ERROR?
BEQ 1\$;; NO - SKIP
TYPE , \$BELL ;; RING BELL
1\$: INC \$ERTTL ;; COUNT THE NUMBER OF ERRORS
MOV (SP), \$ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
SUB #2, \$ERRPC
MOVB @ \$ERRPC, \$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@SWR ;; SKIP TYPEOUT IF SET
BNE 20\$;; SKIP TYPEOUTS
JSR PC, ERRTP ;; GO TO USER ERROR ROUTINE
TYPE , \$CRLF
20\$: CMPB #APTENV, \$ENV ;; RUNNING IN APT MODE
BNE 2\$;; NO, SKIP APT ERROR REPORT
MOVB \$ITEMB, 21\$;; SET ITEM NUMBER AS ERROR NUMBER
JSR PC, SATY4 ;; REPORT FATAL ERROR TO APT
21\$: .BYTE 0
.BYTE 0
22\$: BR 22\$;; APT ERROR LOOP
2\$: TST @SWR ;; HALT ON ERROR
BPL 3\$;; SKIP IF CONTINUE
HALT ;; HALT ON ERROR!
CKSWR ;; TEST FOR CHANGE IN SOFT-SWR
3\$: BIT #BIT09,@SWR ;; LOOP ON ERROR SWITCH SET?
BEQ 4\$;; BR IF NO
4\$

```

060774 013716 001124      MOV    $LPERR,(SP)    ;;FUDGE RETURN FOR LOOPING
061000 005737 001210      4$:   TST    $ESCAPE    ;;CHECK FOR AN ESCAPE ADDRESS
061004 001402              BEQ    5$            ;;BR IF NONE
061006 013716 001210      MOV    $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
061012 022737 032530 000042 5$:   CMP    #SENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
061020 001001              BNE    6$            ;;BRANCH IF NO
061022 000000              HALT                   ;;YES
061024 000002              6$:   RTI                    ;;RETURN
8      .SBTTL  TTY INPUT ROUTINE

```

```

*****
061026 000000      .ENABL  LSB
061030 000000      $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
061032 000000      $TKQIN: .WORD 0     ;;INPUT POINTER
061034 061035      $TKQOUT: .WORD 0   ;;OUTPUT POINTER
                          $TKQSRT: .BLKB 1      ;;TTY KEYBOARD QUEUE
                          $TKQEND=.
                          .EVEN

```

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

```

;*CALL:
;*
;*   JSR    PC,$TKINT
;*   RETURN
;*
061036 005037 061026      $TKINT: CLR    $TKCNT    ;;CLEAR COUNT OF ITEMS IN QUEUE
061042 012737 061034 061030  MOV    #TKQSRT,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
061050 013737 061030 061032  MOV    $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
061056 012737 061106 000060  MOV    #TKSRV,@TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
061064 012737 000200 000062  MOV    #200,@TKVEC+2  ;;'BR' LEVEL 4
061072 005777 120064      TST    @TKB           ;;CLEAR DONE FLAG
061076 012777 000100 120054  MOV    #100,@TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
061104 000207              RTS    PC                ;;RETURN TO CALLER

```

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

```

061106 117746 120050      $TKSRV: MOVB   @TKB,-(SP)  ;;PICKUP THE CHARACTER
061112 042716 177600      BIC    #^C177,(SP)  ;;STRIP THE JUNK
061116 021627 000003      CMP    (SP),#3      ;;IS IT A CONTROL C?
061122 001007              BNE    1$            ;;BRANCH IF NO
061124 104401 062222      TYPE   ,SCNTLC     ;;TYPE A CONTROL-C (^C)
061130 004737 061036      JSR    PC,$TKINT   ;;INIT THE KEYBOARD
061134 005726              TST    (SP)+        ;;CLEAN UP STACK
061136 000137 057034      JMP    SHUT2       ;;CONTROL C RESTART
061142 021627 000007      1$:   CMP    (SP),#7    ;;IS IT A CONTROL G?
061146 001004              BNE    2$            ;;BRANCH IF NO
061150 022737 000176 001154  CMP    #SWREG,SWR   ;;IS SOFT-SWR SELECTED?
061156 001500              BEQ    6$            ;;GO TO SWR CHANGE

```

```

061160
061160 022737 000001 061026 2$: CMP #1,$TKCNT ;; IS THE QUEUE FULL?
061166 001004 BNE 3$ ;; BRANCH IF NO
061170 104401 001212 TYPE ,SBELL ;; RING THE TTY BELL
061174 005726 TST (SP)+ ;; CLEAN CHARACTER OFF OF STACK
061176 000451 BR 5$ ;; EXIT
061200 021627 000023 3$: CMP (SP),#23 ;; IS IT A CONTROL-S?
061204 001021 BNE 32$ ;; BRANCH IF NO
061206 005077 117746 CLR @TKS ;; DISABLE TTY KEYBOARD INTERRUPTS
061212 005726 TST (SP)+ ;; CLEAN CHAR OFF STACK
061214 105777 117740 31$: TSTB @TKS ;; WAIT FOR A CHAR
061220 100375 BPL 31$ ;; LOOP UNTIL ITS THERE
061222 117746 117734 MOVB @TKB,-(SP) ;; GET THE CHARACTER
061226 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
061232 022627 000021 CMP (SP)+,#21 ;; IS IT A CONTROL-Q?
061236 001366 BNE 31$ ;; BRANCH IF NO
061240 012777 000100 117712 MOV #100,@TKS ;; REENABLE TTY KEYBOARD INTERRUPTS
061246 000002 RTI ;; RETURN
061250 005237 061026 32$: INC $TKCNT ;; COUNT THIS CHARACTER
061254 021627 000140 CMP (SP),#140 ;; IS IT UPPER CASE?
061260 002405 BLT 4$ ;; BRANCH IF YES
061262 021627 000175 CMP (SP),#175 ;; IS IT A SPECIAL CHAR?
061266 003002 BGT 4$ ;; BRANCH IF YES
061270 042716 000040 BIC #40,(SP) ;; MAKE IT UPPER CASE
061274 112677 177530 4$: MOVB (SP)+,@TKQIN ;; AND PUT IT IN QUEUE
061300 005237 061030 INC $TKQIN ;; UPDATE THE POINTER
061304 023727 061030 061035 CMP $TKQIN,$$TKQEND ;; GO OFF THE END?
061312 001003 BNE 5$ ;; BRANCH IF NO
061314 012737 061034 061030 MOV $$TKQSR,$TKQIN ;; RESET THE POINTER
061322 000002 5$: 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.
  
```

```

061324 022737 000176 001154 $CKSWR: CMP #SWREG,$SWR ;; IS THE SOFT-SWR SELECTED
061332 001124 BNE 15$ ;; EXIT IF NOT
061334 105777 117620 TSTB @TKS ;; IS A CHAR WAITING?
061340 100121 BPL 15$ ;; IF NOT, EXIT
061342 117746 117614 MOVB @TKB,-(SP) ;; YES
061346 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
061352 021627 000007 CMP (SP),#7 ;; IS IT A CONTROL-G?
061356 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.
  
```

```

061360 123727 001150 000001 6$: CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
061366 001674 BEQ 2$ ;; BRANCH IF YES
061370 005726 TST (SP)+ ;; CLEAR CONTROL-G OFF STACK
061372 004737 061036 JSR PC,$TKINT ;; FLUSH THE TTY INPUT QUEUE
061376 005077 117556 CLR @TKS ;; DISABLE TTY KEYBOARD INTERRUPTS
061402 112737 000001 001151 MOVB #1,$INTAG ;; SET INTERRUPT MODE INDICATOR
  
```

061410	104401	062234		TYPE	,\$CNTLG	::ECHO THE CONTROL-G (^G)
061414	104401	062241		SGT5WR: TYPE	,\$MSWR	::TYPE CURRENT CONTENTS
061420	013746	000176		MOV	SWREG,-(SP)	::SAVE SWREG FOR TYPEOUT
061424	104402			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
061426	104401	062252		TYPE	,\$MNEW	::PROMPT FOR NEW SWR
061432	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
061434	005046			CLR	-(SP)	::THE NEW SWR
061436	105777	117516	7\$:	TSTB	@\$TKS	::CHAR THERE?
061442	100375			BPL	7\$::IF NOT TRY AGAIN
061444	117746	117512		MOVB	@\$TKB,-(SP)	::PICK UP CHAR
061450	042716	177600		BIC	#^C177,(SP)	::MAKE IT 7-BIT ASCII
061454	021627	000003		CMP	(SP),#3	::IS IT A CONTROL-C?
061460	001015			BNE	9\$::BRANCH IF NOT
061462	104401	062222		TYPE	,\$CNTLC	::YES, ECHO CONTROL-C (^C)
061466	062706	000006		ADD	#6,SP	::CLEAN UP STACK
061472	123727	001151	000001	CMPB	\$INTAG,#1	::REENABLE TTY KEYBOARD INTERRUPTS?
061500	001003			BNE	8\$::BRANCH IF NO
061502	012777	000100	117450	MOV	#100,@\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
061510	000137	057034	8\$:	JMP	SHUT2	::CONTROL-C RESTART
061514	021627	000025	9\$:	CMP	(SP),#25	::IS IT A CONTROL-U?
061520	001005			BNE	10\$::BRANCH IF NOT
061522	104401	062227		TYPE	,\$CNTLU	::YES, ECHO CONTROL-U (^U)
061526	062706	000006	20\$:	ADD	#6,SP	::IGNORE PREVIOUS INPUT
061532	000737			BR	19\$::LET'S TRY IT AGAIN
061534	021627	000015	10\$:	CMP	(SP),#15	::IS IT A <CR>?
061540	001022			BNE	16\$::BRANCH IF NO
061542	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
061546	001403			BEQ	11\$::BRANCH IF YES
061550	016677	000002	117376	MOV	2(SP),@SWR	::SAVE NEW SWR
061556	062706	000006	11\$:	ADD	#6,SP	::CLEAR UP STACK
061562	104401	001217	14\$:	TYPE	,\$CRLF	::ECHO <CR> AND <LF>
061566	123727	001151	000001	CMPB	\$INTAG,#1	::RE-ENABLE TTY KBD INTERRUPTS?
061574	001003			BNE	15\$::BRANCH IF NOT
061576	012777	000100	117354	MOV	#100,@\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
061604	000002		15\$:	RTI		::RETURN
061606	004737	060100	16\$:	JSR	PC,\$TYPEC	::ECHO CHAR
061612	021627	000060		CMP	(SP),#60	::CHAR < 0?
061616	002420			BLT	18\$::BRANCH IF YES
061620	021627	000067		CMP	(SP),#67	::CHAR > 7?
061624	003015			BGT	18\$::BRANCH IF YES
061626	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
061632	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
061636	001403			BEQ	17\$::BRANCH IF YES
061640	006316			ASL	(SP)	::NO, SHIFT PRESENT
061642	006316			ASL	(SP)	:: CHAR OVER TO MAKE
061644	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
061646	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
061652	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
061656	000667			BR	7\$::GET THE NEXT ONE
061660	104401	001216	18\$:	TYPE	,\$QUES	::TYPE ?<CR><LF>

061664 000720

.DSABL BR 20\$;;SIMULATE CONTROL-U
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

061666 011646 \$RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC AND
061670 016666 000004 000002 MOV 4(SP),2(SP) ;;THE PS
061676 005066 000004 CLR 4(SP) ;;GET READY FOR A CHARACTER
061702 005046 CLR -(SP) ;;PUT NEW PS ON STACK
061704 012746 061712 MOV #64\$,-(SP) ;;PUT NEW PC ON STACK
061710 000002 RTI ;;POP NEW PC AND PS
061712 64\$:
061712 005737 061026 1\$: TST \$TKCNT ;;WAIT ON A CHARACTER
061716 001775 BEQ 1\$
061720 005337 061026 DEC \$TKCNT ;;DECREMENT THE COUNTER
061724 117766 177102 000004 MOVB @TKQOUT,4(SP) ;;GET ONE CHARACTER
061732 005237 061032 INC \$TKQOUT ;;UPDATE THE POINTER
061736 023727 061032 061035 CMP \$TKQOUT,#TKQEND ;;DID IT GO OFF OF THE END?
061744 001003 BNE 2\$;;BRANCH IF NO
061746 012737 061034 061032 MOV #TKQSRT,\$TKQOUT ;;RESET THE POINTER
061754 000002 2\$: 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

061756 010346 \$RDLIN: MOV R3,-(SP) ;;SAVE R3
061760 005046 CLR -(SP) ;;CLEAR THE RUBOUT KEY
061762 012703 062212 1\$: MOV #TTYIN,R3 ;;GET ADDRESS
061766 022703 062222 2\$: CMP #TTYIN+8.,R3 ;;BUFFER FULL?
061772 101456 BLOS 4\$;;BR IF YES
061774 104411 RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
061776 112613 MOV (SP)+,(R3) ;;GET CHARACTER
062000 122713 000177 10\$: CMPB #177,(R3) ;;IS IT A RUBOUT
062004 001022 BNE 5\$;;BR IF NO
062006 005716 TST (SP) ;;IS THIS THE FIRST RUBOUT?
062010 001007 BNE 6\$;;BR IF NO
062012 112737 000134 062210 MOVB #'\\,9\$;;TYPE A BACK SLASH
062020 104401 062210 TYPE ,9\$
062024 012716 177777 6\$: MOV #-1,(SP) ;;SET THE RUBOUT KEY
062030 005303 DEC R3 ;;BACKUP BY ONE
062032 020327 062212 CMP R3,#TTYIN ;;STACK EMPTY?
062036 103434 BLO 4\$;;BR IF YES
062040 111337 062210 MOVB (R3),9\$;;SETUP TO TYPEOUT THE DELETED CHAR.
062044 104401 062210 TYPE ,9\$;;GO TYPE
062050 000746 BR 2\$;;GO READ ANOTHER CHAR.
062052 005716 5\$: TST (SP) ;;RUBOUT KEY SET?
062054 001406 BEQ 7\$;;BR IF NO

```

062056 112737 000134 062210      MOVB    #' \,9$          ;;TYPE A BACK SLASH
062064 104401 062210              TYPE    ,9$
062070 005016              CLR     (SP)           ;;CLEAR THE RUBOUT KEY
062072 122713 000025      7$:    CMPB    #25,(R3)    ;;IS CHARACTER A CTRL U?
062076 001003              BNE     8$            ;;BR IF NO
062100 104401 062227              TYPE    ,SCNTLU        ;;TYPE A CONTROL 'U'
062104 000726              BR      1$            ;;GO START OVER
062106 122713 000022      8$:    CMPB    #22,(R3)    ;;IS CHARACTER A '^R'?
062112 001011              BNE     3$            ;;BRANCH IF NO
062114 105013              CLRB   (R3)           ;;CLEAR THE CHARACTER
062116 104401 001217              TYPE    ,SCRLF        ;;TYPE A 'CR' & 'LF'
062122 104401 062212              TYPE    ,STTYIN       ;;TYPE THE INPUT STRING
062126 000717              BR      2$            ;;GO PICKUP ANOTHER CHACTER
062130 104401 001216      4$:    TYPE    ,SQUES      ;;TYPE A '?'
062134 000712              BR      1$            ;;CLEAR THE BUFFER AND LOOP
062136 111337 062210      3$:    MOVB    (R3),9$      ;;ECHO THE CHARACTER
062142 104401 062210              TYPE    ,9$
062146 122723 000015      CMPB    #15,(R3)+     ;;CHECK FOR RETURN
062152 001305              BNE     2$            ;;LOOP IF NOT RETURN
062154 105063 177777      CLRB   -1(R3)        ;;CLEAR RETURN (THE 15)
062160 104401 001220              TYPE    ,SLF          ;;TYPE A LINE FEED
062164 005726              TST    (SP)+         ;;CLEAN RUBOUT KEY FROM THE STACK
062166 012603              MOV     (SP)+,R3      ;;RESTORE R3
062170 011646              MOV     (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
062172 016666 000004 000002      MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
062200 012766 062212 000004      MOV     #STTYIN,4(SP)
062206 000002              RTI                    ;;RETURN
062210      000          9$:    .BYTE    0            ;;STORAGE FOR ASCII CHAR. TO TYPE
062211      000              .BYTE    0            ;;TERMINATOR
062212      $TTYIN: .BLKB    8.      ;;RESERVE 8 BYTES FOR TTY INPUT
062222      136      103      015    $CNTLC: .ASCIZ  /^C/<15><12>  ;;CONTROL 'C'
062227      136      125      015    $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
062234      136      107      015    $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
062241      015      012      123    $MSWR:  .ASCIZ  <15><12>/SWR = /
062252      040      040      116    $MNEW:  .ASCIZ  / NEW = /
          .EVEN
          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

```

9

```

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

```

```

*      RDOCT          ;;READ AN OCTAL NUMBER
*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
*                    ;;HIGH ORDER BITS ARE IN $HIOCT

```

```

062264 011646 000004 000002  $RDOCT: MOV     (SP),-(SP)  ;;PROVIDE SPACE FOR THE
062266 016666              MOV     4(SP),2(SP)  ;;INPUT NUMBER
062274 010046              MOV     R0,-(SP)     ;;PUSH R0 ON STACK
062276 010146              MOV     R1,-(SP)     ;;PUSH R1 ON STACK
062300 010246              MOV     R2,-(SP)     ;;PUSH R2 ON STACK
062302 104412      1$:    RDLIN          ;;READ AN ASCII LINE
062304 012600              MOV     (SP)+,R0     ;;GET ADDRESS OF 1ST CHARACTER
062306 005001              CLR     R1           ;;CLEAR DATA WORD
062310 005002              CLR     R2
062312 112046      2$:    MOVB    (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
062314 001412              BEQ    3$           ;;IF ZERO GET OUT

```

```

062316 006301          ASL    R1          ;;*2
062320 006102          ROL    R2
062322 006301          ASL    R1          ;;*4
062324 006102          ROL    R2
062326 006301          ASL    R1          ;;*8
062330 006102          ROL    R2
062332 042716 177770  BIC    #^C7,(SP)    ;;STRIP THE ASCII JUNK
062336 062601          ADD    (SP)+,R1     ;;ADD IN THIS DIGIT
062340 000764          BR     2$           ;;LOOP
062342 005726 3$:    TST    (SP)+       ;;CLEAN TERMINATOR FROM STACK
062344 010166 000012  MOV    R1,12(SP)    ;;SAVE THE RESULT
062350 010237 062364  MOV    R2,$HIOCT
062354 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
062356 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
062360 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
062362 000002          RTI
062364 000000  $HIOCT: .WORD    0    ;;RETURN
                                .SBTTL TRAP DECODER    ;;HIGH ORDER BITS GO HERE
    
```

10

```

*****
*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.
    
```

```

062366 016646 000002  $TRAP: MOV    2(SP),-(SP)    ;;ASSUME THE STATUS OF
062372 042716 000020  BIC    #20,(SP)        ;; THE CALLER--DO NOT ALLOW
062376 012746 062404  MOV    #1$,-(SP)      ;; T-BIT TRAPS
062402 000002          RTI          ;;SET THE NEW STATUS
062404 010046 3$:    MOV    R0,-(SP)    ;;SAVE R0
062406 016600 000002  MOV    2(SP),R0       ;;GET TRAP ADDRESS
062412 005740          TST    -(R0)         ;;BACKUP BY 2
062414 111000          MOV    (R0),R0        ;;GET RIGHT BYTE OF TRAP
062416 006300          ASL    R0            ;;POSITION FOR INDEXING
062420 016000 062440  MOV    $TRPAD(R0),R0  ;;INDEX TO TABLE
062424 000200          RTS    R0           ;;GO TO ROUTINE
    
```

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

```

062426 011646          $TRAP2: MOV    (SP),-(SP)    ;;MOVE THE PC DOWN
062430 016666 000004 000002  MOV    4(SP),2(SP)    ;;MOVE THE PSW DOWN
062436 000002          RTI          ;;RESTORE THE PSW
    
```

.SBTTL TRAP TABLE

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

```

: ROUTINE
: -----
062440 062426  $TRPAD: .WORD    $TRAP2
062442 057666  $TYPE   ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
062444 057464  $TYPOC  ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
062446 057440  $TYPOS  ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
062450 057500  $TYPON  ;;CALL=TYPON      TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
062452 057214  $TYPDS  ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
    
```

```

062454 057140          $TYPBN  ;;CALL=TYPBN   TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
062456 061414          $GTSWR  ;;CALL=GTSWR   TRAP+7(104407)  GET SOFT-SWR SETTING
062460 061324          $CKSWR  ;;CALL=CKSWR   TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
062462 061666          $RDCHR  ;;CALL=RDCHR   TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
062464 061756          $RDLIN  ;;CALL=RDLIN   TRAP+12(104412) TTY TYPEIN STRING ROUTINE
062466 062264          $RDOCT  ;;CALL=RDOCT   TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
062470 057044          $SAVREG ;;CALL=SAVREG   TRAP+14(104414) SAVE R0-R5 ROUTINE
062472 057102          $RESREG ;;CALL=RESREG   TRAP+15(104415) RESTORE R0-R5 ROUTINE
11      .SBTTL  POWER DOWN AND UP ROUTINES

```

```

*****
:POWER DOWN ROUTINE
062474 012737 062634 000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
062502 012737 000340 000026    MOV    #340,@#PWRVEC+2 ;;PRIO:7
062510 010046          MOV    R0,-(SP)        ;;PUSH R0 ON STACK
062512 010146          MOV    R1,-(SP)        ;;PUSH R1 ON STACK
062514 010246          MOV    R2,-(SP)        ;;PUSH R2 ON STACK
062516 010346          MOV    R3,-(SP)        ;;PUSH R3 ON STACK
062520 010446          MOV    R4,-(SP)        ;;PUSH R4 ON STACK
062522 010546          MOV    R5,-(SP)        ;;PUSH R5 ON STACK
062524 017746 116424          MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
062530 010637 062640          MOV    SP,$SAVR6     ;;SAVE SP
062534 012737 062546 000024    MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
062542 000000          HALT
062544 000776          BR      -2                ;;HANG UP

```

```

*****
:POWER UP ROUTINE
062546 012737 062634 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
062554 013706 062640          MOV    $SAVR6,SP      ;;GET SP
062560 005037 062640          CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
062564 005237 062640 1$:    INC    $SAVR6        ;;WAIT FOR THE INC
062570 001375          BNE    1$           ;;OF WORD
062572 012677 116356          MOV    (SP)+,@SWR    ;;POP STACK INTO @SWR
062576 012605          MOV    (SP)+,R5     ;;POP STACK INTO R5
062600 012604          MOV    (SP)+,R4     ;;POP STACK INTO R4
062602 012603          MOV    (SP)+,R3     ;;POP STACK INTO R3
062604 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
062606 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
062610 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
062612 012737 062474 000024    MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
062620 012737 000340 000026    MOV    #340,@#PWRVEC+2 ;;PRIO:7
062626 104401          TYPE    $POWER      ;;REPORT THE POWER FAILURE
062630 062642          $PWRMG: .WORD    $POWER ;;POWER FAIL MESSAGE POINTER
062632 000002          RTI
062634 000000          $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
062636 000776          BR      -2                ;; BEFORE THE POWER DOWN WAS COMPLETE
062640 000000          $SAVR6: 0           ;;PUT THE SP HERE
062642 015 012 120          $POWER: .ASCIZ <15><12>'POWER'
12      .SBTTL  APT COMMUNICATIONS ROUTINE

```

```

*****
062652 112737 000001 063116 $ATY1: MOV    #1,$FFLG ;;TO REPORT FATAL ERROR
062660 112737 000001 063114 $ATY3: MOV    #1,$MFLG ;;TO TYPE A MESSAGE

```

```

062666 000403
062670 112737 000001 063116 $ATY4: BR $ATYC
062676 $ATYC: MOV #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
062676 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
062700 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
062702 105737 063114 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
062706 001450 BEQ 5$ ;;IF NOT: BR
062710 122737 000001 001242 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
062716 001031 BNE 3$ ;;IF NOT: BR
062720 132737 000100 001243 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
062726 001425 BEQ 3$ ;;IF NOT: BR
062730 017600 000004 MOV @4(SP),RO ;;GET MESSAGE ADDR.
062734 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
062742 005737 001222 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
062746 001375 BNE 1$ ;;IF NOT: WAIT
062750 010037 001236 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
062754 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
062756 001376 BNE 2$
062760 163700 001236 SUB $MSGAD,RO ;;SUB START OF MESSAGE
062764 006200 ASR RO ;;GET MESSAGE LNGLTH IN WORDS
062766 010037 001240 MOV RO,$MSGLGT ;;PUT LENGTH IN MAILBOX
062772 012737 000004 001222 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
063000 000413 BR 5$
063002 017637 000004 063026 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
063010 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
063016 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
063022 004737 057666 JSR PC,$TYPE ;;CALL TYPE MACRO
063026 000000 4$: .WORD 0
063030 5$:
063030 105737 063116 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
063034 001416 BEQ 12$ ;;IF NOT: BR
063036 005737 001242 TST $ENV ;;RUNNING UNDER APT?
063042 001413 BEQ 12$ ;;IF NOT: BR
063044 005737 001222 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
063050 001375 BNE 11$ ;;IF NOT: WAIT
063052 017637 000004 001224 MOV @4(SP),$FATAL ;;GET ERROR #
063060 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
063066 005237 001222 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
063072 105037 063116 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
063076 105037 063115 CLRB $LFLG ;;CLEAR LOG FLAG
063102 105037 063114 CLRB $MFLG ;;CLEAR MESSAGE FLAG
063106 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
063110 012600 MOV (SP)+,RO ;;POP STACK INTO RO
063112 000207 RTS PC ;;RETURN
063114 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
063115 000 $LFLG: .BYTE 0 ;;LOG FLAG
063116 000 $FFLG: .BYTE 0 ;;FATAL FLAG
          .EVEN
          APTSIZE = 200
          APTENV = 001
          APTSPOOL = 100
          APTCSUP = 040
000200
000001
000100
000040

```

```

1          .SBTTL  CONSOLE MESSAGES
2
3 063120    200    106    101  SCTMSG: .ASCII  <CRLF>@FAILED TO RECOVER THE BAD SECTOR FILE(DEC 144)@
4 063177    200    117    116          .ASCIZ  <CRLF>@ON THIS DRIVE@
5 063216    075    000          EQUALS: .ASCIZ  @=@
6 063220    101    114    114  ALL:    .ASCIZ  @ALL@<CRLF>
7 063225    040    077    040  QUES:   .ASCIZ  @ ? @
8 063231    054    040    000  COMMA: .ASCIZ  @, @
9 063234    200    124    117  MSHELP: .ASCII  <CRLF>@TO ENSURE THAT NO BAD HEADERS ARE LEFT ON THE DISK@
10 063322   200    120    101          .ASCII  <CRLF>@PACK, THIS PROGRAM SHOULD BE HALTED BY TYPING A (^C)@
11 063410   200    103    117          .ASCII  <CRLF>@CONTROL C. AS A RESULT, THE PROGRAM WILL BE HALTED@
12 063476   200    127    110          .ASCII  <CRLF>@WHEN THE DRIVE UNDER TEST HAS COMPLETED TESTING.@<CRLF>
13 063560   200    124    131          .ASCIZ  <CRLF>@TYPE HELP TEXT (Y/N) ? @
14 063611
15 063611   200    103    110  UBUSQST: .ASCIZ  <CRLF>@CHANGE ADDRESSES (Y/N) ? @
16 063644   200    125    123  CNSL00: .ASCIZ  <CRLF>@USE SAME DEVICES (Y/N) ? @
17 063677   200    102    125  CNSL01: .ASCIZ  <CRLF>@BUS ADDRESS @
18 063715   040    114    111  CNSL02: .ASCIZ  @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
19 063757   126    105    103  CNSL03: .ASCIZ  @VECTOR ADDRESS @
20 063777   040    114    111  CNSL04: .ASCIZ  @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
21 064033   102    122    040  CNSL05: .ASCIZ  @BR LEVEL @
22 064045   040    114    111  CNSL06: .ASCIZ  @ LIMITS - LO= 0, HI= 7@<CRLF><LF>
23 064076   200          131  CNSL07: .ASCII  <CRLF>
24 064077   200    124    116          .ASCII  <CRLF>@TYPE "A" TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
25 064164   200    101          .ASCIZ  <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
26 064241   200          111  CNSL08: .ASCII  <CRLF>
27 064242   040    077    111  CNSL09: .ASCIZ  @ ?ILLEGAL INPUT@<CRLF>
28 064263   200    104    122  MSDRVS: .ASCIZ  <CRLF>/DRIVE(S): /
29 064277   104    122    111  MSGDRV: .ASCIZ  /DRIVE/
30 064305   040    111    123  LODEV:  .ASCIZ  / IS LOAD DEVICE/
31 064325   040    116    117  NOTPRS: .ASCIZ  / NOT PRESENT/
32 064342   040    116    117  NOTAVL: .ASCIZ  / NOT AVAILABLE/
33 064361   040    117    106  OFFLIN: .ASCIZ  / OFF LINE/
34 064373   040    117    116  ONLINE: .ASCIZ  / ON LINE /
35
36          .EVEN
37

```

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.

064406

064406 020000

FNCDTB: FUNCTION CODE TABLE

.WORD OPI :NOP

58	064410	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	064412	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	064414	130000	.WORD	ATA!OPI!IVC	:RECALIBRATE
61	064416	020000	.WORD	OPI	:DRIVE CLEAR
62	064420	030000	.WORD	OPI!IVC	:RELEASE
63	064422	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	064424	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	064426	020000	.WORD	OPI	:READ IN PRESET
66	064430	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	064432	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	064434	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	064436	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	064440	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	064442	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	064444	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	064446	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	064450	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	064452	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	064454	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	064456	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	064460	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	064462	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	064464	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	064466	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	064470	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	064472	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	064474	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	064476	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	064500	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	064502	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	064504	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)
89					

1		
2		
3	064506	001
4	064507	002
5	064510	004
6	064511	010
7	064512	020
8	064513	040
9	064514	100
10	064515	200
11		

.SBTTL ATTENTION (ATA) TABLE

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

Line	Address	Pattern	Category	Count
1				
2				
3	064516			
4	064516			
5	064516	000000	MIXED	0.
6	064520	000001	MIXED	1.
7	064522	000003	MIXED	3.
8	064524	000007	MIXED	7.
9	064526	000017	MIXED	15.
10	064530	000037	MIXED	31.
11	064532	000077	MIXED	63.
12	064534	000177	MIXED	127.
13	064536	000377	MIXED	255.
14	064540	000777	MIXED	511.
15	064542	001777	MIXED	1023.
16	064544	003777	MIXED	2047.
17	064546	007777	MIXED	4095.
18	064550	017777	MIXED	8191.
19	064552	037777	MIXED	16383.
20	064554	077777	MIXED	32767.
21	064556	177777	ONES	65535.
22	064560	177777	ONES	65535.
23	064562	077777	ONES	32767.
24	064564	037777	ONES	16383.
25	064566	017777	ONES	8191.
26	064570	007777	ONES	4095.
27	064572	003777	ONES	2047.
28	064574	001777	ONES	1023.
29	064576	000777	ONES	511.
30	064600	000377	ONES	255.
31	064602	000177	ONES	127.
32	064604	000077	ONES	63.
33	064606	000037	ONES	31.
34	064610	000017	ONES	15.
35	064612	000007	ONES	7.
36	064614	000003	ONES	3.
37	064616	000001	ONES	1.
38	064620	000000	ZEROS	0.
39	064622	000000	ZEROS	0.
40	064624	000001	ZEROS	1.
41	064626	000002	ZEROS	2.
42	064630	000004	ZEROS	4.
43	064632	000010	ZEROS	8.
44	064634	000020	ZEROS	16.
45	064636	000040	ZEROS	32.
46	064640	000100	ZEROS	64.
47	064642	000200	ZEROS	128.
48	064644	000400	ZEROS	256.
49	064646	001000	ZEROS	512.
50	064650	002000	ZEROS	1024.
51	064652	004000	ZEROS	2048.
52	064654	010000	ZEROS	4096.
53	064656	020000	ZEROS	8192.
54	064660	040000	ZEROS	16384.
55	064662	100000	ZEROS	32768.
56	064664	100000	ZEROS	32768.
57	064666	040000	ZEROS	16384.

Line No.	Keycode	Value	Label	Value
58	064670	020000	.WORD	8192.
59	064672	010000	.WORD	4096.
60	064674	004000	.WORD	2048.
61	064676	002000	.WORD	1024.
62	064700	001000	.WORD	512.
63	064702	000400	.WORD	256.
64	064704	000200	.WORD	128.
65	064706	000100	.WORD	64.
66	064710	000040	.WORD	32.
67	064712	000020	.WORD	16.
68	064714	000010	.WORD	8.
69	064716	000004	.WORD	4.
70	064720	000002	.WORD	2.
71	064722	000001	.WORD	1.
72	064724	000000	.WORD	0.
73	064726	177777	.WORD	65535.
74	064730	177776	.WORD	65534.
75	064732	177774	.WORD	65532.
76	064734	177770	.WORD	65528.
77	064736	177760	.WORD	65520.
78	064740	177740	.WORD	65504.
79	064742	177700	.WORD	65472.
80	064744	177600	.WORD	65408.
81	064746	177400	.WORD	65280.
82	064750	177000	.WORD	65024.
83	064752	176000	.WORD	64512.
84	064754	174000	.WORD	63488.
85	064756	170000	.WORD	61440.
86	064760	160000	.WORD	57344.
87	064762	140000	.WORD	49152.
88	064764	100000	.WORD	32768.
89	064766	000000	.WORD	0.
90	064770	000000	.WORD	0.
91	064772	100000	.WORD	32768.
92	064774	140000	.WORD	49152.
93	064776	160000	.WORD	57344.
94	065000	170000	.WORD	61440.
95	065002	174000	.WORD	63488.
96	065004	176000	.WORD	64512.
97	065006	177000	.WORD	65024.
98	065010	177400	.WORD	65280.
99	065012	177600	.WORD	65408.
100	065014	177700	.WORD	65472.
101	065016	177740	.WORD	65504.
102	065020	177760	.WORD	65520.
103	065022	177770	.WORD	65528.
104	065024	177774	.WORD	65532.
105	065026	177776	.WORD	65534.
106	065030	177777	.WORD	65535.
107	065032	125252	.WORD	43690.
108	065034	152525	.WORD	43690./2
109	065036	125252	.WORD	43690.
110	065040	177777	.WORD	65535.
111	065042	177776	.WORD	65534.
112	065044	177775	.WORD	65533.
113	065046	177773	.WORD	65531.
114	065050	177767	.WORD	65527.

EARLY:

115	065052	177757	.WORD	65519.
116	065054	177737	.WORD	65503.
117	065056	177677	.WORD	65471.
118	065060	177577	.WORD	65407.
119	065062	177377	.WORD	65279.
120	065064	176777	.WORD	65023.
121	065066	175777	.WORD	64511.
122	065070	173777	.WORD	63487.
123	065072	167777	.WORD	61439.
124	065074	157777	.WORD	57343.
125	065076	137777	.WORD	49151.
126	065100	077777	.WORD	32767.
127	065102	077777	.WORD	32767.
128	065104	137777	.WORD	49151.
129	065106	157777	.WORD	57343.
130	065110	167777	.WORD	61439.
131	065112	173777	.WORD	63487.
132	065114	175777	.WORD	64511.
133	065116	176777	.WORD	65023.
134	065120	177377	.WORD	65279.
135	065122	177577	.WORD	65407.
136	065124	177677	.WORD	65471.
137	065126	177737	.WORD	65503.
138	065130	177757	.WORD	65519.
139	065132	177767	.WORD	65527.
140	065134	177773	.WORD	65531.
141	065136	177775	.WORD	65533.
142	065140	177776	.WORD	65534.
143	065142	177777	.WORD	65535.
144	065144			
145				

ENRGDT:

				.SBTTL ERROR MESSAGE TABLE		
1						
2						
3	065144	071540	000000	EMT1:	.WORD	EMS1,0
4	065150	071607	071624	EMT2:	.WORD	EMS2,EMS3,0
5	065156	071607	071667	EMT3:	.WORD	EMS2,EMS4,0
6	065164	071732	071762	EMT4:	.WORD	EMS5,EMS6,0
7	065172	071732	072074	EMT5:	.WORD	EMS5,EMS10,0
8	065200	076535	073753	EMT6:	.WORD	EMS167,EMS64,0
9	065206	074521	076562	EMT7:	.WORD	EMS110,EMS170,0
10	065214	072027	000000	EMT10:	.WORD	EMS7,0
11	065220	072074	000000	EMT11:	.WORD	EMS10,0
12	065224	072136	072147	EMT12:	.WORD	EMS11,EMS12,0
13	065232	072210	072221	EMT13:	.WORD	EMS13,EMS14,EMS15,EMS16,0
14	065244	072304	073753	EMT14:	.WORD	EMS17,EMS64,0
15	065252	072136	072367	EMT15:	.WORD	EMS11,EMS21,0
16	065260	072136	072412	EMT16:	.WORD	EMS11,EMS22,EMS27,0
17	065270	072136	072426	EMT17:	.WORD	EMS11,EMS23,EMS30,0
18	065300	072136	072454	EMT20:	.WORD	EMS11,EMS24,EMS30,0
19	065310	072136	072503	EMT21:	.WORD	EMS11,EMS25,EMS27,0
20	065320	072136	072520	EMT22:	.WORD	EMS11,EMS26,EMS27,0
21	065330	072136	072562	EMT23:	.WORD	EMS11,EMS31,EMS30,0
22	065340	072136	072611	EMT24:	.WORD	EMS11,EMS32,EMS30,0
23	065350	072136	072640	EMT25:	.WORD	EMS11,EMS33,EMS30,0
24	065360	072136	072666	EMT26:	.WORD	EMS11,EMS34,EMS30,0
25	065370	072136	072737	EMT27:	.WORD	EMS11,EMS35,EMS30,0
26	065400	072136	072766	EMT30:	.WORD	EMS11,EMS36,EMS30,0
27	065410	072136	073015	EMT31:	.WORD	EMS11,EMS37,EMS30,0
28	065420	072136	073043	EMT32:	.WORD	EMS11,EMS40,EMS30,0
29	065430	072136	073072	EMT33:	.WORD	EMS11,EMS41,EMS30,0
30	065440	072136	073120	EMT34:	.WORD	EMS11,EMS42,EMS30,0
31	065450	072136	073147	EMT35:	.WORD	EMS11,EMS43,EMS30,0
32	065460	072136	073176	EMT36:	.WORD	EMS11,EMS44,EMS30,0
33	065470	072136	073251	EMT37:	.WORD	EMS11,EMS45,EMS30,0
34	065500	074037	072344	EMT40:	.WORD	EMS66,EMS20,0
35	065506	074225	075736	EMT41:	.WORD	EMS75,EMS141,EMS76,0
36	065516	076027	076037	EMT42:	.WORD	EMS144,EMS145,EMS72,EMS76,0
37	065530	073343	073457	EMT43:	.WORD	EMS47,EMS53,EMS76,0
38	065540	074264	073457	EMT44:	.WORD	EMS77,EMS53,EMS76,0
39	065550	074312	073457	EMT45:	.WORD	EMS100,EMS53,EMS76,0
40	065560	074340	073457	EMT46:	.WORD	EMS101,EMS53,EMS76,0
41	065570	073771	073753	EMT47:	.WORD	EMS65,EMS64,0
42	065576	072210	072232	EMT50:	.WORD	EMS13,EMS15,EMS63,0
43	065606	072136	073314	EMT51:	.WORD	EMS11,EMS46,EMS30,EMS67,0
44	065620	073343	073457	EMT52:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
45	065636	073343	073457	EMT53:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
46	065654	073372	073457	EMT54:	.WORD	EMS50,EMS53,EMS67,0
47	065664	075764	076001	EMT55:	.WORD	EMS142,EMS143,EMS53,EMS67,0
48	065676	073421	074161	EMT56:	.WORD	EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
49	065714	076535	074171	EMT57:	.WORD	EMS167,EMS73,EMS67,0
50	065724	073542	074107	EMT60:	.WORD	EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
51	065742	074146	073542	EMT61:	.WORD	EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
52	065762	075715	074107	EMT62:	.WORD	EMS140,EMS67,EMS115,EMS47,EMS70,0
53	065776	000000		EMT63:	.WORD	
54	066000	076122	076165	EMT64:	.WORD	EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
55	066020	073447	077162	EMT65:	.WORD	EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
56	066040	076474	074415	EMT66:	.WORD	EMS165,EMS103,EMS72,EMS124,0
57	066052	076474	074415	EMT67:	.WORD	EMS165,EMS103,EMS72,EMS171,0

58	066064	073314	072344	076367	EMT70:	.WORD	EMS46,EMS20,EMS163,0
59	066074	074146	074340	076367	EMT71:	.WORD	EMS71,EMS101,EMS163,0
60	066104	073343	074161	076367	EMT72:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS140,EMS141,0
61	066122	073343	074161	076367	EMT73:	.WORD	EMS47,EMS72,EMS163,EMS115,EMS141,EMS72,0
62	066140	073542	073457	076367	EMT74:	.WORD	EMS56,EMS53,EMS163,0
63	066150	074146	073542	076367	EMT75:	.WORD	EMS71,EMS56,EMS163,EMS115,EMS150,EMS152,EMS72,0
64	066170	073372	073457	076367	EMT76:	.WORD	EMS50,EMS53,EMS163,0
65	066200	075764	076001	073457	EMT77:	.WORD	EMS142,EMS143,EMS53,EMS163,0
66	066212	074225	075736	076367	EMT100:	.WORD	EMS75,EMS141,EMS163,EMS115,EMS47,EMS70,0
67	066230	074225	076122	076367	EMT101:	.WORD	EMS75,EMS150,EMS163,EMS115,EMS56,EMS73,0
68	066246	076535	074171	076367	EMT102:	.WORD	EMS167,EMS73,EMS163,0
69	066256	076105	076141	074206	EMT103:	.WORD	EMS147,EMS151,EMS74,EMS163,0
70	066270	073421	074206	076367	EMT104:	.WORD	EMS51,EMS74,EMS163,EMS115,EMS50,EMS70,0
71	066306	076474	074312	076367	EMT105:	.WORD	EMS165,EMS100,EMS163,0
72	066316	076474	074264	076367	EMT106:	.WORD	EMS165,EMS77,EMS163,0
73	066326	076027	076037	073457	EMT107:	.WORD	EMS144,EMS145,EMS53,EMS163,EMS115,EMS143,EMS70,0
74	066346	074521	074561	074724	EMT110:	.WORD	EMS110,EMS112,EMS116,EMS111,0
75	066360	074645	071667	000000	EMT111:	.WORD	EMS113,EMS4,0
76	066366	074645	071624	000000	EMT112:	.WORD	EMS113,EMS3,0
77	066374	074521	074721	074724	EMT113:	.WORD	EMS110,EMS115,EMS116,EMS117,EMS114,0
78	066410	074645	074776	000000	EMT114:	.WORD	EMS113,EMS120,0
79	066416	075017	075457	075503	EMT115:	.WORD	EMS121,EMS132,EMS133,0
80	066426	075062	075457	075503	EMT116:	.WORD	EMS122,EMS132,EMS133,0
81	066436	075117	075457	075503	EMT117:	.WORD	EMS123,EMS132,EMS133,0
82	066446	075162	075457	075503	EMT120:	.WORD	EMS124,EMS132,EMS133,0
83	066456	075214	075457	075503	EMT121:	.WORD	EMS125,EMS132,EMS133,0
84	066466	075257	075457	075503	EMT122:	.WORD	EMS126,EMS132,EMS133,0
85	066476	075660	075457	075503	EMT123:	.WORD	EMS137,EMS132,EMS133,0
86	066506	075361	075457	075503	EMT124:	.WORD	EMS130,EMS132,EMS133,0
87	066516	075417	075457	075503	EMT125:	.WORD	EMS131,EMS132,EMS133,0
88	066526	075017	075457	075526	EMT126:	.WORD	EMS121,EMS132,EMS134,EMS123,0
89	066540	075062	075457	075526	EMT127:	.WORD	EMS122,EMS132,EMS134,EMS123,0
90	066552	075117	075457	075526	EMT130:	.WORD	EMS123,EMS132,EMS134,EMS123,0
91	066564	075162	075457	075526	EMT131:	.WORD	EMS124,EMS132,EMS134,EMS123,0
92	066576	075214	075457	075526	EMT132:	.WORD	EMS125,EMS132,EMS134,EMS123,0
93	066610	075257	075457	075526	EMT133:	.WORD	EMS126,EMS132,EMS134,EMS123,0
94	066622	075660	075457	075526	EMT134:	.WORD	EMS137,EMS132,EMS134,EMS123,0
95	066634	075361	075457	075526	EMT135:	.WORD	EMS130,EMS132,EMS134,EMS123,0
96	066646	075417	075457	075526	EMT136:	.WORD	EMS131,EMS132,EMS134,EMS123,0
97	066660	075017	075457	075570	EMT137:	.WORD	EMS121,EMS132,EMS135,0
98	066670	075117	075457	075570	EMT140:	.WORD	EMS123,EMS132,EMS135,0
99	066700	075017	075457	075633	EMT141:	.WORD	EMS121,EMS132,EMS136,0
100	066710	075660	075457	075633	EMT142:	.WORD	EMS137,EMS132,EMS136,0
101	066720	075162	075457	075633	EMT143:	.WORD	EMS124,EMS132,EMS136,0
102	066730	075214	075457	075633	EMT144:	.WORD	EMS125,EMS132,EMS136,0
103	066740	075257	075457	075633	EMT145:	.WORD	EMS126,EMS132,EMS136,0
104	066750	075417	075457	075633	EMT146:	.WORD	EMS131,EMS132,EMS136,0
105	066760	076611	075457	075633	EMT147:	.WORD	EMS171,EMS132,EMS136,0
106	066770	075361	075457	075633	EMT150:	.WORD	EMS130,EMS132,EMS136,0
107	067000	075715	074721	075736	EMT151:	.WORD	EMS140,EMS115,EMS141,EMS70,0
108	067012	075764	074721	076001	EMT152:	.WORD	EMS142,EMS115,EMS143,EMS72,0
109	067024	076027	076037	076066	EMT153:	.WORD	EMS144,EMS145,EMS146,EMS115,EMS143,EMS72,0
110	067042	076027	076037	072344	EMT154:	.WORD	EMS144,EMS145,EMS20,EMS115,EMS143,EMS70,0
111	067060	076122	076165	076233	EMT155:	.WORD	EMS150,EMS152,EMS154,EMS153,0
112	067072	076105	076141	076233	EMT156:	.WORD	EMS147,EMS151,EMS154,EMS155,0
113	067104	076105	076141	076267	EMT157:	.WORD	EMS147,EMS151,EMS156,EMS157,0
114	067116	076337	076267	076322	EMT160:	.WORD	EMS161,EMS156,EMS160,0

115	067126	072503	072541	076267	EMT161:	.WORD	EMS25,EMS27,EMS156,EMS160,0
116	067140	072520	072541	076267	EMT162:	.WORD	EMS26,EMS27,EMS156,EMS160,0
117	067152	073343	076367	075715	EMT163:	.WORD	EMS47,EMS163,EMS140,0
118	067162	073343	072344	000000	EMT164:	.WORD	EMS47,EMS20,0
119	067170	073542	072344	000000	EMT165:	.WORD	EMS56,EMS20,0
120	067176	073314	072344	000000	EMT166:	.WORD	EMS46,EMS20,0
121	067204	077663	072344	000000	EMT167:	.WORD	EMS224,EMS20,0
122	067212	076535	076233	076510	EMT170:	.WORD	EMS167,EMS154,EMS166,0
123	067222	076535	076233	076305	EMT171:	.WORD	EMS167,EMS154,EMS157,0
124	067232	076535	076233	076247	EMT172:	.WORD	EMS167,EMS154,EMS155,0
125	067242	076611	075457	075503	EMT173:	.WORD	EMS171,EMS132,EMS133,0
126	067252	076611	075457	075526	EMT174:	.WORD	EMS171,EMS132,EMS134,EMS123,0
127	067264	074645	076764	000000	EMT175:	.WORD	EMS113,EMS177,0
128	067272	077006	077023	000000	EMT176:	.WORD	EMS200,EMS201,0
129	067300	077121	075736	072552	EMT177:	.WORD	EMS203,EMS141,EMS30,EMS202,0
130	067312	077121	073421	072552	EMT200:	.WORD	EMS203,EMS51,EMS30,EMS202,0
131	067324	077121	077134	072552	EMT201:	.WORD	EMS203,EMS204,EMS30,EMS202,0
132	067336	077121	073372	072552	EMT202:	.WORD	EMS203,EMS50,EMS30,EMS202,0
133	067350	077121	076001	072552	EMT203:	.WORD	EMS203,EMS143,EMS30,EMS202,0
134	067362	076122	074206	077071	EMT204:	.WORD	EMS150,EMS74,EMS202,EMS115,EMS152,EMS72,0
135	067400	073372	074415	074161	EMT205:	.WORD	EMS50,EMS103,EMS72,0
136	067410	074424	074171	077071	EMT206:	.WORD	EMS104,EMS73,EMS202,0
137	067420	074225	075736	074721	EMT207:	.WORD	EMS75,EMS141,EMS115,EMS140,0
138	067432	074225	076122	000000	EMT210:	.WORD	EMS75,EMS150,0
139	067440	073421	074161	074721	EMT211:	.WORD	EMS51,EMS72,EMS115,EMS50,EMS70,0
140	067454	075764	073457	074721	EMT212:	.WORD	EMS142,EMS53,EMS115,EMS143,EMS72,0
141	067470	073372	074415	073457	EMT213:	.WORD	EMS50,EMS103,EMS53,0
142	067500	073447	077162	076510	EMT214:	.WORD	EMS52,EMS205,EMS166,EMS206,EMS115,EMS51,EMS72,0
143	067520	073447	074754	073542	EMT215:	.WORD	EMS52,EMS117,EMS56,EMS163,0
144	067532	073542	072552	073753	EMT216:	.WORD	EMS56,EMS30,EMS64,0
145	067542	073314	072552	073753	EMT217:	.WORD	EMS46,EMS30,EMS64,0
146	067552	072562	072552	073753	EMT220:	.WORD	EMS31,EMS30,EMS64,0
147	067562	073343	072552	073753	EMT221:	.WORD	EMS47,EMS30,EMS64,0
148	067572	073447	074754	074264	EMT222:	.WORD	EMS52,EMS117,EMS77,0
149	067602	073447	074754	073725	EMT223:	.WORD	EMS52,EMS117,EMS63,0
150	067612	000000			EMT224:	.WORD	
151	067614	000000			EMT225:	.WORD	
152	067616	000000			EMT226:	.WORD	
153	067620	000000			EMT227:	.WORD	
154	067622	000000			EMT230:	.WORD	
155	067624	000000			EMT231:	.WORD	
156	067626	000000			EMT232:	.WORD	
157	067630	000000			EMT233:	.WORD	
158	067632	000000			EMT234:	.WORD	
159	067634	000000			EMT235:	.WORD	
160	067636	000000			EMT236:	.WORD	
161	067640	000000			EMT237:	.WORD	
162	067642	000000			EMT240:	.WORD	
163	067644	000000			EMT241:	.WORD	
164	067646	000000			EMT242:	.WORD	
165	067650	000000			EMT243:	.WORD	
166	067652	000000			EMT244:	.WORD	
167	067654	000000			EMT245:	.WORD	
168	067656	076535	075457	077221	EMT246:	.WORD	EMS167,EMS132,EMS207,0
169	067666	076535	075457	077246	EMT247:	.WORD	EMS167,EMS132,EMS210,EMS125,0
170	067700	076535	077257	077246	EMT250:	.WORD	EMS167,EMS211,EMS210,EMS207,EMS206,0
171	067714	077275	077320	000000	EMT251:	.WORD	EMS212,EMS213,0

229	071104	073447	074754	073251	EMT343: .WORD	EMS52,EMS117,EMS45,EMS57,0
230	071116	073447	074754	073176	EMT344: .WORD	EMS52,EMS117,EMS44,EMS57,0
231	071130	073447	074754	077505	EMT345: .WORD	EMS52,EMS117,EMS221,0
232	071140	074502	072344	074721	EMT346: .WORD	EMS106,EMS20,EMS115,EMS223,EMS72,0
233	071154	073447	077162	077555	EMT347: .WORD	EMS52,EMS205,EMS222,EMS206,0
234	071166	074225	076122	074366	EMT350: .WORD	EMS75,EMS150,EMS102,EMS115,EMS56,EMS73,0
235	071204	076535	074171	074366	EMT351: .WORD	EMS167,EMS73,EMS102,0
236	071214	077361	000000		EMT352: .WORD	EMS215,0
237	071220	077432	077121	077402	EMT353: .WORD	EMS217,EMS203,EMS216,0
238	071230	077505	072344	000000	EMT354: .WORD	EMS221,EMS20,0
239						

1	071236	077735	100541	100616	EHT1:	.WORD	EH1,STSH1,STSH2,STSH4,0
2	071250	100541	100616	100743	EHT2:	.WORD	STSH1,STSH2,STSH4,0
3							
4	071260	077754	000000		EHT110:	.WORD	EH110,0
5	071264	077763	000000		EHT111:	.WORD	EH111,0
6							
7	071270	100002	000000		EHT114:	.WORD	EH114,0
8	071274	100031	100541	100616	EHT223:	.WORD	EH223,STSH1,STSH2,STSH4,0
9	071306	100057	100541	100616	EHT256:	.WORD	EH256,STSH1,STSH2,STSH4,0
10							
11	071320	100133	100541	100616	EHT336:	.WORD	EH336,STSH1,STSH2,STSH4,0
12	071332	100172	100541	100616	EHT337:	.WORD	EH337,STSH1,STSH2,STSH4,0
13	071344	100327	100541	100616	EHT344:	.WORD	EH344,STSH1,STSH2,STSH4,0
14							
15	071356	100465	000000		EHT353:	.WORD	EH353,0
16							

1	071362	101002	101076	101114	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
2	071372	101076	101114	101146	EDT2:	.WORD	STSD1,STSD2,STSD4
3							
4	071400	101010			EDT110:	.WORD	ED110
5	071402	101014			EDT111:	.WORD	ED111
6							
7	071404	101022			EDT114:	.WORD	ED114
8	071406	101032	101076	101114	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
9							
10	071416	101042	101076	101114	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
11	071426	101054	101076	101114	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
12	071436	101054	101076	101114	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
13							
14	071450	101066			EDT353:	.WORD	ED353
15							

1	071452	101161	101177	101177	EFT1:	.WORD	EF111,STSF,STSF,STSF
2	071462	101177	101177	101177	EFT2:	.WORD	STSF,STSF,STSF
3							
4	071470	101160			EFT110:	.WORD	EF110
5	071472	101161			EFT111:	.WORD	EF111
6							
7	071474	101163			EFT114:	.WORD	EF114
8	071476	101163	101177	101177	EFT223:	.WORD	EF114,STSF,STSF,STSF
9							
10	071506	101166	101177	101177	EFT336:	.WORD	EF336,STSF,STSF,STSF
11	071516	101166	101177	101177	EFT337:	.WORD	EF336,STSF,STSF,STSF
12	071526	101166	101177	101177	EFT344:	.WORD	EF336,STSF,STSF,STSF
13							
14	071536	101163			EFT353:	.WORD	EF114
15							

		.SBTTL ERROR MESSAGE STRINGS	
1			
2			
3	071540	127	122
4	071607	104	105
5	071624	125	116
6	071667	116	117
7	071732	103	117
8	071762	103	117
9	072027	104	122
10	072074	107	117
11	072136	111	116
12	072147	106	125
13	072210	115	101
14	072221	103	117
15	072232	102	125
16	072254	042	115
17	072304	124	122
18	072344	123	110
19	072367	127	117
20	072412	102	125
21	072426	042	114
22	072454	042	101
23	072503	104	111
24	072520	103	131
25	072541	101	104
26	072552	123	124
27	072562	042	127
28	072611	042	125
29	072640	042	127
30	072666	127	122
31	072737	042	115
32	072766	042	104
33	073015	042	105
34	073043	042	104
35	073072	042	115
36	073120	042	104
37	073147	042	110
38	073176	110	105
39	073251	106	117
40	073314	042	111
41	073343	042	117
42	073372	042	123
43	073421	042	120
44	073447	124	110
45	073457	104	105
46	073471	101	124
47	073513	111	116
48	073542	111	116
49	073617	104	125
50	073645	104	101
51	073660	104	117
52	073707	104	101
53	073725	042	120
54	073753	111	123
55	073771	103	117
56	074037	104	101
57	074107	104	125
			117 EMS1: .ASCIZ @WRONG UNIT SELECTED (RMCS2, BITS 0-2) @
			126 EMS2: .ASCIZ @DEVICE WENT @
			101 EMS3: .ASCIZ @UNAVAILABLE "DVA" (RMCS1, BIT 11) @
			116 EMS4: .ASCIZ @NONEXISTENT "NED" (RMCS2, BIT 12) @
			115 EMS5: .ASCIZ @COMMAND NOT COMPLETED, @
			116 EMS6: .ASCIZ @CONTROLLER NOT READY (RMCS1, BIT 7) @
			111 EMS7: .ASCIZ @DRIVE NOT READY "DRY" (RMDS, BIT 7) @
			040 EMS10: .ASCIZ @GO NOT RESET "GO" (RMCS1, BIT 0) @
			126 EMS11: .ASCIZ @INVALID @
			116 EMS12: .ASCIZ @FUNCTION CODE (RMCS1, BITS 1-5) @
			123 EMS13: .ASCIZ @MASSBUS @
			116 EMS14: .ASCIZ @CONTROL @
			123 EMS15: .ASCIZ @BUS PARITY ERROR @
			103 EMS16: .ASCIZ @"MCPE" (RMCS1, BIT 13) @
			101 EMS17: .ASCIZ @TRANSFER ERROR (RMCS1, BIT 14) @
			117 EMS20: .ASCIZ @SHOULD NOT BE SET @
			122 EMS21: .ASCIZ @WORD COUNT (RMWC) @
			123 EMS22: .ASCIZ @BUS (RMBA) @
			102 EMS23: .ASCIZ @"LBT" (RMDS, BIT 10) @
			117 EMS24: .ASCIZ @"AOE" (RMER1, BIT 09) @
			123 EMS25: .ASCIZ @DISK (RMDA) @
			114 EMS26: .ASCIZ @CYLINDER (RMDC) @
			104 EMS27: .ASCIZ @ADDRESS @
			101 EMS30: .ASCIZ @STATUS @
			114 EMS31: .ASCIZ @"WLE" (RMER1, BIT 11) @
			120 EMS32: .ASCIZ @"UPE" (RMCS2, BIT 13) @
			103 EMS33: .ASCIZ @"WCF" (RMER1, BIT 5) @
			111 EMS34: .ASCIZ @WRITE CHECK ERROR-"WCE" (RMCS2, BIT 14) @
			104 EMS35: .ASCIZ @"MDPE" (RMCS2, BIT 8) @
			103 EMS36: .ASCIZ @"DCK" (RMER1, BIT 15) @
			114 EMS37: .ASCIZ @"ECH" (RMER1, BIT 6) @
			130 EMS40: .ASCIZ @"DLT" (RMCS2, BIT 15) @
			124 EMS41: .ASCIZ @"MXF" (RMCS2, BIT 9) @
			103 EMS42: .ASCIZ @"DTE" (RMER1, BIT 12) @
			101 EMS43: .ASCIZ @"HCRC" (RMER1, BIT 8) @
			122 EMS44: .ASCIZ @HEADER COMPARE ERROR "HCE" (RMER1, BIT 7) @
			101 EMS45: .ASCIZ @FORMAT ERROR "FER" (RMER1, BIT 4) @
			120 EMS46: .ASCIZ @"IAE" (RMER1, BIT 10) @
			113 EMS47: .ASCIZ @"OPI" (RMER1, BIT 13) @
			111 EMS50: .ASCIZ @"SKI" (RMER2, BIT 14) @
			105 EMS51: .ASCIZ @"PIP" (RMDS, BIT 13) @
			124 EMS52: .ASCIZ @THE RM @
			040 EMS53: .ASCIZ @DETECTED @
			103 EMS54: .ASCIZ @AT AN UNEXPECTED @
			126 EMS55: .ASCIZ @INCORRECT DATA DURING @
			122 EMS56: .ASCIZ @INVALID COMMAND ERROR "IVC" (RMER2, BIT 12) @
			124 EMS57: .ASCIZ @DURING DATA TRANSFER @
			105 EMS60: .ASCIZ @DATA READ @
			124 EMS61: .ASCIZ @DOES NOT COMPARE WITH @
			101 EMS62: .ASCIZ @DATA WRITTEN @
			040 EMS63: .ASCIZ @"PAR" (RMER1, BIT 3) @
			115 EMS64: .ASCIZ @IS INCORRECT @
			124 EMS65: .ASCIZ @COMPOSITE ERROR "ERR" (RMDS, BIT 14) @
			124 EMS66: .ASCIZ @DATA PARITY ERROR "DPE" (RMER2, BIT 3) @
			122 EMS67: .ASCIZ @DURING SEEK COMMAND @

58	074134	111	123	040	EMS70:	.ASCIZ	@IS RESET @
59	074146	105	122	122	EMS71:	.ASCIZ	@ERRONEOUS @
60	074161	111	123	040	EMS72:	.ASCIZ	@IS SET @
61	074171	104	111	104	EMS73:	.ASCIZ	@DID NOT SET @
62	074206	104	111	104	EMS74:	.ASCIZ	@DID NOT RESET @
63	074225	114	117	123	EMS75:	.ASCIZ	@LOST @
64	074233	104	125	122	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
65	074264	042	122	115	EMS77:	.ASCIZ	@'RMR' (RMER1, BIT 2) @
66	074312	042	111	114	EMS100:	.ASCIZ	@'ILR' (RMER1, BIT 1) @
67	074340	042	111	114	EMS101:	.ASCIZ	@'ILF' (RMER1, BIT 0) @
68	074366	104	125	122	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
69	074415	105	122	122	EMS103:	.ASCIZ	@ERROR @
70	074424	042	114	102	EMS104:	.ASCIZ	@'LBC' (RMER2, BIT 10) @
71	074453	042	114	123	EMS105:	.ASCIZ	@'LSC' (RMER2, BIT 11) @
72	074502	110	105	101	EMS106:	.ASCIZ	@HEADER ERRORS @
73	074521	102	125	123	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
74	074550	101	104	104	EMS111:	.ASCIZ	@ADDRESS @
75	074561	127	110	105	EMS112:	.ASCII	@WHEN READING/WRITING RM REGISTERS @
76	074623	101	124	040		.ASCIZ	@AT THE FOLLOWING @
77	074645	124	110	105	EMS113:	.ASCIZ	@THE SELECTED DEVICE IS @
78	074675	116	117	116	EMS114:	.ASCIZ	@NONEXISTENT DEVICE @
79	074721	015	012	000	EMS115:	.ASCIZ	<CR><LF>
80	074724	124	110	105	EMS116:	.ASCIZ	@THE MASSBUS CONTROLLER @
81	074754	106	101	111	EMS117:	.ASCIZ	@FAILED TO DETECT @
82	074776	116	117	124	EMS120:	.ASCIZ	@NOT AN RM05/3/2 @
83	075017	103	117	116	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
84	075062	102	125	123	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMBA, @
85	075117	103	117	116	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
86	075162	105	122	122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMER1, @
87	075214	101	124	124	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMAS, @
88	075257	115	101	111	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMMR #1, @
89	075322	105	103	103	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
90	075361	105	103	103	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2, @
91	075417	115	101	111	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMMR2, @
92	075457	116	117	124	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
93	075503	125	116	111	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
94	075526	103	117	116	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
95	075570	122	110	057	EMS135:	.ASCIZ	@RM/RM ERROR CLEAR (RMCS1, BIT 14) @
96	075633	104	122	111	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
97	075660	104	122	111	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMDS @
98	075715	115	105	104	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
99	075736	042	115	117	EMS141:	.ASCIZ	@'MOL' (RMDS, BIT 12) @
100	075764	104	122	111	EMS142:	.ASCIZ	@DRIVE FAULT @
101	076001	042	104	126	EMS143:	.ASCIZ	@'DVC' (RMER2, BIT 7) @
102	076027	125	116	123	EMS144:	.ASCIZ	@UNSAFE @
103	076037	042	125	116	EMS145:	.ASCIZ	@'UNS' (RMER1, BIT 14) @
104	076066	123	110	117	EMS146:	.ASCIZ	@SHOULD BE SET @
105	076105	117	106	106	EMS147:	.ASCIZ	@OFFSET MODE @
106	076122	040	126	117	EMS150:	.ASCIZ	@ VOLUME VALID @
107	076141	042	117	115	EMS151:	.ASCIZ	@'OM' (RMDS, BIT 0) @
108	076165	042	126	126	EMS152:	.ASCIZ	@'VV' (RMDS, BIT 6) @
109	076211	120	101	103	EMS153:	.ASCIZ	@PACK ACK COMMAND @
110	076233	116	117	124	EMS154:	.ASCIZ	@NOT SET BY @
111	076247	117	106	106	EMS155:	.ASCIZ	@OFFSET COMMAND @
112	076267	116	117	124	EMS156:	.ASCIZ	@NOT RESET BY @
113	076305	122	124	103	EMS157:	.ASCIZ	@RTC COMMAND @
114	076322	122	111	120	EMS160:	.ASCIZ	@RIP COMMAND @

115	076337	117	106	106	EMS161: .ASCIZ	@OFFSET REGISTER (RMOF) @
116	076367				EMS162:	;<UNUSED>
117	076367	104	125	122	EMS163: .ASCIZ	@DURING RECALIBRATE @
118	076413	111	123	040	EMS164: .ASCII	@IS INTERMITTENT OR DRIVE DIDNT DROP ON @
119	076462	103	131	114	.ASCIZ	@CYLINDER @
120	076474	125	116	105	EMS165: .ASCIZ	@UNEXPECTED @
121	076510	122	105	103	EMS166: .ASCIZ	@RECALIBRATE COMMAND @
122	076535	042	101	124	EMS167: .ASCIZ	@'ATA' (RMDS, BIT15) @
123	076562	127	110	105	EMS170: .ASCIZ	@WHEN READING REGISTER @
124	076611	105	122	122	EMS171: .ASCIZ	@ERROR REGISTER #2, RMER2, @
125	076644	116	117	116	EMS172: .ASCIZ	@NONRECOVERABLE @
126	076664	122	105	103	EMS173: .ASCIZ	@RECOVERABLE @
127	076701	104	101	124	EMS174: .ASCIZ	@DATA @
128	076707	104	125	122	EMS175: .ASCIZ	@DURING WRITE COMMAND @
129	076735	042	117	120	EMS176: .ASCIZ	@'OPE' (RMER2, BIT 13) @
130	076764	111	116	040	EMS177: .ASCIZ	@IN WRITE PROTECT @
131	077006	103	101	116	EMS200: .ASCIZ	@CAN NOT SET @
132	077023	104	111	101	EMS201: .ASCIZ	@DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 0) @
133	077071	104	125	122	EMS202: .ASCIZ	@DURING DIAGNOSTIC MODE @
134	077121	111	116	103	EMS203: .ASCIZ	@INCORRECT @
135	077134	042	127	122	EMS204: .ASCIZ	@'WRL' (RMDS, BIT 11) @
136	077162	105	130	105	EMS205: .ASCIZ	@EXECUTED @
137	077174	127	111	124	EMS206: .ASCIZ	@WITH COMP ERROR SET @
138	077221	042	107	117	EMS207: .ASCIZ	@'GO' (RMCS1, BIT 0) @
139	077246	127	122	111	EMS210: .ASCIZ	@WRITING @
140	077257	127	101	123	EMS211: .ASCIZ	@WAS RESET BY @
141	077275	120	122	117	EMS212: .ASCIZ	@PROGRAM INTERRUPT @
142	077320	127	101	123	EMS213: .ASCIZ	@WAS NOT GENERATED @
143	077343	123	105	105	EMS214: .ASCIZ	@SEEK COMMAND @
144	077361	120	122	117	EMS215: .ASCIZ	@PROGRAM TIMEOUT @
145	077402	104	125	122	EMS216: .ASCIZ	@DURING LOOK AHEAD TEST @
146	077432	114	117	117	EMS217: .ASCIZ	@LOOK AHEAD REGISTER,RMLA, @
147	077465	123	105	101	EMS220: .ASCIZ	@SEARCH COMMAND @
148	077505	102	101	104	EMS221: .ASCIZ	@BAD SECTOR ERROR 'BSE' (RMER2, BIT 15) @
149	077555	101	040	104	EMS222: .ASCIZ	@A DATA TRANSFER COMMAND @
150	077606	110	105	101	EMS223: .ASCIZ	@HEADER COMPARE INHIBIT 'HCI' (RMOF, BIT 10) @
151	077663	116	117	116	EMS224: .ASCIZ	@NONEXISTENT MEMORY 'NEM' (RMCS2, BIT 11) @
152						

1	077735	105	130	120	EH1:	.ASCIZ	@EXPCTD	RECEVD@			
2	077754	102	125	123	EH110:	.ASCIZ	@BUSADR@				
3	077763	040	122	115	EH111:	.ASCIZ	@ RMCS2	RMCS1@			
4											
5	100002	122	105	103	EH114:	.ASCIZ	@RECEVD	SNGPRT	DULPRT@		
6	100031	105	130	120	EH223:	.ASCIZ	@EXPCTD	RECEVD	DATA@		
7	100057	105	130	120	EH256:	.ASCII	@EXPCTD	RECEVD	RGSTR@<CRLF>		
8	100105	123	124	101		.ASCIZ	@STATUS	STATUS	INDEX@		
9											
10	100133	107	104	101	EH336:	.ASCIZ	@GDADRS	GDDATA	BDADRS	BDDATA@	
11	100172	122	115	103	EH337:	.ASCII	@RMCS2	STATUS	FAILING	DATA@<CRLF>	
12	100231	137	137	137		.ASCII	@			@<CRLF>	
13	100270	105	130	120		.ASCIZ	@EXPCTD	RECEVD	--BIT--	ADDRESS@	
14											
15	100327	122	115	105	EH344:	.ASCII	@RMER1	STATUS	HEADER	FAILING@<CRLF>	
16	100367	137	137	137		.ASCII	@		WORD	BIT@<CRLF>	
17	100425	105	130	120		.ASCIZ	@EXPCTD	RECEVD	NUMBER	POSITON@	
18	100465	105	130	120	EH353:	.ASCII	@EXPCTD	RECEVD@<CRLF>			
19	100504	074	103	122		.ASCIZ	@<CRLF>	RMLA	RMLA	RMOF @	
20											
21	100541	040	122	115	STSH1:	.ASCII	@ RMCS1	RMCS2	RMDS	RMER1	RMER2@
22	100606	040	040	040		.ASCIZ	@	RMAS@			
23	100616	040	122	115	STSH2:	.ASCII	@ RMWC	RMBA	RMDA	RMOF	RMDC@
24	100663	040	040	040		.ASCIZ	@	RMEC1	RMEC2@		
25	100705	040	122	115	STSH3:	.ASCIZ	@ RMDA	RMDC	RMOF	RMLA@	
26	100743	040	122	115	STSH4:	.ASCIZ	@ RMMR1	RMMR2	RMDT	RMSN@	
27											
28						.EVEN					

1	101002	001140	001142	000000	ED1:	.WORD	\$GDDAT,\$BDDAT,0
2	101010	001276	000000		ED110:	.WORD	\$BASE,0
3	101014	001174	001176	000000	ED111:	.WORD	\$TMPO,\$TMP1,0
4							
5	101022	001362	001176	001200	ED114:	.WORD	RMDTI,\$TMP1,\$TMP2,0
6	101032	001140	001142	001174	ED223:	.WORD	\$GDDAT,\$BDDAT,\$TMPO,0
7							
8	101042	001134	001140	001136	ED336:	.WORD	\$GDADR,\$GDDAT,\$BDADR,\$BDDAT,0
9							
10	101054	001140	001142	001174	ED337:	.WORD	\$GDDAT,\$BDDAT,\$TMPO,\$TMP1,0
11	101066	001140	001142	001442	ED353:	.WORD	\$GDDAT,\$BDDAT,RMOFO,0
12							
13	101076	001334	001344	001346	STSD1:	.WORD	RMCS11,RMCS21,RMDSI,RMER11,RMER21,RMASI,0
14	101114	001336	001340	001342	STSD2:	.WORD	RMWC1,RMBA1,RMDA1,RMOFI,RMDCI,RMEC11
15	101130	001402	000000			.WORD	RMEC21,0
16	101134	001342	001370	001366	STSD3:	.WORD	RMDA1,RMDCI,RMOFI,RMLA1,0
17	101146	001360	001374	001362	STSD4:	.WORD	RMMR11,RMMR21,RMDTI,RMSNI,0
18							

1	101160	000			EF110:	.BYTE	0
2	101161	000	000		EF111:	.BYTE	0,0
3	101163	000	000	000	EF114:	.BYTE	0,0,0
4	101166	000	000	000	EF336:	.BYTE	0,0,0,0
5	101172	000	000	000	EF337:	.BYTE	0,0,0,0,0
6							
7	101177	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0
8					.EVEN		
9							

```
1          ;STORAGE FOR GENERAL DATA TRANSFERRS
2 101206   BUFFER:
3 101206   BUFONE: .BLKW 258.
4 102212   BUFTWO: .BLKW 258.
5
6          ;STORAGE FOR MANUFACTURES 16 BIT MODE BAD SECTOR FILE
7 103216   000000 000000 MFGFIL: .WORD 0,0           ;2 HEADER WORDS
8          .BLKW 256.           ;256. WORDS OF DATA
9 104222   177777           .WORD -1           ;TERMINATOR IF FILE IS FULL
10
11         ;STORAGE FOR USERS 16 BIT MODE BAD SECTOR FILE
12 104224   000000 000000 USRFIL: .WORD 0,0           ;2 HEADER WORDS
13         .BLKW 256.           ;256. WORDS OF DATA
14 105230   177777           .WORD -1           ;TERMINATOR IF FILE IS FULL
15
16         .-BUFFER
17
18 101206   HELP:
19 101206   200           .ASCII <CRLF>
20 101207   200           .ASCII <CRLF>
21 101210   114          111          123   .ASCII @LIST OF TESTS@<CRLF>
22 101226   055          055          055   .ASCII @-----@<CRLF>
23 101244   124          061          011   .ASCII @T1      CONTROLLER ACCESS TEST@<CRLF>
24 101276   124          062          011   .ASCII @T2      DEVICE AVAILABLE TEST@<CRLF>
25 101327   124          063          011   .ASCII @T3      DRIVE TYPE TEST@<CRLF>
26 101352   124          064          011   .ASCII @T4      FORMAT ZEROS@<CRLF>
27 101372   124          065          011   .ASCII @T5      ZERO FILL TEST@<CRLF>
28 101414   124          066          011   .ASCII @T6      FORMAT CHECK ZEROS@<CRLF>
29 101442   124          067          011   .ASCII @T7      FORMAT CHECK ZEROS W/ WCE ERROR@<CRLF>
30 101505   124          061          060   .ASCII @T10     FORMAT ONES@<CRLF>
31 101525   124          061          061   .ASCII @T11     FORMAT CHECK ONES@<CRLF>
32 101553   124          061          062   .ASCII @T12     FORMAT CHECK ONES W/ WCE ERRORS@<CRLF>
33 101617   124          061          063   .ASCII @T13     FORMAT MULTIPLE SECTORS@<CRLF>
34 101653   124          061          064   .ASCII @T14     FORMAT W/ HEAD SWITCHING@<CRLF>
35 101710   124          061          065   .ASCII @T15     FORMAT W/ MID TRANSFER SEEK@<CRLF>
36 101750   124          061          066   .ASCII @T16     FORMAT W/ IMPLIED SEEK@<CRLF>
37 102003   124          061          067   .ASCII @T17     FORMAT EACH SECTOR ADDRESS@<CRLF>
38 102042   124          062          060   .ASCII @T20     FORMAT EACH TRACK ADDRESS@<CRLF>
39 102100   124          062          061   .ASCII @T21     FORMAT PRIME CYLINDERS@<CRLF>
40 102133   124          062          062   .ASCII @T22     READ HEADER & DATA IN LAST SECTOR@<CRLF>
41 102201   124          062          063   .ASCII @T23     READ HEADER & DATA W/ AOE ERROR@<CRLF>
42 102245   124          062          064   .ASCII @T24     READ INVALID SECTOR ADDRESS@<CRLF>
43 102305   124          062          065   .ASCII @T25     READ INVALID TRACK ADDRESS@<CRLF>
44 102344   124          062          066   .ASCII @T26     READ INVALID CYLINDER ADDRESS@<CRLF>
45 102406   124          062          067   .ASCII @T27     FORMAT AT OFFSET@<CRLF>
46 102433   124          063          060   .ASCII @T30     IVC FORMAT TEST@<CRLF>
47 102457   124          063          061   .ASCII @T31     FORMAT ERROR TEST@<CRLF>
48 102505   124          063          062   .ASCII @T32     FORMAT HCE, FIRST HEADER WORD@<CRLF>
49 102547   124          063          063   .ASCII @T33     FORMAT HCE, SECOND HEADER WORD@<CRLF>
50 102612   200
51 102613   117          120          105   .ASCII @OPERATIONAL SWITCH SETTINGS@<CRLF>
52 102647   055          055          055   .ASCII @-----@<CRLF>
53 102703   123          127          111   .ASCII @SWITCH          USE@<CRLF>
54 102720   055          055          055   .ASCII @-----@<CRLF>
55 102755   040          040          061   .ASCII @ 15      HALT ON ERROR@<CRLF>
56 103001   040          040          061   .ASCII @ 14      LOOP ON TEST@<CRLF>
57 103024   040          040          061   .ASCII @ 13      INHIBIT ERROR TYPEOUTS@<CRLF>
```

58	103061	040	040	061	.ASCII	@	12	@<CRLF>
59	103070	040	040	061	.ASCII	@	11	INHIBIT ITERATIONS@<CRLF>
60	103121	040	040	061	.ASCII	@	10	BELL ON ERROR@<CRLF>
61	103145	040	040	040	.ASCII	@	9	LOOP ON ERROR@<CRLF>
62	103171	040	040	040	.ASCII	@	8	LOOP ON TEST IN SWR<7:0>@<CRLF>
63	103230	040	040	040	.ASCII	@	7	TN128@<CRLF>
64	103244	040	040	040	.ASCII	@	6	TN64@<CRLF>
65	103257	040	040	040	.ASCII	@	5	TN32@<CRLF>
66	103272	040	040	040	.ASCII	@	4	TN16@<CRLF>
67	103305	040	040	040	.ASCII	@	3	TN8@<CRLF>
68	103317	040	040	040	.ASCII	@	2	TN4@<CRLF>
69	103331	040	040	040	.ASCII	@	1	TN2@<CRLF>
70	103343	040	040	040	.ASCIZ	@	0	TN1@<CRLF>
71								
72	000200			.END			200	

ABASE = 176700
 ACDW1 = 000000
 ACDW2 = 000000
 ACKSTS = 047462
 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
 ADEVN = 000000
 ADR = 000001
 AENV = 000000
 AENVN = 000000
 AFATAL = 000000
 ALL = 063220
 AMADR1 = 000000
 AMADR2 = 000000
 AMADR3 = 000000
 AMADR4 = 000000
 AMAMS1 = 000000
 AMAMS2 = 000000
 AMAMS3 = 000000
 AMAMS4 = 000000
 AMSGAD = 000000
 AMSGLG = 000000
 MSGTY = 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 = 001514
 ASNDC = 001512
 ASWREG = 000000
 ATA = 100000
 ATESTN = 000000

ATNMSK = 000377
 ATNTBL = 064506
 AUNIT = 000000
 AUSWR = 000000
 AVECT1 = 120254
 AVECT2 = 000000
 A16 = 000400
 A17 = 001000
 BACK = 000000
 BADSCT = 034276
 BADTMO = 005340
 BAI = 000010
 BB00 = 000001
 BB01 = 000002
 BB02 = 000004
 BB03 = 000010
 BB04 = 000020
 BB05 = 000040
 BB06 = 000100
 BB07 = 000200
 BB08 = 000400
 BB09 = 001000
 BIT0 = 000001
 BIT00 = 000001
 BIT01 = 000002
 BIT02 = 000004
 BIT03 = 000010
 BIT04 = 000020
 BIT05 = 000040
 BIT06 = 000100
 BIT07 = 000200
 BIT08 = 000400
 BIT09 = 001000
 BIT1 = 000002
 BIT10 = 002000
 BIT11 = 004000
 BIT12 = 010000
 BIT13 = 020000
 BIT14 = 040000
 BIT15 = 100000
 BIT2 = 000004
 BIT3 = 000010
 BIT4 = 000020
 BIT5 = 000040
 BIT6 = 000100
 BIT7 = 000200
 BIT8 = 000400
 BIT9 = 001000
 BOTADR = 033310
 BOTFLG = 033312
 BPTVEC = 000014
 BSE = 100000
 BUFFER = 101206
 BUFONE = 101206
 BUFTWO = 102212
 CC = 004000
 CH = 002000

CHRCNT = 033313
 CKSWR = 104410
 CLKADR = 001516
 CLKVCT = 001520
 CLR = 000040
 CLRSTS = 046602
 CMNSTA = 007612
 CMPBUF = 036462
 CMPERR = 044600
 CNSL00 = 063644
 CNSL01 = 063677
 CNSL02 = 063715
 CNSL03 = 063757
 CNSL04 = 063777
 CNSL05 = 064033
 CNSL06 = 064045
 CNSL07 = 064076
 CNSL08 = 064241
 CNSL09 = 064242
 CNTCLR = 046464
 COMMA = 063231
 CONT = 000100
 CR = 000015
 CRLF = 000200
 CTFLG = 001326
 CYLMSK = 001777
 DBCK = 100000
 DBEN = 040000
 DBL = 002000
 DCK = 100000
 DDISP = 177570
 DEBL = 020000
 DEVSEL = 045012
 DISPLA = 001156
 DISPRE = 000174
 DLT = 100000
 DMD = 000001
 DPE = 000010
 DPEHI = 040000
 DPELO = 020000
 DPR = 000400
 DRQ = 004000
 DRVCLR = 000010
 DRVSTS = 052020
 DRY = 000200
 DSWR = 177570
 DTASTS = 052622
 DTE = 010000
 DTO = 010000
 DULPRT = 024024
 DVA = 004000
 DVC = 000200
 EARLY = 065032
 EBL = 020000
 ECH = 000100
 ECI = 004000
 ECRC = 001000

EDT1 = 071362
 EDT110 = 071400
 EDT111 = 071402
 EDT114 = 071404
 EDT2 = 071372
 EDT223 = 071406
 EDT336 = 071416
 EDT337 = 071426
 EDT344 = 071436
 EDT353 = 071450
 ED1 = 101002
 ED110 = 101010
 ED111 = 101014
 ED114 = 101022
 ED223 = 101032
 ED336 = 101042
 ED337 = 101054
 ED353 = 101066
 EECC = 000020
 EFT1 = 071452
 EFT110 = 071470
 EFT111 = 071472
 EFT114 = 071474
 EFT2 = 071462
 EFT223 = 071476
 EFT336 = 071506
 EFT337 = 071516
 EFT344 = 071526
 EFT353 = 071536
 EF110 = 101160
 EF111 = 101161
 EF114 = 101163
 EF336 = 101166
 EF337 = 101172
 EHT1 = 071236
 EHT110 = 071260
 EHT111 = 071264
 EHT114 = 071270
 EHT2 = 071250
 EHT223 = 071274
 EHT256 = 071306
 EHT336 = 071320
 EHT337 = 071332
 EHT344 = 071344
 EHT353 = 071356
 EH1 = 077735
 EH110 = 077754
 EH111 = 077763
 EH114 = 100002
 EH223 = 100031
 EH256 = 100057
 EH336 = 100133
 EH337 = 100172
 EH344 = 100327
 EH353 = 100465
 EMS1 = 071540
 EMS10 = 072074

EMS100 = 074312
 EMS101 = 074340
 EMS102 = 074366
 EMS103 = 074415
 EMS104 = 074424
 EMS105 = 074453
 EMS106 = 074502
 EMS11 = 072136
 EMS110 = 074521
 EMS111 = 074550
 EMS112 = 074561
 EMS113 = 074645
 EMS114 = 074675
 EMS115 = 074721
 EMS116 = 074724
 EMS117 = 074754
 EMS12 = 072147
 EMS120 = 074776
 EMS121 = 075017
 EMS122 = 075062
 EMS123 = 075117
 EMS124 = 075162
 EMS125 = 075214
 EMS126 = 075257
 EMS127 = 075322
 EMS13 = 072210
 EMS130 = 075361
 EMS131 = 075417
 EMS132 = 075457
 EMS133 = 075503
 EMS134 = 075526
 EMS135 = 075570
 EMS136 = 075633
 EMS137 = 075660
 EMS14 = 072221
 EMS140 = 075715
 EMS141 = 075736
 EMS142 = 075764
 EMS143 = 076001
 EMS144 = 076027
 EMS145 = 076037
 EMS146 = 076066
 EMS147 = 076105
 EMS15 = 072232
 EMS150 = 076122
 EMS151 = 076141
 EMS152 = 076165
 EMS153 = 076211
 EMS154 = 076233
 EMS155 = 076247
 EMS156 = 076267
 EMS157 = 076305
 EMS16 = 072254
 EMS160 = 076322
 EMS161 = 076337
 EMS162 = 076367
 EMS163 = 076367

EMS164	076413	EMS44	073176	EMT125	066516	EMT207	067420	EMT271	070166
EMS165	076474	EMS45	073251	EMT126	066526	EMT21	065310	EMT272	070204
EMS166	076510	EMS46	073314	EMT127	066540	EMT210	067432	EMT273	070222
EMS167	076535	EMS47	073343	EMT13	065232	EMT211	067440	EMT274	070240
EMS17	072304	EMS5	071732	EMT130	066552	EMT212	067454	EMT275	070256
EMS170	076562	EMS50	073372	EMT131	066564	EMT213	067470	EMT276	070270
EMS171	076611	EMS51	073421	EMT132	066576	EMT214	067500	EMT277	070304
EMS172	076644	EMS52	073447	EMT133	066610	EMT215	067520	EMT3	065156
EMS173	076664	EMS53	073457	EMT134	066622	EMT216	067532	EMT30	065400
EMS174	076701	EMS54	073471	EMT135	066634	EMT217	067542	EMT300	070322
EMS175	076707	EMS55	073513	EMT136	066646	EMT22	065320	EMT301	070342
EMS176	076735	EMS56	073542	EMT137	066660	EMT220	067552	EMT302	070364
EMS177	076764	EMS57	073617	EMT14	065244	EMT221	067562	EMT303	070376
EMS2	071607	EMS6	071762	EMT140	066670	EMT222	067572	EMT304	070410
EMS20	072344	EMS60	073645	EMT141	066700	EMT223	067602	EMT305	070422
EMS200	077006	EMS61	073660	EMT142	066710	EMT224	067612	EMT306	070434
EMS201	077023	EMS62	073707	EMT143	066720	EMT225	067614	EMT307	070444
EMS202	077071	EMS63	073725	EMT144	066730	EMT226	067616	EMT31	065410
EMS203	077121	EMS64	073753	EMT145	066740	EMT227	067620	EMT310	070464
EMS204	077134	EMS65	073771	EMT146	066750	EMT23	065330	EMT311	070476
EMS205	077162	EMS66	074037	EMT147	066760	EMT230	067622	EMT312	070510
EMS206	077174	EMS67	074107	EMT15	065252	EMT231	067624	EMT313	070522
EMS207	077221	EMS7	072027	EMT150	066770	EMT232	067626	EMT314	070542
EMS21	072367	EMS70	074134	EMT151	067000	EMT233	067630	EMT315	070554
EMS210	077246	EMS71	074146	EMT152	067012	EMT234	067632	EMT316	070566
EMS211	077257	EMS72	074161	EMT153	067024	EMT235	067634	EMT317	070600
EMS212	077275	EMS73	074171	EMT154	067042	EMT236	067636	EMT32	065420
EMS213	077320	EMS74	074206	EMT155	067060	EMT237	067640	EMT320	070610
EMS214	077343	EMS75	074225	EMT156	067072	EMT24	065340	EMT321	070620
EMS215	077361	EMS76	074233	EMT157	067104	EMT240	067642	EMT322	070630
EMS216	077402	EMS77	074264	EMT16	065260	EMT241	067644	EMT323	070636
EMS217	077432	EMTVEC=	000030	EMT160	067116	EMT242	067646	EMT324	070646
EMS22	072412	EMT1	065144	EMT161	067126	EMT243	067650	EMT325	070660
EMS220	077465	EMT10	065214	EMT162	067140	EMT244	067652	EMT326	070672
EMS221	077505	EMT100	066212	EMT163	067152	EMT245	067654	EMT327	070710
EMS222	077555	EMT101	066230	EMT164	067162	EMT246	067656	EMT33	065430
EMS223	077606	EMT102	066246	EMT165	067170	EMT247	067666	EMT330	070722
EMS224	077663	EMT103	066256	EMT166	067176	EMT25	065350	EMT331	070732
EMS23	072426	EMT104	066270	EMT167	067204	EMT250	067700	EMT332	070744
EMS24	072454	EMT105	066306	EMT17	065270	EMT251	067714	EMT333	070756
EMS25	072503	EMT106	066316	EMT170	067212	EMT252	067722	EMT334	070774
EMS26	072520	EMT107	066326	EMT171	067222	EMT253	067730	EMT335	071012
EMS27	072541	EMT11	065220	EMT172	067232	EMT254	067746	EMT336	071032
EMS3	071624	EMT110	066346	EMT173	067242	EMT255	067756	EMT337	071042
EMS30	072552	EMT111	066360	EMT174	067252	EMT256	067766	EMT34	065440
EMS31	072562	EMT112	066366	EMT175	067264	EMT257	067776	EMT340	071052
EMS32	072611	EMT113	066374	EMT176	067272	EMT26	065360	EMT341	071064
EMS33	072640	EMT114	066410	EMT177	067300	EMT260	070010	EMT342	071072
EMS34	072666	EMT115	066416	EMT2	065150	EMT261	070022	EMT343	071104
EMS35	072737	EMT116	066426	EMT20	065300	EMT262	070042	EMT344	071116
EMS36	072766	EMT117	066436	EMT200	067312	EMT263	070052	EMT345	071130
EMS37	073015	EMT12	065224	EMT201	067324	EMT264	070062	EMT346	071140
EMS4	071667	EMT120	066446	EMT202	067336	EMT265	070102	EMT347	071154
EMS40	073043	EMT121	066456	EMT203	067350	EMT266	070122	EMT35	065450
EMS41	073072	EMT122	066466	EMT204	067362	EMT267	070134	EMT350	071166
EMS42	073120	EMT123	066476	EMT205	067400	EMT27	065370	EMT351	071204
EMS43	073147	EMT124	066506	EMT206	067410	EMT270	070152	EMT352	071214

EMT353	071220	FNCMSK=	000077	IR	=	000100	PHA	=	000200	RMDAI	001342						
EMT354	071230	F0	=	000002	IVC	=	010000	PIP	=	020000	RMDAO	001416					
EMT36	065460	F1	=	000004	LBC	=	002000	PIRQ	=	177772	RMDB	=	000022				
EMT37	065470	F2	=	000010	LBT	=	002000	PIRQVE=	000240	RMDBI	001356						
EMT4	065164	F3	=	000020	LF	=	000012	PLFS	=	002000	RMDBO	001432					
EMT40	065500	F4	=	000040	LODEV	064305	PRIERR	040106	PRO	=	000000	RMDC	=	000034			
EMT41	065506	GENBUF	036224	LS	=	000004	PR0	=	000000	RMDCI	001370						
EMT42	065516	GET	037110	LSC	=	004000	PR1	=	000040	RMDCO	001444						
EMT43	065530	GETBUF	001334	LST	=	000002	PR2	=	000100	RMDS	=	000012					
EMT44	065540	GETINX	001522	LSTRK	001332	MCLK	=	004000	PR3	=	000140	RMDSI	001346				
EMT45	065550	GETSTS	037024	MCPE	=	020000	PR4	=	000200	RMDSO	001422						
EMT46	065560	GO	=	000001	MDF	=	000100	PR5	=	000240	RMDT	=	000026				
EMT47	065570	GTSWR	=	104407	MDPE	=	000400	PR6	=	000300	RMDTI	001362					
EMT5	065172	HCE	=	000200	MEDENB	001510	PR7	=	000340	RMDTO	001436						
EMT50	065576	HCI	=	002000	MFGFIL	103216	PS	=	177776	RMEC1	=	000044					
EMT51	065606	HCRC	=	000400	MI	=	000004	PSEL	=	002000	RMEC11	001400					
EMT52	065620	HELP	101206	MIXED	064516	MOH	=	020000	PSW	=	177776	RMEC10	001454				
EMT53	065636	HT	=	000011	MOC	=	000400	PWRVEC=	000024	PUT	037360	RMEC2	=	000046			
EMT54	065654	IAE	=	002000	MOL	=	010000	PUTBUF	001410	PUTINX	001551	RMEC21	001402				
EMT55	065664	IDXMSK=	000077	MRD	=	002000	RDCHR	=	104411	QES	063225	RMEC20	001456				
EMT56	065676	IE	=	000100	MS	=	000040	RDOCT	=	104413	RCLSTS	050256	RMER1	=	000014		
EMT57	065714	ILF	=	000001	MSC	=	000002	RDY	=	000200	RD	=	000070	RMER11	001350		
EMT6	065200	ILF02	=	000002	MSDRVS	064263	RDLIN	=	104412	RECAL	=	000006	RMER10	001424			
EMT60	065724	ILF24	=	000024	MSE	=	100000	RDICT	=	104413	RESREG=	104415	RMER2	=	000042		
EMT61	065742	ILF26	=	000026	MSER	=	000200	READY	007764	RESVEC=	000010	RMLA	=	000020	RMER21	001376	
EMT62	065762	ILF30	=	000030	MSGDRV	064277	RHR	=	000072	RG	=	040000	RMLAI	001354			
EMT63	065776	ILF32	=	000032	MSHELP	063234	RIP	=	000020	RGDTP	064516	RMLAO	001430				
EMT64	066000	ILF34	=	000034	MUR	=	001000	RELEASE=	000012	RH	=	000072	RMMR1	=	000024		
EMT65	066020	ILF36	=	000036	MWD	=	000010	RMAS	=	000016	RIP	=	000020	RMMR11	001360		
EMT66	066040	ILF40	=	000040	MWP	=	000010	RMASI	001352	RIP	=	000020	RMMR10	001434			
EMT67	066052	ILF42	=	000042	MXF	=	001000	RMASO	001426	RIP	=	000020	RMMR2	=	000040		
EMT67	066052	ILF44	=	000044	NDTMSK=	115760	RMB	=	000004	RIP	=	000020	RMMR21	001374			
EMT67	066052	ILF46	=	000046	NED	=	010000	RMBAE	=	000050	RIP	=	000020	RMMR20	001450		
EMT7	065206	ILF48	=	000048	NEM	=	004000	RMBAE1	001404	RIP	=	000020	RMOF	=	000032		
EMT70	066064	ILF48	=	000048	NOP	=	000000	RMBAE0	001460	RIP	=	000020	RMOF1	001366			
EMT71	066074	ILF54	=	000054	NOTAVL	064342	RMB	=	000004	RIP	=	000020	RMOFO	001442			
EMT71	066074	ILF56	=	000056	NOTPRS	064325	RMBAI	001340	RIP	=	000020	RIP	=	000020	RMR	=	000004
EMT72	066104	ILF58	=	000058	NSA	=	100000	RMBAO	001414	RIP	=	000020	RMSN	=	000030		
EMT73	066122	ILF64	=	000064	OCC	=	100000	RMC1	=	000000	RIP	=	000020	RMSNI	001364		
EMT74	066140	ILF66	=	000066	OFD	=	000200	RMC11	001334	RIP	=	000020	RMSNO	001440			
EMT74	066140	ILF68	=	000068	OFFLIN	064361	RMC2	=	000010	RIP	=	000020	RMWC	=	000002		
EMT75	066150	ILF74	=	000074	OFFSET=	000014	RMC3	=	000052	RIP	=	000020	RMWCI	001336			
EMT75	066150	ILF76	=	000076	OM	=	000001	RMC4	=	000000	RIP	=	000020	RMWCO	001412		
EMT76	066170	ILR	=	000002	ONES	064556	RMC10	001410	RIP	=	000020	RIP	=	000020	RQA	=	100000
EMT77	066200	ILRG50=	000050	ONLINE	064373	OPE	=	020000	RMC11	001334	RIP	=	000020	RQB	=	040000	
ENRGDT	065144	ILRG52=	000052	OPI	=	020000	RMC10	001410	RMC21	001344	RIP	=	000020	RTC	=	000016	
EQUALS	063216	ILRG54=	000054	PACACK=	000022	OR	=	000200	RMC20	001420	RIP	=	000020	R6	=	X000006	
ERR	=	040000	ILRG56=	000056	PAKACK=	000022	PACACK=	000022	RMC30	001462	RIP	=	000020	R7	=	X000007	
ERRNMB	033306	ILRG60=	000060	PAR	=	000010	PAR	=	000010	RMC31	001406	RIP	=	000020	SADMSK=	000377	
ERROR	=	104000	ILRG62=	000062	PAT	=	000020	PAT	=	000020	RMC33	=	000052	SAVREG=	104414		
ERRTYP	032550	ILRG64=	000064	PDA	=	000400	PDA	=	000400	RMC30	001462	RIP	=	000020	SA1	=	000001
ERRVEC=	000004	ILRG66=	000066	PGE	=	002000	PGE	=	002000	RMC31	001406	RIP	=	000020	SA16	=	000020
ERTY00	033314	ILRG68=	000068	PGM	=	001000	PGM	=	001000	RMC30	001462	RIP	=	000020			
ERTY01	033322	ILRG70=	000070									RIP	=	000020			
ERTY02	033332	ILRG72=	000072									RIP	=	000020			
ERTY03	033341	ILRG74=	000074									RIP	=	000020			
ERTY04	033347	ILRG76=	000076									RIP	=	000020			
ESRC	=	004000	IOTVEC=	000020								RIP	=	000020			
FER	=	000020	IPCK0	=	000001							RIP	=	000020			
FIND	=	000001	IPCK1	=	000002							RIP	=	000020			
FMT16	=	010000	IPCK2	=	000004							RIP	=	000020			
FNCDTB	064406	IPCK3	=	000010								RIP	=	000020			

SA2 = 000002
 SA4 = 000004
 SAB = 000010
 SC = 100000
 SCOPE = 000004
 SCTMSG = 063120
 SCTMSK = 003700
 SCO = 000100
 SC1 = 000200
 SC2 = 000400
 SC3 = 001000
 SC4 = 002000
 SEARCH = 000030
 SECERR = 040740
 SEEK = 000004
 SEKSTS = 045224
 SHUT = 056756
 SHUT2 = 057034
 SIZCLK = 037600
 SKI = 040000
 SNGPRT = 020024
 STACK = 001100
 STANDA = 006706
 START = 005420
 STCDRV = 056222
 STKLMT = 177774
 STOP = 056726
 STSD1 = 101076
 STSD2 = 101114
 STSD3 = 101134
 STSD4 = 101146
 STSF = 101177
 STSH1 = 100541
 STSH2 = 100616
 STSH3 = 100705
 STSH4 = 100743
 SWR = 001154
 SWREG = 000176
 SW0 = 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
 TADMSK = 177400
 TAG = 020000
 TAGADR = 001114
 TAP = 040000
 TA1 = 000400
 TA16 = 010000
 TA2 = 001000
 TA4 = 002000
 TAB = 004000
 TBITVE = 000014
 TIMEOUT = 037722
 TKVEC = 000060
 TPVEC = 000064
 TRAPVE = 000034
 TRE = 040000
 TRTVEC = 000014
 TST = 010000
 TSTNMB = 033304
 TSTPRP = 033352
 TSTQUE = 001464
 TST1 = 010130
 TST10 = 014130
 TST11 = 014750
 TST12 = 015540
 TST13 = 016476
 TST14 = 017272
 TST15 = 020104
 TST16 = 020716
 TST17 = 021540
 TST2 = 010314
 TST20 = 022302
 TST21 = 023044
 TST22 = 023606
 TST23 = 024112
 TST24 = 024416
 TST25 = 024760
 TST26 = 025302
 TST27 = 025620
 TST3 = 010500
 TST30 = 026532
 TST31 = 027354
 TST32 = 030136
 TST33 = 031306
 TST4 = 010702
 TST5 = 011544
 TST6 = 012400
 TST7 = 013170
 TYPBN = 104406
 TYPDS = 104405
 TYPE = 104401

TYPOC = 104402
 TYPON = 104404
 TYPOS = 104403
 UBUSQS = 063611
 UNS = 040000
 UNTMSK = 000007
 UPE = 020000
 USE = 040000
 USRFIL = 104224
 UO = 000001
 U1 = 000002
 U2 = 000004
 VV = 000100
 WC = 000040
 WCD = 000050
 WCE = 040000
 WCEHI = 010000
 WCELO = 040000
 WCF = 000040
 WCH = 000052
 WD = 000060
 WH = 000062
 WLE = 004000
 WRL = 004000
 XSIZ = 006462
 XXDP = 001330
 ZEROS = 064620
 \$APTHD = 001100
 \$ATYC = 062676
 \$ATY1 = 062652
 \$ATY3 = 062660
 \$ATY4 = 062670
 \$AUTOB = 001150
 \$BASE = 001276
 \$BDADR = 001136
 \$BDDAT = 001142
 \$BELL = 001212
 \$BIN = 057212
 \$CDW1 = 001302
 \$CDW2 = 001304
 \$CHARC = 060216
 \$CKSWR = 061324
 \$CMTAG = 001114
 \$CM3 = 000000
 \$CM4 = 000005
 \$CNTLC = 062222
 \$CNTLG = 062234
 \$CNTLU = 062227
 \$CPUOP = 001250
 \$CRLF = 001217
 \$DBLK = 057430
 \$DDW0 = 001306
 \$DDW1 = 001310
 \$DDW2 = 001312
 \$DDW3 = 001314
 \$DDW4 = 001316
 \$DDW5 = 001320

\$DDW6 = 001322
 \$DDW7 = 001324
 \$DEVCT = 001232
 \$DEVM = 001300
 \$DOAGN = 032540
 \$DTBL = 057420
 \$ENDAD = 032530
 \$ENDCT = 032374
 \$ENULL = 032544
 \$ENV = 001242
 \$ENVM = 001243
 \$EOP = 032340
 \$EOPCT = 032366
 \$EOSP = 032302
 \$ERFLG = 001117
 \$ERMAX = 001131
 \$ERRR = 060626
 \$ERRPC = 001132
 \$ERRTB = 001600
 \$ERTTL = 001126
 \$ESCAP = 001210
 \$ETABL = 001242
 \$ETEND = 001326
 \$FATAL = 001224
 \$FFLG = 063116
 \$FILLC = 001172
 \$FILLS = 001171
 \$GDADR = 001134
 \$GDDAT = 001140
 \$GET42 = 032520
 \$GTSWR = 061414
 \$HD = 000000
 \$HIBTS = 001100
 \$HIOCT = 062364
 \$ICNT = 001120
 \$ILLUP = 062634
 \$INTAG = 001151
 \$ITEMB = 001130
 \$LF = 001220
 \$LFLG = 063115
 \$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 = 063114
 \$MNEW = 062252
 \$MSGAD = 001236
 \$MSGLG = 001240
 \$MSGTY = 001222

\$MSWR = 062241
 \$MTYP1 = 001253
 \$MTYP2 = 001257
 \$MTYP3 = 001263
 \$MTYP4 = 001267
 \$MXCNT = 060536
 \$NULL = 001170
 \$NWTST = 000001
 \$OCNT = 057662
 \$OMODE = 057664
 \$OVER = 060522
 \$PASS = 001230
 \$PASTM = 001106
 \$POWER = 062642
 \$PWRDN = 062474
 \$PWRMG = 062630
 \$PWRUP = 062546
 \$QUES = 001216
 \$RDCHR = 061666
 \$RDLIN = 061756
 \$RDOCT = 062264
 \$RDSZ = 000010
 \$RESRE = 057102
 \$RTNAD = 032542
 \$SAVRE = 057044
 \$SAVR6 = 062640
 \$SCOPE = 060222
 \$SETUP = 000137
 \$STUP = 177777
 \$SVLAD = 060466
 \$SVPC = 000204
 \$SWR = 167400
 \$SWREG = 001244
 \$SWRMK = 000000
 \$SWOBT = 060540
 \$TESTN = 001226
 \$TIMES = 001206
 \$TKB = 001162
 \$TKCNT = 061026
 \$TKINT = 061036
 \$TKQEN = 061035
 \$TKQIN = 061030
 \$TKQOU = 061032
 \$TKQSR = 061034
 \$TKS = 001160
 \$TKSRV = 061106
 \$TMP0 = 001174
 \$TMP1 = 001176
 \$TMP2 = 001200
 \$TMP3 = 001202
 \$TMP4 = 001204
 \$TN = 000034
 \$TPB = 001166
 \$TPFLG = 001173
 \$TPS = 001164
 \$TRAP = 062366
 \$TRAP2 = 062426

STRP = 000016
STRPAD 062440
STSTM 001104
STSTNM 001116
STTYIN 062212

STYPBN 057140
STYPDS 057214
STYPE 057666
STYPEC 060100
STYPEX 060220

STYPOC 057464
STYPON 057500
STYPOS 057440
SUNIT 001234
SUNITM 001110

SUSWR 001246
\$VECT1 001272
\$VECT2 001274
\$XOFF = 000023
\$XON = 000021

\$XTSTR 060240
\$\$GET4= 000000
\$\$SW08= 000034
\$OFILL 057663
.\$X = 001100

. ABS. 105232 000
 000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 55176 WORDS (216 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
CZRMNA.BIC,CZRMNA/C=CZRMNA.DOC,CZRMNA,SYSMAC/M

SINTAG 5-0W 38-8 38-8 38-8 38-8 38-8* D 6

SEQ 0274

\$SETUP 4-782 4-782 4-782 4-782 4-782 4-782 4-782^F 4-782⁶# 4-782# 4-782# 4-782# 4-782# 4-782# 4-782# 9-19
SEQ 0276

38-10

38-10

38-10

38-10

38-10

38-10

38-10^H 6

38-10

38-10

38-10

38-10

38-10

38-10
SEQ 0278

A TESTN 5-0 5-0

L 6

SEQ 0282

ATNMSK	4-596#	34-55												
ATNTBL	9-74	9-96	10-149	11-11	41-3#									
AUNIT	5-0	5-0												
AUSWR	5-0	5-0												
AVECT1	4-780#	5-0	5-0											
AVECT2	5-0	5-0												
BACK	12-102	12-102#	12-110	12-110#	12-122	12-122#	12-131	12-131#	12-135	12-135#	12-135#	12-146	12-146#	12-152
	12-152#	12-156	12-156#	12-156#	12-158	12-158#	12-158#	12-160	12-160#	12-160#	12-167	12-167#	12-173	12-173#
	12-177	12-177#	12-177#	12-179	12-179#	12-179#	12-181	12-181#	12-181#	12-185	12-185#	12-203	12-203#	12-211
	12-211#	12-223	12-223#	12-232	12-232#	12-236	12-236#	12-236#	12-248	12-248#	12-254	12-254#	12-258	12-258#
	12-258#	12-260	12-260#	12-260#	12-262	12-262#	12-262#	12-270	12-270#	12-276	12-276#	12-280	12-280#	12-280#
	12-282	12-282#	12-282#	12-284	12-284#	12-284#	12-288	12-288#	12-301	12-301#	12-309	12-309#	12-321	12-321#
	12-330	12-330#	12-334	12-334#	12-334#	12-345	12-345#	12-351	12-351#	12-355	12-355#	12-355#	12-357	12-357#
	12-357#	12-359	12-359#	12-359#	12-365	12-365#	12-371	12-371#	12-375	12-375#	12-375#	12-375#	12-377	12-377#
	12-379	12-379#	12-379#	12-394	12-394#	12-402	12-402#	12-414	12-414#	12-423	12-423#	12-427	12-427#	12-427#
	12-438	12-438#	12-444	12-444#	12-448	12-448#	12-448#	12-450	12-450#	12-450#	12-452	12-452#	12-452#	12-461
	12-461#	12-467	12-467#	12-471	12-471#	12-471#	12-478	12-478#	12-478#	12-496	12-496#	12-519	12-519#	12-527
	12-527#	12-539	12-539#	12-548	12-548#	12-552	12-552#	12-552#	12-563	12-563#	12-569	12-569#	12-573	12-573#
	12-573#	12-575	12-575#	12-575#	12-577	12-577#	12-577#	12-584	12-584#	12-590	12-590#	12-594	12-594#	12-594#
	12-596	12-596#	12-596#	12-598	12-598#	12-598#	12-602	12-602#	12-615	12-615#	12-623	12-623#	12-635	12-635#
	12-644	12-644#	12-648	12-648#	12-648#	12-659	12-659#	12-665	12-665#	12-669	12-669#	12-669#	12-671	12-671#
	12-671#	12-673	12-673#	12-673#	12-679	12-679#	12-685	12-685#	12-689	12-689#	12-689#	12-691	12-691#	12-691#
	12-693	12-693#	12-693#	12-708	12-708#	12-716	12-716#	12-728	12-728#	12-737	12-737#	12-741	12-741#	12-741#
	12-752	12-752#	12-758	12-758#	12-762	12-762#	12-762#	12-764	12-764#	12-764#	12-766	12-766#	12-766#	12-775
	12-775#	12-781	12-781#	12-785	12-785#	12-785#	12-792	12-792#	12-792#	12-808	12-808#	12-832	12-832#	12-840
	12-840#	12-852	12-852#	12-861	12-861#	12-865	12-865#	12-865#	12-876	12-876#	12-882	12-882#	12-886	12-886#
	12-886#	12-888	12-888#	12-888#	12-890	12-890#	12-890#	12-896	12-896#	12-902	12-902#	12-906	12-906#	12-906#
	12-908	12-908#	12-908#	12-910	12-910#	12-910#	12-926	12-926#	12-935	12-935#	12-947	12-947#	12-956	12-956#
	12-960	12-960#	12-960#	12-971	12-971#	12-977	12-977#	12-981	12-981#	12-981#	12-983	12-983#	12-983#	12-985
	12-985#	12-985#	12-991	12-991#	12-997	12-997#	12-:01	12-:01#	12-:01#	12-:03	12-:03#	12-:03#	12-:05	12-:05#
	12-:05#	12-:21	12-:21#	12-:31	12-:31#	12-:43	12-:43#	12-:52	12-:52#	12-:56	12-:56#	12-:56#	12-:67	12-:67#
	12-:73	12-:73#	12-:77	12-:77#	12-:77#	12-:79	12-:79#	12-:79#	12-:81	12-:81#	12-:81#	12-:87	12-:87#	12-:93
	12-:93#	12-:97	12-:97#	12-:97#	12-:99	12-:99#	12-:99#	12-:01	12-:01#	12-:01#	12-:16	12-:16#	12-:24	12-:24#
	12-:38	12-:38#	12-:47	12-:47#	12-:51	12-:51#	12-:51#	12-:63	12-:63#	12-:69	12-:69#	12-:73	12-:73#	12-:73#
	12-:75	12-:75#	12-:75#	12-:77	12-:77#	12-:77#	12-:83	12-:83#	12-:89	12-:89#	12-:93	12-:93#	12-:93#	12-:95
	12-:95#	12-:95#	12-:97	12-:97#	12-:97#	12-<17	12-<17#	12-<25	12-<25#	12-<39	12-<39#	12-<46	12-<46#	12-<50
	12-<50#	12-<50#	12-<52	12-<52#	12-<52#	12-<54	12-<54#	12-<54#	12-<61	12-<61#	12-<67	12-<67#	12-<71	12-<71#
	12-<71#	12-<73	12-<73#	12-<73#	12-<75	12-<75#	12-<75#	12-<79	12-<79#	12-=00	12-=00#	12-=08	12-=08#	12-=22
	12-=22#	12-=29	12-=29#	12-=33	12-=33#	12-=33#	12-=35	12-=35#	12-=35#	12-=37	12-=37#	12-=37#	12-=44	12-=44#
	12-=50	12-=50#	12-=54	12-=54#	12-=54#	12-=56	12-=56#	12-=56#	12-=58	12-=58#	12-=58#	12-=62	12-=62#	12-=83
	12-=83#	12-=92	12-=92#	12->05	12->05#	12->13	12->13#	12->17	12->17#	12->17#	12->19	12->19#	12->19#	12->21
	12->21#	12->21#	12->27	12->27#	12->32	12->32#	12->36	12->36#	12->36#	12->38	12->38#	12->38#	12->40	12->40#
	12->40#	12->44	12->44#	12->67	12->67#	12->80	12->80#	12->87	12->87#	12->91	12->91#	12->91#	12->93	12->93#
	12->93#	12->95	12->95#	12->95#	12-?13	12-?13#	12-?24	12-?24#	12-?31	12-?31#	12-?35	12-?35#	12-?35#	12-?37
	12-?37#	12-?37#	12-?39	12-?39#	12-?39#	12-?55	12-?55#	12-?68	12-?68#	12-?75	12-?75#	12-?79	12-?79#	12-?79#
	12-?81	12-?81#	12-?81#	12-?83	12-?83#	12-?83#	12-@11	12-@11#	12-@24	12-@24#	12-@31	12-@31#	12-@35	12-@35#
	12-@35#	12-@37	12-@37#	12-@37#	12-@39	12-@39#	12-@39#	12-@60	12-@60#	12-@73	12-@73#	12-@80	12-@80#	12-@84
	12-@84#	12-@84#	12-@86	12-@86#	12-@86#	12-@88	12-@88#	12-@88#	12-A08	12-A08#	12-A16	12-A16#	12-A28	12-A28#
	12-A39	12-A39#	12-A46	12-A46#	12-A50	12-A50#	12-A50#	12-A52	12-A52#	12-A54	12-A54#	12-A54#	12-A65	12-A65#
	12-A65#	12-A78	12-A78#	12-A84	12-A84#	12-A88	12-A88#	12-A88#	12-A90	12-A90#	12-A90#	12-A92	12-A92#	12-A92#
	12-A96	12-A96#	12-B15	12-B15#	12-B23	12-B23#	12-B31	12-B31#	12-B47	12-B47#	12-B54	12-B54#	12-B58	12-B58#
	12-B58#	12-B67	12-B67#	12-B67#	12-B71	12-B71#	12-B78	12-B78#	12-B85	12-B85#	12-B89	12-B89#	12-B89#	12-B98
	12-B98#	12-B98#	12-C14	12-C14#	12-C22	12-C22#	12-C36	12-C36#	12-C43	12-C43#	12-C47	12-C47#	12-C47#	12-C49
	12-C49#	12-C49#	12-C51	12-C51#	12-C51#	12-C61	12-C61#	12-C67	12-C67#	12-C71	12-C71#	12-C71#	12-C78	12-C78#
	12-C78#	12-C85	12-C85#	12-C85#	12-D06	12-D06#	12-D14	12-D14#	12-D36	12-D36#	12-D43	12-D43#	12-D47	12-D47#
	12-D47#	12-D49	12-D49#	12-D49#	12-D51	12-D51#	12-D51#	12-D58	12-D58#	12-D64	12-D64#	12-D68	12-D68#	12-D68#

12-D79 12-D79# 12-D79# 12-D92 12-D92# 12-D92# 12-E05^N 6 12-E05# 12-E05# 12-E17 12-E17# 12-E17# 12-E38 12-E38#
SEQ 0284

DULPRT	4-653#	12-71	12-76	12-81	12-85									
DVA	4-497#	9-89	12-51	20-43	21-37									
DVC	4-687#	29-127	31-117	33-92	33-130	24-67	24-70	28-37	31-33	34-17	34-18			
EARLY	42-107#					33-133	33-266	35-232	35-236	35-239	35-284	36-89	36-94	36-97
EBL	4-630#													
ECH	4-582#	4-590	16-225	18-25	25-452	25-467	25-485	25-491	35-423	40-77	40-78	40-85	40-86	
EC1	4-658#	18-21	25-487	35-420										
ECRC	4-634#													
ED1	45-1	49-1#												
ED110	45-4	49-2#												
ED111	45-5	49-3#												
ED114	45-7	49-5#												
ED223	45-8	49-6#												
ED336	45-10	49-8#												
ED337	45-11	45-12	49-10#											
ED353	45-14	49-11#												
EDT1	7-4	7-7	7-10	7-13	7-16	7-19	7-25	7-28	7-31	7-34	7-37	7-40	7-43	7-46
	7-49	7-52	7-55	7-58	7-61	7-64	7-67	7-70	7-73	7-76	7-79	7-82	7-85	7-88
	7-91	7-94	7-97	7-100	7-103	7-106	7-109	7-112	7-115	7-118	7-121	7-124	7-127	7-131
	7-134	7-137	7-140	7-143	7-147	7-151	7-155	7-161	7-164	7-167	7-170	7-173	7-176	7-179
	7-183	7-186	7-189	7-192	7-195	7-198	7-201	7-204	7-207	7-210	7-213	7-216	7-219	7-237
	7-240	7-243	7-246	7-249	7-252	7-255	7-258	7-261	7-264	7-267	7-270	7-273	7-276	7-279
	7-282	7-285	7-288	7-291	7-294	7-297	7-300	7-303	7-306	7-309	7-312	7-315	7-318	7-321
	7-324	7-327	7-330	7-333	7-336	7-339	7-342	7-345	7-348	7-355	7-358	7-361	7-364	7-370
	7-373	7-376	7-379	7-382	7-385	7-388	7-391	7-394	7-397	7-400	7-403	7-406	7-409	7-412
	7-415	7-418	7-421	7-424	7-433	7-436	7-439	7-442	7-445	7-505	7-508	7-511	7-520	7-523
	7-526	7-532	7-535	7-538	7-541	7-544	7-547	7-550	7-553	7-557	7-560	7-564	7-567	7-570
	7-574	7-578	7-582	7-587	7-591	7-595	7-598	7-601	7-604	7-607	7-610	7-613	7-616	7-619
	7-622	7-625	7-628	7-631	7-634	7-637	7-640	7-643	7-646	7-649	7-652	7-655	7-658	7-661
	7-664	7-667	7-670	7-673	7-676	7-679	7-694	7-697	7-703	7-706	7-709	7-712	7-715	7-724
	45-1#													
EDT110	7-222	45-4#												
EDT111	7-225	7-228	45-5#											
EDT114	7-234	45-7#												
EDT2	7-427	7-430	7-514	7-517	45-2#									
EDT223	7-448	7-529	45-8#											
EDT336	7-352	7-682	7-688	7-691	45-10#									
EDT337	7-685	45-11#												
EDT344	7-700	45-12#												
EDT353	7-721	45-14#												
EECC	4-639#													
EF110	46-4	50-1#												
EF111	46-1	46-5	50-2#											
EF114	46-7	46-8	46-14	50-3#										
EF336	46-10	46-11	46-12	50-4#										
EF337	50-5#													
EFT1	7-4	7-7	7-10	7-13	7-16	7-19	7-25	7-28	7-31	7-34	7-37	7-40	7-43	7-46
	7-49	7-52	7-55	7-58	7-61	7-64	7-67	7-70	7-73	7-76	7-79	7-82	7-85	7-88
	7-91	7-94	7-97	7-100	7-103	7-106	7-109	7-112	7-115	7-118	7-121	7-124	7-127	7-131
	7-134	7-137	7-140	7-143	7-147	7-151	7-155	7-161	7-164	7-167	7-170	7-173	7-176	7-179
	7-183	7-186	7-189	7-192	7-195	7-198	7-201	7-204	7-207	7-210	7-213	7-216	7-219	7-237
	7-240	7-243	7-246	7-249	7-252	7-255	7-258	7-261	7-264	7-267	7-270	7-273	7-276	7-279
	7-282	7-285	7-288	7-291	7-294	7-297	7-300	7-303	7-306	7-309	7-312	7-315	7-318	7-321
	7-324	7-327	7-330	7-333	7-336	7-339	7-342	7-345	7-348	7-355	7-358	7-361	7-364	7-370
	7-373	7-376	7-379	7-382	7-385	7-388	7-391	7-394	7-397	7-400	7-403	7-406	7-409	7-412
	7-415	7-418	7-421	7-424	7-433	7-436	7-439	7-442	7-445	7-505	7-508	7-511	7-520	7-523

7-526 7-532 7-535 7-538 7-541 7-544 7-547^G 7-550 7-553 7-557 7-560 7-564 7-567 7-570
SEQ 0290



EMS162 47-116#

K 7

SEQ 0294

EMS4 43-5 43-75 47-6#

M 7

SEQ 0296

F
T
G
G
G
G
G
G
G
G
G
G

EMT111	7-225	43-75#
EMT112	7-228	43-76#
EMT113	7-231	43-77#
EMT114	7-234	43-78#
EMT115	7-237	43-79#
EMT116	7-240	43-80#
EMT117	7-243	43-81#
EMT12	7-31	43-12#
EMT120	7-246	43-82#
EMT121	7-249	43-83#
EMT122	7-252	43-84#
EMT123	7-255	43-85#
EMT124	7-258	43-86#
EMT125	7-261	43-87#
EMT126	7-264	43-88#
EMT127	7-267	43-89#
EMT13	7-34	43-13#
EMT130	7-270	43-90#
EMT131	7-273	43-91#
EMT132	7-276	43-92#
EMT133	7-279	43-93#
EMT134	7-282	43-94#
EMT135	7-285	43-95#
EMT136	7-288	43-96#
EMT137	7-291	43-97#
EMT14	7-37	43-14#
EMT140	7-294	43-98#
EMT141	7-297	43-99#
EMT142	7-300	43-100#
EMT143	7-303	43-101#
EMT144	7-306	43-102#
EMT145	7-309	43-103#
EMT146	7-312	43-104#
EMT147	7-315	43-105#
EMT15	7-40	43-15#
EMT150	7-318	43-106#
EMT151	7-321	43-107#
EMT152	7-324	43-108#
EMT153	7-327	43-109#
EMT154	7-330	43-110#
EMT155	7-333	43-111#
EMT156	7-336	43-112#
EMT157	7-339	43-113#
EMT16	7-43	43-16#
EMT160	7-342	43-114#
EMT161	7-345	43-115#
EMT162	7-348	43-116#
EMT163	43-117#	
EMT164	7-355	43-118#
EMT165	7-358	43-119#
EMT166	7-361	43-120#
EMT167	7-364	43-121#
EMT17	7-46	43-17#
EMT170	43-122#	
EMT171	7-370	43-123#
EMT172	7-373	43-124#

EMT174	7-379	43-126#
EMT175	7-382	43-127#
EMT176	7-385	43-128#
EMT177	7-388	43-129#
EMT2	7-7	43-4#
EMT20	7-49	43-18#
EMT200	7-391	43-130#
EMT201	7-394	43-131#
EMT202	7-397	43-132#
EMT203	7-400	43-133#
EMT204	7-403	43-134#
EMT205	7-406	43-135#
EMT206	7-409	43-136#
EMT207	7-412	43-137#
EMT21	7-52	43-19#
EMT210	7-415	43-138#
EMT211	7-418	43-139#
EMT212	7-421	43-140#
EMT213	7-424	43-141#
EMT214	7-427	43-142#
EMT215	7-430	43-143#
EMT216	7-433	43-144#
EMT217	7-436	43-145#
EMT22	7-55	43-20#
EMT220	7-439	43-146#
EMT221	7-442	43-147#
EMT222	7-445	43-148#
EMT223	7-448	43-149#
EMT224	43-150#	
EMT225	43-151#	
EMT226	43-152#	
EMT227	43-153#	
EMT23	7-58	43-21#
EMT230	43-154#	
EMT231	43-155#	
EMT232	43-156#	
EMT233	43-157#	
EMT234	43-158#	
EMT235	43-159#	
EMT236	43-160#	
EMT237	43-161#	
EMT24	7-61	43-22#
EMT240	43-162#	
EMT241	43-163#	
EMT242	43-164#	
EMT243	43-165#	
EMT244	43-166#	
EMT245	43-167#	
EMT246	7-505	43-168#
EMT247	7-508	43-169#
EMT25	7-64	43-23#
EMT250	7-511	43-170#
EMT251	7-514	43-171#
EMT252	7-517	43-172#
EMT253	7-520	43-173#
EMT254	7-523	43-174#

EMT256	7-529	43-176#
EMT257	7-532	43-177#
EMT26	7-67	43-24#
EMT260	7-535	43-178#
EMT261	7-538	43-179#
EMT262	7-541	43-180#
EMT263	7-544	43-181#
EMT264	7-547	43-182#
EMT265	7-550	43-183#
EMT266	7-553	43-184#
EMT267	7-557	43-185#
EMT27	7-70	43-25#
EMT270	7-560	43-186#
EMT271	7-564	43-187#
EMT272	7-567	43-188#
EMT273	7-570	43-189#
EMT274	7-574	43-190#
EMT275	7-578	43-191#
EMT276	7-582	43-192#
EMT277	7-587	43-193#
EMT3	7-10	43-5#
EMT30	7-73	43-26#
EMT300	7-591	43-194#
EMT301	7-595	43-195#
EMT302	7-598	43-196#
EMT303	7-601	43-197#
EMT304	7-604	43-198#
EMT305	7-607	43-199#
EMT306	7-610	43-200#
EMT307	7-613	43-201#
EMT31	7-76	43-27#
EMT310	7-616	43-202#
EMT311	7-619	43-203#
EMT312	7-622	43-204#
EMT313	7-625	43-205#
EMT314	7-628	43-206#
EMT315	7-631	43-207#
EMT316	7-634	43-208#
EMT317	7-637	43-209#
EMT32	7-79	43-28#
EMT320	7-640	43-210#
EMT321	7-643	43-211#
EMT322	7-646	43-212#
EMT323	7-649	43-213#
EMT324	7-652	43-214#
EMT325	7-655	43-215#
EMT326	7-658	43-216#
EMT327	7-661	43-217#
EMT33	7-82	43-29#
EMT330	7-664	43-218#
EMT331	7-667	43-219#
EMT332	7-670	43-220#
EMT333	7-673	43-221#
EMT334	7-676	43-222#
EMT335	7-679	43-223#
EMT336	7-352	7-682 43-224#

EMT337 7-685 43-225#

H 8

SEQ 0304

EMT34	7-85	43-30#
EMT340	7-688	43-226#
EMT341	7-691	43-227#
EMT342	7-694	43-228#
EMT343	7-697	43-229#
EMT344	7-700	43-230#
EMT345	7-703	43-231#
EMT346	7-706	43-232#
EMT347	7-709	43-233#
EMT35	7-88	43-31#
EMT350	7-712	43-234#
EMT351	7-715	43-235#
EMT352	7-718	43-236#
EMT353	7-721	43-237#
EMT354	7-724	43-238#
EMT36	7-91	43-32#
EMT37	7-94	43-33#
EMT4	7-13	43-6#
EMT40	7-97	43-34#
EMT41	7-100	43-35#
EMT42	7-103	43-36#
EMT43	7-106	43-37#
EMT44	7-109	43-38#
EMT45	7-112	43-39#
EMT46	7-115	43-40#
EMT47	7-118	43-41#
EMT5	7-16	43-7#
EMT50	7-121	43-42#
EMT51	7-124	43-43#
EMT52	7-127	43-44#
EMT53	7-131	43-45#
EMT54	7-134	43-46#
EMT55	7-137	43-47#
EMT56	7-140	43-48#
EMT57	7-143	43-49#
EMT6	7-19	43-8#
EMT60	7-147	43-50#
EMT61	7-151	43-51#
EMT62	7-155	43-52#
EMT63	43-53#	
EMT64	7-161	43-54#
EMT65	7-164	43-55#
EMT66	7-167	43-56#
EMT67	7-170	43-57#
EMT7	7-22	43-9#
EMT70	7-173	43-58#
EMT71	7-176	43-59#
EMT72	7-179	43-60#
EMT73	7-183	43-61#
EMT74	7-186	43-62#
EMT75	7-189	43-63#
EMT76	7-192	43-64#
EMT77	7-195	43-65#
EMTVEC	4-491#	9-19*
ENRGDT	42-144#	
EQUALS	39-5#	

ERR

4-560#

25-84

25-86

25-102

25-136

J 8

SEQ 0306

ERRNMB	14-28*	14-29*	14-32	14-36	14-41	14-133#									
ERROR	4-491#														
ERRTYP	14-15#	38-7													
ERRVEC	4-491#	9-19	9-19*	9-19*	9-21*	9-22*	12-5	12-6	12-7*	12-8*	12-19*	12-20*	12-24*	12-25*	
	12-38	12-39	12-40*	12-41*	12-45*	12-46*	12-57*	12-58*	20-35	20-35	20-39*	20-40*	20-70*	20-70*	
	21-29	21-29	21-33*	21-34*	21-59*	21-59*	22-4	22-5	22-6*	22-7*	22-13*	22-20*	22-21*	23-9	
	23-9	23-10*	23-11*	23-35*	23-35*	28-20	28-20	28-21*	28-22*	28-51*	28-51*	30-12	30-12	30-13*	
	30-14*	30-25*	30-25*	38-6	38-6*	38-6*	38-6*								
ERTY00	14-22	14-138#													
ERTY01	14-26	14-139#													
ERTY02	14-31	14-140#													
ERTY03	14-33	14-141#													
ERTY04	14-125	14-142#													
ESRC	4-632#														
F0	4-502#	25-29													
F1	4-501#	25-29													
F2	4-500#	25-29	26-113												
F3	4-499#	25-29													
F4	4-498#	25-29													
FER	4-584#	4-590	12-C75	12-C82	12-D76	12-D83	16-225	25-416	35-348	35-365	35-368	35-391	35-394		
FIND	12-102	12-102#	12-102#	12-110	12-110#	12-110#	12-122	12-122#	12-122#	12-131	12-131#	12-131#	12-135	12-135#	
	12-135#	12-146	12-146#	12-146#	12-152	12-152#	12-152#	12-156	12-156#	12-156#	12-158	12-158#	12-158#	12-160	
	12-160#	12-160#	12-167	12-167#	12-167#	12-173	12-173#	12-173#	12-177	12-177#	12-177#	12-179	12-179#	12-179#	
	12-181	12-181#	12-181#	12-185	12-185#	12-185#	12-203	12-203#	12-203#	12-211	12-211#	12-211#	12-223	12-223#	
	12-223#	12-232	12-232#	12-232#	12-236	12-236#	12-236#	12-248	12-248#	12-248#	12-254	12-254#	12-254#	12-258	
	12-258#	12-258#	12-260	12-260#	12-260#	12-262	12-262#	12-262#	12-270	12-270#	12-270#	12-276	12-276#	12-276#	
	12-280	12-280#	12-280#	12-282	12-282#	12-282#	12-284	12-284#	12-284#	12-288	12-288#	12-288#	12-301	12-301#	
	12-301#	12-309	12-309#	12-309#	12-321	12-321#	12-321#	12-330	12-330#	12-330#	12-334	12-334#	12-334#	12-345	
	12-345#	12-345#	12-351	12-351#	12-351#	12-355	12-355#	12-355#	12-357	12-357#	12-357#	12-359	12-359#	12-359#	
	12-365	12-365#	12-365#	12-371	12-371#	12-371#	12-375	12-375#	12-375#	12-377	12-377#	12-377#	12-379	12-379#	
	12-379#	12-394	12-394#	12-394#	12-402	12-402#	12-402#	12-414	12-414#	12-414#	12-423	12-423#	12-423#	12-427	
	12-427#	12-427#	12-438	12-438#	12-438#	12-444	12-444#	12-444#	12-448	12-448#	12-448#	12-450	12-450#	12-450#	
	12-452	12-452#	12-452#	12-461	12-461#	12-461#	12-467	12-467#	12-467#	12-471	12-471#	12-471#	12-478	12-478#	
	12-478#	12-496	12-496#	12-496#	12-519	12-519#	12-519#	12-527	12-527#	12-527#	12-539	12-539#	12-539#	12-548	
	12-548#	12-548#	12-552	12-552#	12-552#	12-563	12-563#	12-563#	12-569	12-569#	12-569#	12-573	12-573#	12-573#	
	12-575	12-575#	12-575#	12-577	12-577#	12-577#	12-584	12-584#	12-584#	12-590	12-590#	12-590#	12-594	12-594#	
	12-594#	12-596	12-596#	12-596#	12-598	12-598#	12-598#	12-602	12-602#	12-602#	12-615	12-615#	12-615#	12-623	
	12-623#	12-623#	12-635	12-635#	12-635#	12-644	12-644#	12-644#	12-648	12-648#	12-648#	12-659	12-659#	12-659#	
	12-665	12-665#	12-665#	12-669	12-669#	12-669#	12-671	12-671#	12-671#	12-673	12-673#	12-673#	12-679	12-679#	
	12-679#	12-685	12-685#	12-685#	12-689	12-689#	12-689#	12-691	12-691#	12-691#	12-693	12-693#	12-693#	12-708	
	12-708#	12-708#	12-716	12-716#	12-716#	12-728	12-728#	12-728#	12-737	12-737#	12-737#	12-741	12-741#	12-741#	
	12-752	12-752#	12-752#	12-758	12-758#	12-758#	12-762	12-762#	12-762#	12-764	12-764#	12-764#	12-766	12-766#	
	12-766#	12-775	12-775#	12-775#	12-781	12-781#	12-781#	12-785	12-785#	12-785#	12-792	12-792#	12-792#	12-808	
	12-808#	12-808#	12-832	12-832#	12-832#	12-840	12-840#	12-840#	12-852	12-852#	12-852#	12-861	12-861#	12-861#	
	12-865	12-865#	12-865#	12-876	12-876#	12-876#	12-882	12-882#	12-886	12-886#	12-886#	12-886#	12-888	12-888#	
	12-888#	12-890	12-890#	12-890#	12-896	12-896#	12-896#	12-902	12-902#	12-902#	12-906	12-906#	12-906#	12-908	
	12-908#	12-908#	12-910	12-910#	12-910#	12-926	12-926#	12-926#	12-935	12-935#	12-935#	12-947	12-947#	12-947#	
	12-956	12-956#	12-956#	12-960	12-960#	12-960#	12-971	12-971#	12-971#	12-977	12-977#	12-977#	12-981	12-981#	
	12-981#	12-983	12-983#	12-983#	12-985	12-985#	12-985#	12-991	12-991#	12-991#	12-997	12-997#	12-997#	12-:01	
	12-:01#	12-:01#	12-:03	12-:03#	12-:03#	12-:05	12-:05#	12-:05#	12-:21	12-:21#	12-:21#	12-:31	12-:31#	12-:31#	
	12-:43	12-:43#	12-:43#	12-:52	12-:52#	12-:52#	12-:56	12-:56#	12-:56#	12-:67	12-:67#	12-:73	12-:73#	12-:73#	
	12-:73#	12-:77	12-:77#	12-:77#	12-:79	12-:79#	12-:79#	12-:81	12-:81#	12-:81#	12-:87	12-:87#	12-:87#	12-:93	
	12-:93#	12-:93#	12-:97	12-:97#	12-:97#	12-:99	12-:99#	12-:99#	12-:01	12-:01#	12-:01#	12-:16	12-:16#	12-:16#	
	12-:24	12-:24#	12-:24#	12-:38	12-:38#	12-:38#	12-:47	12-:47#	12-:47#	12-:51	12-:51#	12-:51#	12-:63	12-:63#	
	12-:63#	12-:69	12-:69#	12-:69#	12-:73	12-:73#	12-:73#	12-:75	12-:75#	12-:75#	12-:77	12-:77#	12-:77#	12-:83	
	12-:83#	12-:83#	12-:89	12-:89#	12-:89#	12-:93	12-:93#	12-:93#	12-:95	12-:95#	12-:95#	12-:97	12-:97#	12-:97#	

12-<17 12-<17# 12-<17# 12-<25 12-<25# 12-<25# 12-<39^L 8 12-<39# 12-<39# 12-<46 12-<46# 12-<46# 12-<50 12-<50#
SEQ 0308

	12-<50#	12-<52	12-<52#	12-<52#	12-<54	12-<54#	12-<54#	12-<61	12-<61#	12-<61#	12-<67	12-<67#	12-<67#	12-<71
	12-<71#	12-<71#	12-<73	12-<73#	12-<73#	12-<75	12-<75#	12-<75#	12-<79	12-<79#	12-<79#	12-=00	12-=00#	12-=00#
	12-=08	12-=08#	12-=08#	12-=22	12-=22#	12-=22#	12-=29	12-=29#	12-=29#	12-=33	12-=33#	12-=33#	12-=35	12-=35#
	12-=35#	12-=37	12-=37#	12-=37#	12-=44	12-=44#	12-=44#	12-=50	12-=50#	12-=50#	12-=54	12-=54#	12-=54#	12-=56
	12-=56#	12-=56#	12-=58	12-=58#	12-=58#	12-=62	12-=62#	12-=62#	12-=83	12-=83#	12-=83#	12-=92	12-=92#	12-=92#
	12->05	12->05#	12->05#	12->13	12->13#	12->13#	12->17	12->17#	12->17#	12->19	12->19#	12->19#	12->21	12->21#
	12->21#	12->27	12->27#	12->27#	12->32	12->32#	12->32#	12->36	12->36#	12->36#	12->38	12->38#	12->38#	12->40
	12->40#	12->40#	12->44	12->44#	12->44#	12->67	12->67#	12->67#	12->80	12->80#	12->80#	12->87	12->87#	12->87#
	12->91	12->91#	12->91#	12->93	12->93#	12->93#	12->95	12->95#	12->95#	12-?13	12-?13#	12-?13#	12-?24	12-?24#
	12-?24#	12-?31	12-?31#	12-?31#	12-?35	12-?35#	12-?35#	12-?37	12-?37#	12-?37#	12-?39	12-?39#	12-?39#	12-?55
	12-?55#	12-?55#	12-?68	12-?68#	12-?68#	12-?75	12-?75#	12-?75#	12-?79	12-?79#	12-?79#	12-?81	12-?81#	12-?81#
	12-?83	12-?83#	12-?83#	12-@11	12-@11#	12-@11#	12-@24	12-@24#	12-@24#	12-@31	12-@31#	12-@31#	12-@35	12-@35#
	12-@35#	12-@37	12-@37#	12-@37#	12-@39	12-@39#	12-@39#	12-@60	12-@60#	12-@60#	12-@73	12-@73#	12-@73#	12-@80
	12-@80#	12-@80#	12-@84	12-@84#	12-@84#	12-@86	12-@86#	12-@86#	12-@88	12-@88#	12-@88#	12-A08	12-A08#	12-A08#
	12-A16	12-A16#	12-A16#	12-A28	12-A28#	12-A28#	12-A39	12-A39#	12-A39#	12-A46	12-A46#	12-A46#	12-A50	12-A50#
	12-A50#	12-A52	12-A52#	12-A52#	12-A54	12-A54#	12-A54#	12-A65	12-A65#	12-A65#	12-A78	12-A78#	12-A78#	12-A84
	12-A84#	12-A84#	12-A88	12-A88#	12-A88#	12-A90	12-A90#	12-A90#	12-A92	12-A92#	12-A92#	12-A96	12-A96#	12-A96#
	12-B15	12-B15#	12-B15#	12-B23	12-B23#	12-B23#	12-B31	12-B31#	12-B31#	12-B47	12-B47#	12-B47#	12-B54	12-B54#
	12-B54#	12-B58	12-B58#	12-B58#	12-B67	12-B67#	12-B67#	12-B71	12-B71#	12-B71#	12-B78	12-B78#	12-B78#	12-B85
	12-B85#	12-B85#	12-B89	12-B89#	12-B89#	12-B98	12-B98#	12-B98#	12-C14	12-C14#	12-C14#	12-C22	12-C22#	12-C22#
	12-C36	12-C36#	12-C36#	12-C43	12-C43#	12-C43#	12-C47	12-C47#	12-C47#	12-C49	12-C49#	12-C49#	12-C51	12-C51#
	12-C51#	12-C61	12-C61#	12-C61#	12-C67	12-C67#	12-C67#	12-C71	12-C71#	12-C71#	12-C78	12-C78#	12-C78#	12-C85
	12-C85#	12-C85#	12-D06	12-D06#	12-D06#	12-D14	12-D14#	12-D14#	12-D36	12-D36#	12-D36#	12-D43	12-D43#	12-D43#
	12-D47	12-D47#	12-D47#	12-D49	12-D49#	12-D49#	12-D51	12-D51#	12-D51#	12-D58	12-D58#	12-D58#	12-D64	12-D64#
	12-D64#	12-D68	12-D68#	12-D68#	12-D79	12-D79#	12-D79#	12-D92	12-D92#	12-D92#	12-E05	12-E05#	12-E05#	12-E17
	12-E17#	12-E17#	12-E38	12-E38#	12-E38#	12-E49	12-E49#	12-E49#	12-E69	12-E69#	12-E69#	12-E77	12-E77#	12-E77#
	12-E99	12-E99#	12-E99#	12-F06	12-F06#	12-F06#	12-F10	12-F10#	12-F10#	12-F12	12-F12#	12-F12#	12-F14	12-F14#
	12-F14#	12-F21	12-F21#	12-F21#	12-F27	12-F27#	12-F27#	12-F31	12-F31#	12-F31#	12-F38	12-F38#	12-F38#	12-F50
	12-F50#	12-F50#	12-F71	12-F71#	12-F71#	12-F82	12-F82#	12-F82#	12-F82#	12-F82#	12-F38	12-F38#	12-F38#	12-F50
FMT16	4-657#	12-187	12-189	12-198	12-296	12-389	12-514	12-610	12-703	12-825	12-921	12-:16	12-:11	12-<11
	12-<94	12-=78	12->60	12-?05	12-?91	12-?94	12-@04	12-@53	12-A03	12-B10	12-C57	12-C87	12-C99	12-D72
	12-E32	12-E64	12-F65	16-60	17-32	17-34	18-27	29-58	35-183					
FNCDTB	25-130	40-55#												
FNCMSK	4-504#	34-17	35-341											
GENBUF	12-106	12-207	12-305	12-398	12-523	12-619	12-712	12-836	12-932	12-:27	12-:20	12-<21	12-=04	12-=87
	12-A12	12-B19	12-C18	12-D10	12-E36	12-E73	12-F69	17-20#						
GET	12-67	12-131	12-152	12-173	12-232	12-254	12-276	12-330	12-351	12-371	12-423	12-444	12-467	12-496
	12-548	12-569	12-590	12-644	12-665	12-685	12-737	12-758	12-781	12-808	12-861	12-882	12-902	12-956
	12-977	12-997	12-:52	12-:73	12-:93	12-:47	12-:69	12-:89	12-<46	12-<67	12-=29	12-=50	12->13	12->32
	12->87	12-?31	12-?75	12-@31	12-@80	12-A46	12-A84	12-B54	12-B85	12-C43	12-C67	12-D43	12-D64	12-F06
	12-F27	15-47	15-90	15-114	15-144	15-168	15-199	16-76	16-97	16-129	16-163	16-192	20-31#	
GETBUF	6-0#	20-37												
GETINX	6-0#	12-67*	12-67*	12-493*	12-494*	12-805*	12-806*	19-13	20-38					
GETSTS	12-126	12-227	12-325	12-418	12-543	12-639	12-732	12-856	12-951	12-:47	12-:42	12-<43	12-=26	12->09
	12->84	12-?28	12-?72	12-@28	12-@77	12-A43	12-B51	12-B82	12-C40	12-D40	12-F03	15-46	16-75	19-10#
GNS	4-784	4-784	9-6	9-25	9-39	9-50	9-56	9-57	11-24	13-20	13-20	37-18	38-10	38-10
	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10
	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10
GO	4-503#	12-114	12-139	12-164	12-215	12-241	12-267	12-313	12-338	12-363	12-406	12-431	12-459	12-531
	12-556	12-581	12-627	12-652	12-677	12-720	12-745	12-773	12-844	12-869	12-894	12-939	12-964	12-989
	12-:35	12-:60	12-:85	12-:30	12-:56	12-:81	12-<29	12-<58	12-=12	12-=41	12-=96	12->25	12->64	12-?10
	12-?51	12-@07	12-@56	12-A06	12-A20	12-A32	12-A57	12-A70	12-B35	12-B75	12-C26	12-C59	12-D26	12-D55
	12-E40	12-E89	12-F18	12-F73	15-126	15-181	16-87	16-119	16-153	16-182	23-21	24-92	24-104	24-109
	24-125	25-51	25-54	35-342										
GTSWR	9-25	11-27	38-10#											
HCE	4-581#	4-590	12-E02	12-E09	12-F35	12-F42	16-225	25-379	25-384	25-399	25-401	25-414	25-429	35-348

35-378 35-381 35-391 35-394 40-77 40-78 40-81^N 40-85⁸ 40-86

SEQ 0310

LST

4-642#

31-82

34-70

C 9

SEQ 0312

SECRET

LSTRK	6-0# 29-53	11-47* 35-178	11-55*	12-:14	12-:23	12-=67	12->58	12-?03	12-002	16-58	16-364	16-385	17-50	26-62
MCLK	4-615#													
MCPE	4-725#	24-92	24-109	24-122	24-125	35-27	35-30							
MDF	4-620#													
MDPE	4-741#	25-454	35-438	35-441										
MEDENB	6-0#	11-43*	15-125*	16-40	16-284*									
MFGFIL	16-61	16-266*	16-274*	16-323	51-7#									
MI	4-623#													
MIXED	42-4#													
MOC	4-618#													
MOH	4-649#													
MOL	4-562# 33-158	9-92 34-30	29-146 35-88	29-156 35-252	29-157 35-505	29-162 35-506	29-167 35-528	29-195 35-533	32-37 36-35	32-40 36-36	33-68 36-40	33-145 36-45	33-146	33-153
MRD	4-616#													
MS	4-621#													
MSC	4-624#													
MSDRVS	10-116	39-28#												
MSE	4-692#	12-D70	17-29											
MSER	4-619#													
MSGDRV	9-98	39-29#												
MSHELP	10-27	39-9#												
MUR	4-617#													
MWD	4-640#	31-83	34-71											
MWP	4-622#													
MXF	4-740#	25-343	35-50	35-63	35-66									
NDTMSK	4-590#													
NED	4-737#	9-86	12-47	24-74	24-78	28-31								
NEM	4-738#	25-328												
NOP	4-507#													
NOTAVL	9-88	39-32#												
NOTPRS	9-85	39-31#												
NSA	4-647#													
OCC	4-628#													
OFD	4-660#	12-A98	12-B00											
OFFLIN	9-91	39-33#												
OFFSET	4-513#	12-A20	12-A57											
OM	4-569#	33-145	33-201	33-204	34-29	35-453	35-456							
ONES	12-521	12-617	12-710	42-21#										
ONLINE	39-34#													
OPE	4-683#													
OPI	4-575# 36-42 40-70 40-84	25-182 40-57 40-71 40-85	29-140 40-58 40-72 40-86	29-164 40-59 40-73 40-87	32-51 40-60 40-74 40-88	32-69 40-61 40-75 40-88	32-73 40-62 40-76	33-24 40-63 40-77	33-61 40-64 40-78	33-64 40-65 40-79	33-155 40-66 40-80	35-83 40-67 40-81	35-86 40-68 40-82	35-530 40-69 40-83
OR	4-742#													
PACACK	4-517#	16-119												
PAKACK	4-516#	4-517	15-126											
PAR	4-585#	24-137	24-141	24-153	29-30	33-24	33-29	33-34	35-41	35-46				
PAT	4-745#													
PDA	4-635#													
PGE	4-739#													
PGM	4-565#	34-29												
PHA	4-636#													
PIP	4-561# 36-72	15-178 36-77	16-150	29-156	29-177	29-182	33-145	33-216	33-221	34-29	35-505	35-511	35-516	36-35

PIRQ

4-491#

E 9

SEQ 0314

RMDT1 6-0# 12-67* 12-69 12-71 12-74 12-76 12-79¹ 9 12-81 49-5 49-17

SEQ 0318



RMDTO	6-0#													
RMEC1	4-711#													
RMEC11	6-0#	18-32	49-14											
RMEC10	6-0#													
RMEC2	4-712#	19-15												
RMEC21	6-0#	18-53	19-14	31-94	34-94	49-15								
RMEC20	6-0#													
RMER1	4-700#	24-145												
RMER11	6-0#	12-C75	12-C80	12-C81	12-D76	12-D81	12-D82	12-E02	12-E07	12-E08	12-F35	12-F40	12-F41	16-224
	18-23	18-25	24-137	24-152	24-154	25-80	25-156	25-183	25-226	25-241	25-284	25-361	25-387	25-402
	25-417	25-470	25-489	25-492	26-89	26-105	26-120	26-132	27-24	29-30	29-37	29-66	29-140	29-143
	29-164	31-71	32-51	32-55	32-57	32-58	32-69	32-71	32-72	32-83	32-85	32-86	32-97	32-99
	32-100	32-111	32-113	32-114	33-24	33-29	33-33	33-35	33-45	33-47	33-49	33-61	33-63	33-65
	33-78	33-80	33-82	33-155	33-232	33-236	33-238	33-240	33-250	33-252	33-254	33-264	33-268	33-270
	34-41	35-41	35-45	35-47	35-83	35-85	35-87	35-128	35-131	35-133	35-135	35-145	35-147	35-149
	35-159	35-161	35-163	35-189	35-279	35-282	35-286	35-288	35-297	35-299	35-301	35-311	35-313	35-314
	35-348	35-352	35-354	35-356	35-365	35-367	35-369	35-378	35-380	35-382	35-391	35-393	35-395	35-413
	35-415	35-417	35-423	35-479	35-481	35-483	35-530	36-42	49-13					
RMER10	6-0#	24-141												
RMER2	4-710#													
RMER21	6-0#	12-B60	12-B62	12-B63	12-B91	12-B93	12-B94	12-D89	12-D94	12-D95	15-55	16-84	16-220	24-139
	25-82	25-200	25-256	25-432	27-27	29-32	29-85	29-88	29-99	29-127	29-130	29-179	29-208	31-116
	33-31	33-92	33-99	33-101	33-103	33-116	33-118	33-120	33-130	33-132	33-134	33-171	33-218	33-266
	34-103	35-43	35-111	35-114	35-116	35-203	35-207	35-232	35-236	35-238	35-240	35-250	35-254	35-256
	35-266	35-268	35-270	35-284	35-466	35-468	35-470	35-513	35-546	36-58	36-74	36-88	36-94	36-96
	36-98	36-108	36-110	36-112	49-13									
RMER20	6-0#													
RMHR	4-708#													
RMHR1	6-0#													
RMHR0	6-0#													
RMLA	4-702#													
RMLA1	6-0#	49-16												
RMLAO	6-0#													
RMMR1	4-703#	12-B28	12-B38											
RMMR11	6-0#	31-81	34-69	49-17										
RMMR10	6-0#	12-B27*	12-B36*											
RMMR2	4-709#													
RMMR21	6-0#	31-103	34-81	49-17										
RMMR20	6-0#													
RMOF	4-706#	12-118	12-219	12-317	12-410	12-535	12-631	12-724	12-848	12-943	12-:39	12-:34	12-<33	12-=16
	12->00	12->72	12-?18	12-?62	12-@18	12-@67	12-A24	12-A61	12-B41	12-C30	12-D30	12-E44	12-E93	12-F77
	16-67													
RMOF1	6-0#	18-21	18-27	25-377	25-487	35-344	35-420	49-14	49-16					
RMOFO	6-0#	12-94*	12-187	12-189*	12-198*	12-296*	12-389*	12-514*	12-610*	12-703*	12-825*	12-921*	12-:16*	12-:11*
	12-<11*	12-<94*	12-=78*	12->60*	12-?05*	12-?48*	12-?91	12-?94*	12-@04*	12-@53*	12-A03*	12-A98	12-B00*	12-B10*
	12-C08*	12-C56*	12-C57*	12-C87	12-C99*	12-E32*	12-E64*	12-F65*	16-60*	17-32	29-58	35-183	49-11	
RMR	4-586#	32-51	32-83	32-87	33-232	33-250	33-253	35-128	35-159	35-162				
RMSN	4-705#													
RMSNI	6-0#	49-17												
RMSNO	6-0#													
RMWC	4-772#	12-11*	12-12	12-141	12-243	12-340	12-433	12-558	12-654	12-747	12-871	12-966	12-:62	12-:58
	12-<34	12-=17	12->01	12->74	12-?19	12-?63	12-@19	12-@68	12-A34	12-A73	12-B42	12-C31	12-D31	12-E45
	12-E94	12-F79	16-66											
RMWC1	6-0#	18-72	26-10	26-23	26-44	26-115	49-14							
RMWCO	6-0#	12-98*	12-199*	12-240*	12-266*	12-297*	12-390*	12-515*	12-611*	12-704*	12-828*	12-922*	12-:17*	12-:12*
	12-<13*	12-<96*	12-=79*	12->62*	12-?08*	12-?50*	12-@05*	12-@54*	12-A04*	12-B11*	12-C09*	12-D02*	12-E33*	12-E65*

12-F66* 16-59* 16-302 17-22 18-73 26-24 26-45^K 9

SEQ 0320

SW02

4-491

4-491#

M 9

SEQ 0322

TST20 12-<89# 38-6

B 10

SEQ 0324

TST21	12-72#	38-6												
TST22	12-754#	38-6												
TST23	12-700#	38-6												
TST24	12-744#	38-6												
TST25	12-799#	38-6												
TST26	12-749#	38-6												
TST27	12-798#	38-6												
TST3	12-63#	38-6												
TST30	12-B05#	38-6												
TST31	12-C03#	38-6												
TST32	12-C93#	38-6												
TST33	12-E56#	38-6												
TST4	12-91#	38-6												
TST5	12-193#	38-6												
TST6	12-291#	38-6												
TST7	12-384#	38-6												
TSTNMB	14-24*	14-25*	14-27	14-132#										
TSTPRP	12-110	12-211	12-309	12-402	12-527	12-623	12-716	12-840	12-935	12-:31	12-;24	12-<25	12-=08	12-=92
TSTQUE	12->67	12-?13	12-?55	12-@11	12-@60	12-A16	12-B23	12-B71	12-C22	12-D14	12-E38	12-E77	12-F71	15-38#
	6-0#	11-4	11-5*	11-42	11-50	12-2	12-34	12-63	12-91	12-193	12-291	12-384	12-509	12-605
	12-698	12-822	12-915	12-:10	12-:06	12-<06	12-<89	12-=72	12->54	12-?00	12-?44	12-?99	12-@49	12-@98
	12-B05	12-C03	12-C93	12-E56	13-11	13-13*	13-17	13-17*	28-24	30-17	31-56	34-53		
TYPBN	14-122	38-10#												
TYPDS	13-20	13-20	14-116	38-10#										
TYPE	9-6	9-25	9-39	9-45	9-50	9-56	9-57	9-97	9-98	9-100	9-107	10-11	10-14	10-17
	10-22	10-27	10-30	10-35	10-37	10-41	10-42	10-45	10-50	10-55	10-57	10-63	10-67	10-73
	10-79	10-83	10-94	10-100	10-113	10-114	10-116	10-121	10-127	10-138	10-139	10-144	10-152	11-24
	12-102	12-203	12-301	12-394	12-519	12-615	12-708	12-832	12-926	12-:21	12-;16	12-<17	12-=00	12-=83
	12-A08	12-B15	12-C14	12-D06	12-E69	13-20	13-20	13-20	14-21	14-22	14-26	14-31	14-33	14-38
	14-76	14-82	14-86	14-99	14-105	14-107	14-125	14-127	37-18	38-2	38-3	38-4	38-5	38-7
	38-7	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8
	38-8	38-8	38-8	38-8	38-8	38-10#	38-11							
TYPOC	9-8	10-56	14-119	38-8	38-10#									
TYPON	38-10#													
TYPOS	9-42	9-53	9-99	10-70	10-91	14-23	14-27	14-32	14-34	38-10#				
UO	4-749#													
U1	4-749#													
U2	4-749#													
UBUSQS	10-42	39-14#												
UNS	4-574#	31-72	32-51	32-55	32-59	33-232	33-264	33-269	35-279	35-282	35-287			
UNTMSK	4-753#	24-50	24-52											
UPE	4-736#	25-270												
USE	4-693#	12-D70	17-29											
USRFIL	16-254	16-257	16-267*	16-275*	16-326	51-12#								
VV	4-568#	15-122	16-116	25-196	29-91	29-156	29-157	29-206	29-211	32-23	32-26	33-106	33-145	33-146
	33-169	33-175	34-29	35-90	35-119	35-505	35-506	35-544	35-549	36-35	36-36	36-56	36-61	
WC	4-638#	31-82	34-70											
WCD	4-524#													
WCE	4-735#	12-475	12-481	12-789	12-794	25-169	40-77	40-78						
WCEHI	4-760#													
WCELO	4-761#													
WCF	4-583#	4-590	25-240	35-479	35-482									
WCH	4-525#	12-363	12-459	12-677	12-773	12-894	12-989	12-:85	12-;81					
WD	4-528#													
WH	4-529#	12-100	12-139	12-201	12-241	12-299	12-338	12-392	12-431	12-517	12-556	12-613	12-652	12-706
	12-745	12-830	12-869	12-924	12-964	12-:19	12-:60	12-;14	12-;56	12-<15	12-<29	12-<98	12-=12	12-=81

12--96 12-A06 12-A32 12-B13 12-B35 12-C12 12-C26^{D 10} 12-D04 12-D26 12-E35 12-E40 12-E67 12-E89 12-F68
SEQ 0326

WLE	12-F73	35-342											
WRL	4-577#	4-590	25-221	25-225	25-238	25-253	25-268	35-279	35-311	35-315	35-323	40-81	40-82
XSIZ	4-563#	25-223	35-318										
XXDP	9-67#	10-123	10-153										
ZEROS	6-0#	9-30*	9-33*	9-34	9-36*	9-41	9-52	9-77	9-79				
	12-104	12-205	12-303	12-396	12-834	12-930	12-:25	12-;18	12-C16	12-D08	12-E71	42-38#	

SSCMRE	4-791#													
SSCMTM	4-791#	5-0	5-0	5-0	5-0	5-0								
SSESCA	4-491#													
SSNEWT	4-491#	12-2	12-34	12-63	12-91	12-193	12-291	12-384	12-509	12-605	12-698	12-822	12-915	12-:10
SSSET	12-:06	12-<06	12-<89	12-=72	12->54	12-?00	12-?44	12-?99	12-@49	12-@98	12-B05	12-C03	12-C93	12-E56
SSSETM	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10#
SSSKIP	9-19	9-19#												
.SACT1	4-491#													
.SACT1	4-483#	4-786												
.SAPT8	4-483#	5-0	5-0#											
.SAPTH	4-483#	4-789												
.SAPTY	4-483#	38-12												
.SCATC	4-479#	4-784												
.SCMTA	4-480#	4-791												
.SEOP	4-480#	13-20												
.SERRO	4-480#	38-7												
.SERRT	4-480#													
.SPOWE	4-482#	38-11												
.SRDDE	4-481#													
.SRDOC	4-481#	38-9												
.SREAD	4-481#	38-8												
.SSAVE	4-482#	38-1												
.SSCOP	4-480#	38-6												
.SSIZE	4-482#													
.STRAP	4-482#	38-10												
.STYPB	4-481#	38-2												
.STYPD	4-481#	38-3												
.STYPE	4-480#	38-5												
.STYPO	4-481#	38-4												
.EQUAT	4-479#	4-491												
.HEADE	4-479#	4-487												
.SETUP	4-479#	4-782												
.SWRHI	4-479#	4-488												
.SWRLO	4-479#	4-488#	4-489											
CALCLR	4-194#	12-36	12-65											
CALSUB	4-208#	12-102	12-110	12-122	12-131	12-135	12-146	12-152	12-156	12-158	12-160	12-167	12-173	12-177
	12-179	12-181	12-185	12-203	12-211	12-223	12-232	12-236	12-248	12-254	12-258	12-260	12-262	12-270
	12-276	12-280	12-282	12-284	12-288	12-301	12-309	12-321	12-330	12-334	12-345	12-351	12-355	12-357
	12-359	12-365	12-371	12-375	12-377	12-379	12-394	12-402	12-414	12-423	12-427	12-438	12-444	12-448
	12-450	12-452	12-461	12-467	12-471	12-478	12-496	12-519	12-527	12-539	12-548	12-552	12-563	12-569
	12-573	12-575	12-577	12-584	12-590	12-594	12-596	12-598	12-602	12-615	12-623	12-635	12-644	12-648
	12-659	12-665	12-669	12-671	12-673	12-679	12-685	12-689	12-691	12-693	12-708	12-716	12-728	12-737
	12-741	12-752	12-758	12-762	12-764	12-766	12-775	12-781	12-785	12-792	12-808	12-832	12-840	12-852
	12-861	12-865	12-876	12-882	12-886	12-888	12-890	12-896	12-902	12-906	12-908	12-910	12-926	12-935
	12-947	12-956	12-960	12-971	12-977	12-981	12-983	12-985	12-991	12-997	12-:01	12-:03	12-:05	12-:21
	12-:31	12-:43	12-:52	12-:56	12-:67	12-:73	12-:77	12-:79	12-:81	12-:87	12-:93	12-:97	12-:99	12-:01
	12-:16	12-:24	12-:38	12-:47	12-:51	12-:63	12-:69	12-:73	12-:75	12-:77	12-:83	12-:89	12-:93	12-:95
	12-:97	12-<17	12-<25	12-<39	12-<46	12-<50	12-<52	12-<54	12-<61	12-<67	12-<71	12-<73	12-<75	12-<79
	12-=00	12-=08	12-=22	12-=29	12-=33	12-=35	12-=37	12-=44	12-=50	12-=54	12-=56	12-=58	12-=62	12-=83
	12-=92	12->05	12->13	12->17	12->19	12->21	12->27	12->32	12->36	12->38	12->40	12->44	12->67	12->80
	12->87	12->91	12->93	12->95	12-?13	12-?24	12-?31	12-?35	12-?37	12-?39	12-?55	12-?68	12-?75	12-?79
	12-?81	12-?83	12-@11	12-@24	12-@31	12-@35	12-@37	12-@39	12-@60	12-@73	12-@80	12-@84	12-@86	12-@88
	12-A08	12-A16	12-A28	12-A39	12-A46	12-A50	12-A52	12-A54	12-A65	12-A78	12-A84	12-A88	12-A90	12-A92
	12-A96	12-B15	12-B23	12-B31	12-B47	12-B54	12-B58	12-B67	12-B71	12-B78	12-B85	12-B89	12-B98	12-C14
	12-C22	12-C36	12-C43	12-C47	12-C49	12-C51	12-C61	12-C67	12-C71	12-C78	12-C85	12-D06	12-D14	12-D36
	12-D43	12-D47	12-D49	12-D51	12-D58	12-D64	12-D68	12-D79	12-D92	12-E05	12-E17	12-E38	12-E49	12-E69

	12-E77	12-E99	12-F06	12-F10	12-F12	12-F14	12-F21	12-F27	12-F31	12-F38	12-F50	12-F71	12-F82	
COMMEN	12-E77													
ENDCOM	4-491#													
ERROR	4-491#													
	4-12#	11-23	12-26	12-36	12-49	12-53	12-59	12-65	12-67	12-86	12-102	12-110	12-122	12-131
	12-135	12-146	12-152	12-156	12-158	12-160	12-167	12-173	12-177	12-179	12-181	12-185	12-203	12-211
	12-223	12-232	12-236	12-248	12-254	12-258	12-260	12-262	12-270	12-276	12-280	12-282	12-284	12-288
	12-301	12-309	12-321	12-330	12-334	12-345	12-351	12-355	12-357	12-359	12-365	12-371	12-375	12-377
	12-379	12-394	12-402	12-414	12-423	12-427	12-438	12-444	12-448	12-450	12-452	12-461	12-467	12-471
	12-478	12-483	12-496	12-502	12-506	12-519	12-527	12-539	12-548	12-552	12-563	12-569	12-573	12-575
	12-577	12-584	12-590	12-594	12-596	12-598	12-602	12-615	12-623	12-635	12-644	12-648	12-659	12-665
	12-669	12-671	12-673	12-679	12-685	12-689	12-691	12-693	12-708	12-716	12-728	12-737	12-741	12-752
	12-758	12-762	12-764	12-766	12-775	12-781	12-785	12-792	12-796	12-808	12-814	12-819	12-832	12-840
	12-852	12-861	12-865	12-876	12-882	12-886	12-888	12-890	12-896	12-902	12-906	12-908	12-910	12-926
	12-935	12-947	12-956	12-960	12-971	12-977	12-981	12-983	12-985	12-991	12-997	12-:01	12-:03	12-:05
	12-:21	12-:31	12-:43	12-:52	12-:56	12-:67	12-:73	12-:77	12-:79	12-:81	12-:87	12-:93	12-:97	12-:99
	12-:01	12-:16	12-:24	12-:38	12-:47	12-:51	12-:63	12-:69	12-:73	12-:75	12-:77	12-:83	12-:89	12-:93
	12-:95	12-:97	12-<17	12-<25	12-<39	12-<46	12-<50	12-<52	12-<54	12-<61	12-<67	12-<71	12-<73	12-<75
	12-<79	12-=00	12-=08	12-=22	12-=29	12-=33	12-=35	12-=37	12-=44	12-=50	12-=54	12-=56	12-=58	12-=62
	12-=83	12-=92	12->05	12->13	12->17	12->19	12->21	12->27	12->32	12->36	12->38	12->40	12->44	12->67
	12->80	12->87	12->91	12->93	12->95	12-?13	12-?24	12-?31	12-?35	12-?37	12-?39	12-?55	12-?68	12-?75
	12-?79	12-?81	12-?83	12-a11	12-a24	12-a31	12-a35	12-a37	12-a39	12-a60	12-a73	12-a80	12-a84	12-a86
	12-a88	12-A08	12-A16	12-A28	12-A39	12-A46	12-A50	12-A52	12-A54	12-A65	12-A78	12-A84	12-A88	12-A90
	12-A92	12-A96	12-B15	12-B23	12-B31	12-B47	12-B54	12-B58	12-B65	12-B67	12-B71	12-B78	12-B85	12-B89
	12-B96	12-B98	12-C14	12-C22	12-C36	12-C43	12-C47	12-C49	12-C51	12-C61	12-C67	12-C71	12-C78	12-C83
	12-C85	12-D06	12-D14	12-D36	12-D43	12-D47	12-D49	12-D51	12-D58	12-D64	12-D68	12-D79	12-D84	12-D92
	12-D97	12-E05	12-E12	12-E17	12-E38	12-E49	12-E69	12-E77	12-E99	12-F06	12-F10	12-F12	12-F14	12-F21
	12-F27	12-F31	12-F38	12-F45	12-F50	12-F71	12-F82							
ESCAPE	4-491#													
GETPRI	4-491#													
GETREG	4-171#	12-67												
GETSWR	4-491#	9-25	9-25#	11-27										
MSG	4-8#													
MULT	4-491#													
NEWTST	4-491#	12-2	12-34	12-63	12-91	12-193	12-291	12-384	12-509	12-605	12-698	12-822	12-915	12-:10
	12-:06	12-<06	12-<89	12-=72	12->54	12-?00	12-?44	12-?99	12-a49	12-a98	12-B05	12-C03	12-C93	12-E56
NWTST	4-44#	12-2	12-34	12-63	12-91	12-193	12-291	12-384	12-509	12-605	12-698	12-822	12-915	12-:10
	12-:06	12-<06	12-<89	12-=72	12->54	12-?00	12-?44	12-?99	12-a49	12-a98	12-B05	12-C03	12-C93	12-E56
POP	4-491#	12-19	12-20	12-24	12-25	12-45	12-46	12-57	12-58	16-71	16-279	16-289	16-399	17-65
	18-95	19-25	19-26	19-27	20-70	21-59	22-20	22-21	23-35	24-147	24-151	25-131	28-51	30-25
	31-60	34-58	34-59	38-1	38-3	38-9	38-11	38-11	38-12	38-12				
PUSH	4-491#	12-5	12-6	12-38	12-39	16-47	16-263	16-282	16-298	17-20	18-17	19-10	19-11	19-12
	20-35	21-29	22-4	22-5	23-9	24-143	25-128	28-20	30-12	31-54	34-51	34-52	38-1	38-3
	38-9	38-11	38-11	38-12	38-12	38-12								
PUTREG	4-449#													
REPORT	4-491#													
RGBFMC	4-77#	6-0	6-0											
SETPRI	4-491#	9-23	11-41	37-6	37-10	38-8								
SETTRA	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10	38-10#
SETUP	4-491#	9-19												
SKIP	4-491#													
SLASH	4-491#													
STARS	4-491#	4-786	4-789	4-787	4-789	5-0	5-0	5-0	12-2	12-2	12-34	12-34	12-63	12-63
	12-91	12-91	12-193	12-193	12-291	12-291	12-384	12-384	12-509	12-509	12-605	12-605	12-698	12-698
	12-822	12-822	12-915	12-915	12-:10	12-:10	12-:06	12-:06	12-<06	12-<06	12-<89	12-<89	12-=72	12-=72
	12->54	12->54	12-?00	12-?00	12-?44	12-?44	12-?99	12-?99	12-a49	12-a49	12-a98	12-a98	12-B05	12-B05
	12-C03	12-C03	12-C93	12-C93	12-E28	12-E30	12-E56	12-E56	12-F61	12-F63	13-20	14-2	15-57	15-70

15-84 15-107 15-139 15-161 15-194 16-211 16-291^{H 10} 16-295 24-2 25-26 25-34 25-119 26-1 26-3
SEQ 0330

	31-2	38-1	38-2	38-3	38-4	38-5	38-6	38-7	38-8	38-8	38-8	38-8	38-8	38-9
SWRSU	38-10	38-11	38-11	38-12										
TAGS	4-491#	9-19	9-19#											
TRMTRP	4-111#	5-0												
TYPBIN	38-10#													
TYPDEC	4-491#	13-20	13-20											
TYPNAM	4-479#	4-491#	9-25											
TYPNUM	4-491#													
TYPOCS	4-491#	9-99	14-23	14-27	14-32	14-34								
TYPOCT	4-491#	10-56	38-8											
TYPTXT	4-491#	9-6	9-39	9-50	9-56	9-57	11-24	13-20	13-20	37-18				
XPER	4-20#	7-4	7-7	7-10	7-13	7-16	7-19	7-22	7-25	7-28	7-31	7-34	7-37	7-40
	7-43	7-46	7-49	7-52	7-55	7-58	7-61	7-64	7-67	7-70	7-73	7-76	7-79	7-82
	7-85	7-88	7-91	7-94	7-97	7-100	7-103	7-106	7-109	7-112	7-115	7-118	7-121	7-124
	7-127	7-131	7-134	7-137	7-140	7-143	7-147	7-151	7-155	7-158	7-161	7-164	7-167	7-170
	7-173	7-176	7-179	7-183	7-186	7-189	7-192	7-195	7-198	7-201	7-204	7-207	7-210	7-213
	7-216	7-219	7-222	7-225	7-228	7-231	7-234	7-237	7-240	7-243	7-246	7-249	7-252	7-255
	7-258	7-261	7-264	7-267	7-270	7-273	7-276	7-279	7-282	7-285	7-288	7-291	7-294	7-297
	7-300	7-303	7-306	7-309	7-312	7-315	7-318	7-321	7-324	7-327	7-330	7-333	7-336	7-339
	7-342	7-345	7-348	7-352	7-355	7-358	7-361	7-364	7-367	7-370	7-373	7-376	7-379	7-382
	7-385	7-388	7-391	7-394	7-397	7-400	7-403	7-406	7-409	7-412	7-415	7-418	7-421	7-424
	7-427	7-430	7-433	7-436	7-439	7-442	7-445	7-448	7-451	7-454	7-457	7-460	7-463	7-466
	7-469	7-472	7-475	7-478	7-481	7-484	7-487	7-490	7-493	7-496	7-499	7-502	7-505	7-508
	7-511	7-514	7-517	7-520	7-523	7-526	7-529	7-532	7-535	7-538	7-541	7-544	7-547	7-550
	7-553	7-557	7-560	7-564	7-567	7-570	7-574	7-578	7-582	7-587	7-591	7-595	7-598	7-601
	7-604	7-607	7-610	7-613	7-616	7-619	7-622	7-625	7-628	7-631	7-634	7-637	7-640	7-643
	7-646	7-649	7-652	7-655	7-658	7-661	7-664	7-667	7-670	7-673	7-676	7-679	7-682	7-685
	7-688	7-691	7-694	7-697	7-700	7-703	7-706	7-709	7-712	7-715	7-718	7-721	7-724	